

Oracle® E-Business Suite
Integrated SOA Gateway開発者ガイド
リリース12.2
部品番号F26893-01

2019年2月

Oracle E-Business Suite Integrated SOA Gateway 開発者ガイド リリース 12.2

部品番号 F26893-01

Copyright © 2008, 2019, Oracle and/or its affiliates. All rights reserved.

原著者: Melody Yang

原著協力者: Rekha Ayothi, Sudipto Chakraborty, Rajesh Ghosh, Vardhan Kale, Megha Mathpal, Ravindra Nadakuditi, Saritha Nalagandla, Dilbagh Sardar, Vijay Shanmugam, Shivdas Tomar, Abhishek Verma

このソフトウェアおよび関連ドキュメントの使用と開示は、ライセンス契約の制約条件に従うものとし、知的財産に関する法律により保護されています。ライセンス契約で明示的に許諾されている場合もしくは法律によって認められている場合を除き、形式、手段に関係なく、いかなる部分も使用、複写、複製、翻訳、放送、修正、ライセンス供与、送信、配布、発表、実行、公開または表示することはできません。このソフトウェアのリパース・エンジニアリング、逆アセンブル、逆コンパイルは互換性のために法律によって規定されている場合を除き、禁止されています。

ここに記載された情報は予告なしに変更される場合があります。また、誤りが無いことの保証はいたしかねます。誤りを見つけた場合は、オラクルまでご連絡ください。

このソフトウェアまたは関連ドキュメントを、米国政府機関もしくは米国政府機関に代わってこのソフトウェアまたは関連ドキュメントをライセンスされた者に提供する場合は、次の通知が適用されます。

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

このソフトウェアまたはハードウェアは様々な情報管理アプリケーションでの一般的な使用のために開発されたものです。このソフトウェアまたはハードウェアは、危険が伴うアプリケーション(人的傷害を発生させる可能性があるアプリケーションを含む)への用途を目的として開発されていません。このソフトウェアまたはハードウェアを危険が伴うアプリケーションで使用する際、安全に使用するために、適切な安全装置、バックアップ、冗長性(redundancy)、その他の対策を講じることは使用者の責任となります。このソフトウェアまたはハードウェアを危険が伴うアプリケーションで使用したことに起因して損害が発生しても、Oracle Corporation およびその関連会社は一切の責任を負いかねます。

Oracle および Java はオラクルおよびその関連会社の登録商標です。その他の社名、商品名等は各社の商標または登録商標である場合があります。

Intel, Intel Xeon は、Intel Corporation の商標または登録商標です。すべての SPARC の商標はライセンスをもとに使用し、SPARC International, Inc.の商標または登録商標です。AMD、Opteron、AMD ロゴ、AMD Opteron ロゴは、Advanced Micro Devices, Inc.の商標または登録商標です。UNIX は、The Open Group の登録商標です。

このソフトウェアまたはハードウェア、そしてドキュメントは、第三者のコンテンツ、製品、サービスへのアクセス、あるいはそれらに関する情報を提供することがあります。適用されるお客様と Oracle Corporation との間の契約に別段の定めがある場合を除いて、Oracle Corporation およびその関連会社は、第三者のコンテンツ、製品、サービスに関して一切の責任を負わず、いかなる保証もいたしません。適用されるお客様と Oracle Corporation との間の契約に定めがある場合を除いて、Oracle Corporation およびその関連会社は、第三者のコンテンツ、製品、サービスへのアクセスまたは使用によって損失、費用、あるいは損害が発生しても一切の責任を負いかねます。

オラクルのアクセシビリティについての詳細情報は、Oracle Accessibility Program の Web サイト (<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>)を参照してください。

サポートをご契約のお客様には、My Oracle Support を通して電子支援サービスを提供しています。詳細情報は (<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info>)か、聴覚に障害のあるお客様は (<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs>)を参照してください。

目次

ご意見をお寄せください

はじめに

1 Oracle E-Business Suite Integrated SOA Gateway の概要	1-1
Oracle E-Business Suite Integrated SOA Gateway の概要	1-1
主要なコンポーネントの機能および定義	1-2
2 統合インタフェースおよびサービスの検出および表示	
概要	2-1
統合インタフェースの検索および表示	2-1
インタフェース詳細のレビュー	2-5
SOAP Web サービスの生成	2-6
WSDL 要素詳細のレビュー	2-8
WADL 要素詳細のレビュー	2-18
SOAP メッセージについて	2-25
REST メッセージについて	2-44
3 Web サービスとしての PL/SQL API の使用	
概要	3-1
PL/SQL SOAP サービスの使用	3-2
BPEL プロセスによる SOA コンポジット・アプリケーションからの同期 Web サービスの起動	3-6
BPEL プロセスによる SOA コンポジット・アプリケーションの作成	3-7
Web サービスのパートナ・リンクの作成	3-10
ファイル・アダプタのパートナ・リンクの追加	3-11
Invoke アクティビティの追加	3-19
Assign アクティビティの追加	3-23

同期 BPEL プロセスによる SOA コンポジットのデプロイおよびテスト	3-37
BPEL プロセスによる SOA コンポジットのデプロイ	3-38
SOA コンポジット・アプリケーションのテスト	3-43
BPEL プロセスによる SOA コンポジット・アプリケーションからの非同期 Web サービスの起動	3-46
BPEL プロセスによる SOA コンポジット・アプリケーションの作成	3-49
パートナ・リンクの追加	3-52
Invoke アクティビティの追加	3-53
受入アクティビティの追加	3-55
Assign アクティビティの追加	3-57
非同期 BPEL プロセスによる SOA コンポジットのデプロイおよびテスト	3-63
BPEL プロセスによる SOA コンポジットのデプロイ	3-64
SOA コンポジット・アプリケーションのテスト	3-69
PL/SQL REST サービスの使用	3-72
HTTP Basic 認証および REST ヘッダー付き XML ペイロードを使用した REST サービスの起動	3-73
PL/SQL REST Web サービスの配置	3-74
配置済 WADL からのリソース情報の記録	3-76
Java クラスを含むプロジェクトの作成	3-77
Java クラスを使用した REST サービスの起動	3-83
トークン・ベース認証および JSON ペイロードを使用した REST サービスの起動	3-83
PL/SQL REST Web サービスの配置	3-85
配置済 WADL URL の記録	3-87
Java クラスを含むプロジェクトの作成	3-88
Java クライアントを使用した REST サービスの起動	3-99

4 REST サービスとしての Java API の使用

概要	4-1
HTTP GET メソッドを使用した Java Bean サービスの起動	4-2
REST サービスの配置	4-2
セキュリティ付与の作成	4-4
配置済 WADL からのリソース情報の記録	4-6
Java クラスを含むプロジェクトの作成	4-9
Java クラスを使用した REST サービスの起動	4-15
カスタム Java Bean サービスへの注釈付けと起動	4-16
カスタム Java API の作成およびコンパイル	4-17
カスタム Java クラスおよびソース・ファイルの配置	4-34
注釈付けされたカスタム Java Bean サービスの解析と統合リポジトリへのアップロード	4-34
カスタム Java Bean サービスの配置	4-36
セキュリティ付与の作成	4-37

配置済 WADL からのリソース情報の記録.....	4-38
Javascript を使用して HTML からのカスタム REST サービスを起動する	4-41
トークン・ベース認証および XML ペイロードを使用したアプリケーション・モジュール・サービスの 起動	4-48
REST サービスの配置	4-49
配置済 WADL URL の記録.....	4-50
セキュリティ付与の作成	4-51
Java クラスを含むプロジェクトの作成	4-52
Java クラスを使用した REST サービスの起動	4-60

5 XML ゲートウェイ・インバウンドおよびアウトバウンド・インタフェースの使用

概要	5-1
XML ゲートウェイ・インバウンド・サービスの使用	5-2
設計時における XML ゲートウェイ・インバウンド・サービスの使用	5-2
新しい BPEL プロセスによる SOA コンポジット・アプリケーションの作成.....	5-7
パートナ・リンクの作成	5-11
ファイル・アダプタのパートナ・リンクの追加	5-12
Invoke アクティビティの追加	5-16
Assign アクティビティの追加	5-17
実行時における BPEL プロセスによる SOA コンポジットのデプロイおよびテスト.....	5-22
BPEL プロセスによる SOA コンポジットのデプロイ	5-22
BPEL プロセスによる SOA コンポジット・アプリケーションのテスト	5-26
サブスクリプション・モデルを介した XML ゲートウェイ・アウトバウンドの使用	5-32
設計時における XML ゲートウェイ・アウトバウンド・サービスの使用	5-32
新しい BPEL プロセスによる SOA コンポジット・アプリケーションの作成.....	5-36
AQ アダプタのパートナ・リンクの作成	5-37
受入アクティビティの追加	5-46
ファイル・アダプタのパートナ・リンクの追加	5-48
Invoke アクティビティの追加	5-51
Assign アクティビティの追加	5-53
実行時における BPEL プロセスによる SOA コンポジット・アプリケーションのデプロイおよびテスト.....	5-54
BPEL プロセスによる SOA コンポジット・アプリケーションのデプロイ.....	5-55
BPEL プロセスによる SOA コンポジット・アプリケーションのテスト	5-58

6 サブスクリプション・モデルを介したビジネス・イベントの使用

概要	6-1
設計時に BPEL プロセスによる SOA コンポジット・アプリケーションを作成する場合のビジネス・イ ベントの使用	6-2
新しい BPEL プロセスによる SOA コンポジット・アプリケーションの作成.....	6-3

AQアダプタのパートナー・リンクの作成	6-5
受入アクティビティの追加	6-11
ファイル・アダプタのパートナー・リンクの追加.....	6-13
Invoke アクティビティの追加	6-19
Assign アクティビティの追加	6-21
実行時における BPEL プロセスによる SOA コンポジット・アプリケーションのデプロイおよびテスト	6-22
BPEL プロセスによる SOA コンポジット・アプリケーションのデプロイ.....	6-23
BPEL プロセスによる SOA コンポジット・アプリケーションのテスト	6-26

7 コンカレント・プログラムの使用

概要	7-1
設計時におけるコンカレント・プログラム WSDL の使用	7-2
新しい BPEL プロセスによる SOA コンポジット・アプリケーションの作成	7-6
Web サービスのパートナー・リンクの作成	7-8
ファイル・アダプタのパートナー・リンクの追加	7-9
Invoke アクティビティの追加	7-14
Assign アクティビティの追加	7-16
実行時における BPEL プロセスによる SOA コンポジットのデプロイおよびテスト.....	7-22
BPEL プロセスによる SOA コンポジット・アプリケーションのデプロイ.....	7-23
BPEL プロセスによる SOA コンポジット・アプリケーションのテスト	7-26

8 オープン・インタフェース REST サービスの使用

概要	8-1
オープン・インタフェース表 REST サービスの使用	8-1
REST サービスの配置	8-2
セキュリティ付与の作成	8-3
配置済 WADL からのリソース情報の記録.....	8-4
コマンドラインを使用した REST サービスの起動	8-4

9 ビジネス・サービス・オブジェクトの使用

概要	9-1
設計時におけるビジネス・サービス・オブジェクト WSDL の使用	9-2
BPEL プロセスによる SOA コンポジット・アプリケーションの作成.....	9-6
パートナー・リンクの作成.....	9-9
ファイル・アダプタのパートナー・リンクの追加	9-9
Invoke アクティビティの追加	9-14
Assign アクティビティの追加	9-18

実行時における BPEL プロセスによる SOA コンポジットのデプロイおよびテスト	9-26
BPEL プロセスによる SOA コンポジットのデプロイ	9-26
BPEL プロセスによる SOA コンポジット・アプリケーションのテスト	9-29
10 コンポジット・サービス - BPEL の使用	
概要	10-1
コンポジット・サービス - BPEL の表示	10-2
コンポジット・サービス - BPEL のダウンロード	10-2
BPEL プロセスの変更およびデプロイ	10-3
11 カスタム統合インタフェースの作成および使用	
概要	11-1
カスタム統合インタフェースの作成	11-2
ネイティブ・インタフェース・タイプのカスタム統合インタフェースの作成	11-2
カスタム・コンポジット・サービス - BPEL の作成	11-9
ワークフロー XML ロダーを使用したカスタム・ビジネス・イベントの作成	11-15
Web サービスとしてのカスタム統合インタフェースの使用	11-23
設計時に BPEL プロセスによる SOA コンポジット・アプリケーションを作成する場合のカスタム・	
インタフェース WSDL の使用	11-24
新しい BPEL プロセスによる SOA コンポジット・アプリケーションの作成	11-26
Web サービスのパートナー・リンクの作成	11-28
ファイル・アダプタのパートナー・リンクの追加	11-28
Invoke アクティビティの追加	11-33
Assign アクティビティの追加	11-34
実行時における BPEL プロセスによる SOA コンポジットのデプロイおよびテスト	11-39
BPEL プロセスによる SOA コンポジットのデプロイ	11-39
BPEL プロセスによる SOA コンポジット・アプリケーションのテスト	11-42
12 Oracle Workflow ビジネス・イベント・システムを使用した Web サービスの起動	
Oracle Workflow およびサービス起動フレームワークの概要	12-1
サービス起動フレームワークを使用した Web サービス起動	12-2
WSDL のメッセージ・パターンについて	12-3
Web サービス起動メタデータの定義	12-5
ステップ 1: Web サービス起動側ビジネス・イベントの作成	12-6
ステップ 2: 起動側イベントに対するローカル・サブスクリプションおよびエラー・イベント・サブ	
スクリプションの作成	12-7
ステップ 3: 受入イベントおよびサブスクリプションの作成(オプション)	12-14
Web サービス入力メッセージ・パートについて	12-16
WS セキュリティのサポート	12-21
Web サービス応答での Oracle E-Business Suite へのコールバック	12-27

Web サービスの起動.....	12-28
エラーの管理	12-35
Web サービス起動のテスト.....	12-36
Web サービス起動失敗のトラブルシューティング	12-40
シード Java ルール関数の拡張	12-46
その他の起動使用の考慮事項	12-53

A 統合リポジトリ・アノテーションの標準

一般的なガイドライン	A-1
Java アノテーション	A-4
PL/SQL アノテーション.....	A-17
コンカレント・プログラム・アノテーション	A-23
XML ゲートウェイ・アノテーション	A-25
ビジネス・イベント・アノテーション.....	A-35
ビジネス・エンティティ・アノテーション・ガイドライン	A-41
コンポジット・サービス - BPEL アノテーション・ガイドライン	A-115
アノテーション用語集.....	A-120

B サーバー接続の構成

アプリケーションサーバー接続.....	B-1

C サンプル・ペイロード

仕入先出荷およびデビット要求の作成のためのサンプル・ペイロード	C-1
インバウンド・プロセス発注 XML 取引のためのサンプル・ペイロード	C-3

用語集

索引

ご意見をお寄せください

Oracle E-Business Suite Integrated SOA Gateway 開発者ガイド リリース 12.2 部品番号 F26893-01

このマニュアルの品質および利便性について、お客様からのご意見、ご提案をお待ちしております。お寄せ頂いたご意見は、弊社製品をご利用のお客様により満足していただくための参考とさせていただきます。例:

- 実装手順が正確に漏れなく記載されていましたか?
- 手順の脈略やつながりを理解できましたか?
- 記載内容に誤りはありませんでしたか?
- 情報の構成はタスクの実施に役立ちましたか?
- 異なる情報や画像が必要でしたか? その場合、それはどのような形式の情報ですか?
- 正しい例が使用されていましたか? 他の例が必要でしたか?

記載内容に誤りを見つけた場合またはご意見をお寄せいただく際は、お客様のお名前、製品のライセンス所有者名(会社名)、マニュアルのタイトル、部品番号、章番号、項番号およびページ番号(該当する場合)をお知らせください。

注意: ご報告いただく前に、マニュアルの最新版で問題箇所が訂正されていないかどうかをご確認ください。
www.oracle.com にある新しい Oracle E-Business Suite リリース・オンライン・ドキュメント CD をご利用ください。
ここには、最新のドキュメント・ライブラリと最近リリースまたは改訂されたすべてのドキュメントが収録されています。

ご意見は、次の E メール・アドレス宛にお送りください。 appsdoc_us@oracle.com

お客様のお名前、住所、電子メール・アドレス、電話番号(任意)をご記載ください。

Oracle ソフトウェアに関するサポートをご希望の場合は、お客様のサポート担当者または Oracle サポート・サービスにお問合せください。

Oracle ソフトウェアのトレーニングまたは指導をご希望のお客様は、Oracle University 製品について弊社の現地事業所にお問合せください。オラクル社の事業所の一覧は、弊社 Web サイト(www.oracle.com)でご確認いただけます。

はじめに

対象読者

*Oracle E-Business Suite Integrated SOA Gateway 開発者ガイド*のリリース 12.2 へようこそ。

このマニュアルは、次の内容に関する実務的な知識があることを前提としています。

- 該当するビジネス・エリアにおける原則および慣行。
- コンピュータ・デスクトップ・アプリケーションの使用および用語。
- Oracle E-Business Suite の統合インタフェース。
- B2B、A2A および BP 統合。

このドキュメントは、Oracle E-Business Suite についての知識があることを前提としています。これは、これらのアプリケーションの機能的な要件を監督し、その機能をユーザーに対して配置する技術コンサルタント、実装担当者およびシステム統合コンサルタント向けに記述されています。

Oracle E-Business Suite を使用したことがない場合は、Oracle University で提供されている、Oracle E-Business Suite の 1 つ以上の研修クラスに参加することをお勧めします。

Oracle E-Business Suite 製品の詳細は、関連情報の入手元(ページ xii)を参照してください。

ドキュメントのアクセシビリティについて

オラクルのアクセシビリティについての詳細情報は、Oracle Accessibility Program の Web サイト <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc> を参照してください。

Oracle Support へのアクセス

サポートをご契約のお客様には、My Oracle Support を通して電子支援サービスを提供しています。詳細情報は、(<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info>)か、聴覚に障害のあるお客様は (<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs>)を参照してください。

構成

- 1 Oracle E-Business Suite Integrated SOA Gateway の概要
 - 2 統合インタフェースおよびサービスの検出および表示
 - 3 Web サービスとしての PL/SQL API の使用
 - 4 REST サービスとしての Java API の使用
 - 5 XML ゲートウェイ・インバウンドおよびアウトバウンド・インタフェースの使用
 - 6 サブスクリプション・モデルを介したビジネス・イベントの使用
 - 7 コンカレント・プログラムの使用
 - 8 オープン・インタフェース REST サービスの使用
 - 9 ビジネス・サービス・オブジェクトの使用
 - 10 コンポジット・サービス - BPEL の使用
 - 11 カスタム統合インタフェースの作成および使用
 - 12 Oracle Workflow ビジネス・イベント・システムを使用した Web サービスの起動
 - A 統合リポジトリ・アノテーションの標準
 - B サーバー接続の構成
 - C サンプル・ペイロード
- 用語集

関連情報の入手元

このマニュアルは Oracle E-Business Suite ドキュメント・ライブラリに含まれています。このマニュアル内に Oracle E-Business Suite の他のドキュメントに関する言及がある場合は、最新のリリース 12.2 バージョンのドキュメントのみを参照してください。

オンライン・ドキュメント

すべての Oracle E-Business Suite ドキュメントはオンラインで入手できます(HTML または PDF)。

- **オンライン・ヘルプ** - オンライン・ヘルプのパッチ(HTML)は My Oracle Support で入手できます。
- **Oracle E-Business Suite ドキュメント・ライブラリ** - Oracle E-Business Suite ソフトウェアのディストリビューションに含まれており、各リリース時点における PDF 版のドキュメントを提供します。
- **Oracle E-Business Suite ドキュメント Web ライブラリ** - Oracle Technology Network (http://docs.oracle.com/cd/E26401_01/index.htm)から入手可能で、Oracle E-Business Suite リリース 12.2 ドキュメントの最新更新版を提供します。ほとんどのドキュメントで PDF 版と HTML 版が用意されています。

- **リリース・ノート** - このリリースでの変更点、新機能、既知の問題およびその他の詳細は、対象製品のリリース・ノートで My Oracle Support から入手してご確認ください。
- **Oracle Electronic Technical Reference Manual** - Oracle Electronic Technical Reference Manual (eTRM) には、データベース図と各 Oracle E-Business Suite 製品のデータベース表、フォーム、レポートおよびプログラムに関する詳細が記載されています。これらの情報は、既存のアプリケーションのデータを変換し Oracle E-Business Suite データを非 Oracle アプリケーションに統合して、Oracle E-Business Suite 製品のカスタム・レポートを作成する際に便利です。Oracle eTRM は My Oracle Support から入手できます。

関連ガイド

次の関連ドキュメントがお手元にあることをご確認ください。インストール固有の要件によっては、その他のマニュアルまたはガイドが必要になることがあります。

Oracle Cloud Oracle Integration Cloud での Oracle E-Business Suite アダプタの使用

このマニュアルは、Oracle Integration Cloud (または *Oracle Cloud Oracle E-Business Suite アダプタの使用* では Oracle Integration Cloud Service) での統合のために、Oracle E-Business Suite アダプタを介して、サポートされている Oracle E-Business Suite インタフェースと REST サービスを安全に接続して使用する方法について説明しています。

このマニュアルは、Oracle Cloud Platform as a Service (PaaS) の統合ドキュメントの一部として、Oracle Cloud ドキュメントで入手可能です。

Oracle E-Business Suite の概念

Oracle E-Business Suite リリース 12.2 のデプロイまたは大幅な構成変更を計画しているお客様向けのドキュメントです。Oracle E-Business Suite のアーキテクチャとテクノロジー・スタックについて説明した後に、戦略的なトピックへと話を進めます。特定の目標を達成するのに必要な作業の概要およびインストールと構成の選択肢が記載されています。

Oracle E-Business Suite Desktop Integration Framework 開発者ガイド

Oracle E-Business Suite Desktop Integration Framework は、Oracle Web Applications Desktop Integrator で使用するためのカスタム・インテグレータを定義できる開発ツールです。このマニュアルでは、インテグレータとすべての関連サポート・オブジェクトを定義および管理する方法、およびインテグレータの定義をダウンロードとアップロードする方法について説明しています。

Oracle E-Business Suite 開発者ガイド

このマニュアルには、Oracle E-Business Suite の開発スタッフが従うコーディング標準が含まれます。*Oracle E-Business Suite フォーム・ベース製品のユーザー・インタフェース標準* で説明されているユーザー・インタフェースを実装するために必要な Oracle Application Object Library コンポーネントについて説明しています。このマニュアルには、Oracle E-Business Suite と統合できるようにカスタムの Oracle Forms Developer フォームを構築する際に役立つ情報が含まれます。加えて、このマニュアルには、コンカレント・プログラム、フレックスフィールド、メッセージ、ロギングなどの機能でのカスタマイズに関する情報も含まれます。

Oracle E-Business Suite Integrated SOA Gateway ユーザーズ・ガイド

このマニュアルでは、ハイレベルのサービス有効化プロセスについて説明しており、ユーザーが Oracle Integration Repository 内に存在する統合インタフェース定義やサービスをブラウズおよび表示できる方法について説明しています。

Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド

このマニュアルでは、統合管理者が、統合インタフェース(ネイティブのパッケージ化された統合インタフェース、コンポジット・サービス(BPEL タイプ)およびカスタム統合インタフェースを含む)の Web サービス・アクティビティを管理できる方法について説明しています。また、Oracle Workflow ビジネス・イベント・システムを使用して Oracle E-Business Suite から Web サービスを起動する方法、および Web サービス・セキュリティを管理し、ログを構成し、SOAP メッセージをモニターする方法についても説明しています。

Oracle E-Business Suite メンテナンス・ガイド

このマニュアルでは、Oracle E-Business Suite システムにパッチを適用する方法の説明、adop パッチ・ユーティリティの説明、典型的なパッチ適用操作を実行するうえでのガイドラインおよびヒントを提供しています。また、システムの円滑な実行を保つためのメンテナンス戦略およびツールについても説明しています。

Oracle E-Business Suite モバイル・アプリ管理者ガイド リリース 12.1 および 12.2

このマニュアルでは、Oracle E-Business Suite モバイル・アプリからの接続をサポートするために Oracle E-Business Suite インスタンスを設定する方法について説明しています。また、Oracle E-Business Suite モバイル・アプリを構成するための共通管理タスク、およびサポートされているモバイル・アプリに対してプッシュ通知を有効にするための設定タスクについても説明しています。このマニュアルには、ロギングやトラブルシューティングに関する情報も含まれています。

Oracle E-Business Suite モバイル・アプリ開発者ガイド リリース 12.1 および 12.2

このマニュアルでは、モバイル・アプリケーション・アーカイブ(MAA)を使用してエンタープライズ分散モバイル・アプリを開発する方法および企業ブランディングを実現する方法について説明しています。また、サポートされているモバイル・アプリのプッシュ通知を実装する場合に必要なタスクについても説明しています。加えて、Oracle E-Business Suite REST サービスを実装し、Oracle E-Business Suite Mobile Foundation のログイン・コンポーネントまたはモバイル・アプリ開発フレームワーク(必要に応じて)を使用することでカスタム・モバイル・アプリを開発する方法について説明しています。

Oracle E-Business Suite セキュリティ・ガイド

このマニュアルには、アクセス管理、ユーザー管理、機能セキュリティ、データ・セキュリティ、セキュア構成、監査など、包括的なセキュリティ関連トピックに関する情報が含まれています。また、Oracle E-Business Suite をシングル・サインオン環境に統合する方法についても説明しています。

Oracle E-Business Suite ユーザーズ・ガイド

このマニュアルでは、Oracle E-Business Suite のユーザー・インタフェース(UI)を使用したナビゲート、データの入力と問合せおよびコンカレント要求の実行方法について説明しています。これには、UI の作業環境設定およびカスタマイズに関する情報が含まれています。加えて、このマニュアルでは、Oracle E-Business Suite のアクセシビリティ機能およびキーボード・ショートカットについても説明しています。

Oracle Fusion Middleware Oracle E-Business Suite Adapter ユーザーズ・ガイド

このマニュアルでは、Oracle E-Business Suite と取引先間の統合を開発する際の Oracle E-Business Suite アダプタ(以前の Oracle Fusion Middleware 11g リリースでの Adapter for Oracle Applications)の使用について説明しています。

このマニュアルは、Oracle Fusion Middleware 12c ドキュメント・ライブラリおよび Oracle Fusion Middleware 11g ドキュメント・ライブラリで入手可能です。

Oracle Fusion Middleware Oracle WebLogic Server の紹介

このマニュアルでは、Oracle WebLogic Server の機能の概要を示し、これらを使用してエンタープライズ対応ソリューションを作成する方法について説明しています。このマニュアルは Oracle Fusion Middleware 11g ドキュメント・ライブラリで入手可能です。

Oracle Diagnostics Framework ユーザーズ・ガイド

このマニュアルには、Oracle Diagnostics Framework を使用した Oracle E-Business Suite の実装および診断テストの管理に関する情報が含まれています。

Oracle Workflow 管理者ガイド

このマニュアルでは、ワークフロー対応プロセスを含むすべての製品に必要な設定ステップの完了方法について説明しています。また、Oracle Applications Manager を使用してワークフロー・プロセスおよびビジネス・イベントを管理する方法、実行時ワークフロー・プロセスの進捗のモニター方法およびワークフロー・ユーザーに送信される通知の管理方法についても説明しています。

Oracle Workflow 開発者ガイド

このマニュアルでは、新規ワークフロー・ビジネス・プロセスを定義したり、既存の Oracle E-Business Suite 埋込みワークフロー・プロセスをカスタマイズする方法について説明しています。また、Oracle Mobile Approvals for Oracle E-Business Suite のメッセージ・メタデータを構成する方法、およびビジネス・イベントとイベント・サブスクリプションを定義およびカスタマイズする方法についても説明しています。

Oracle Workflow API リファレンス

このマニュアルでは、開発者および管理者が Oracle Workflow にアクセスするために提供されている API について説明しています。

Oracle XML Gateway ユーザーズ・ガイド

このマニュアルでは、Oracle XML Gateway の機能と、Message Designer、Oracle XML Gateway 設定、実行エンジン、メッセージ・キュー、Oracle Transport Agent などの Oracle XML Gateway アーキテクチャの各コンポーネントについて説明しています。また、すべてのビジネス・トランザクションおよび取引先との間で交換されるメッセージを記録するコラボレーション履歴の使用方法についても説明しています。

このマニュアルでは、Oracle Workflow ビジネス・イベント・システムとの統合、および B2B 取引についても取り上げています。

統合リポジトリ

Oracle Integration Repository は Oracle E-Business Suite アプリケーションで公開されているサービス・エンドポイントに関する情報をまとめたものです。

Oracle E-Business Suite のすべてのビジネス・サービス・インタフェースをまとめたカタログを提供します。このツールを使用すると、システム、アプリケーションまたはビジネス・パートナーとの統合に適したビジネス・サービス・インタフェースを簡単に見つけてデプロイできます。

Oracle Integration Repository は Oracle E-Business Suite の一部として提供されます。インスタンスにパッチを適用すると、このリポジトリのコンテンツは使用環境のインタフェースのリビジョンに合わせて自動的に更新されます。

データベース・ツールを使用して Oracle E-Business Suite データを変更しない

Oracle E-Business Suite のデータを変更する際は、SQL*Plus や Oracle Data Browser、データベース・トリガーなどのツールを使用しないでください。

オラクルでは、Oracle データベースでデータを作成、保存、変更、取得および保守するための強力なツールを各種提供しています。しかし、SQL*Plus などの Oracle ツールを使用して Oracle E-Business Suite データを変更すると、データの整合性が失われ、データ変更の監査ができなくなる恐れがあります。

Oracle E-Business Suite の表は相互に関連しているため、Oracle E-Business Suite フォームを使用して変更を加えると、他の多数の表も同時に更新されます。しかし、Oracle E-Business Suite 以外で Oracle E-Business Suite データを変更した場合、1 つの表で行を変更し、その変更を関連する表に反映しないという状況が発生する可能性があります。表間の同期が失われると誤った情報を取得する可能性があり、Oracle E-Business Suite 全域で想定外の結果が生じる恐れがあります。

Oracle E-Business Suite を使用してデータを変更すると、その変更の有効性が Oracle E-Business Suite によって自動的にチェックされます。また、だれが情報を変更したかの履歴も維持されます。データベース・ツールを使用してデータベース表に情報を入力した場合、不正な情報を格納してしまう可能性があります。また、SQL*Plus などのデータベース・ツールでは変更記録が保持されないため、だれが情報を変更したのかを追跡できなくなります。

Oracle E-Business Suite Integrated SOA Gatewayの概要

Oracle E-Business Suite Integrated SOA Gateway の概要

Oracle E-Business Suite Integrated SOA Gateway (ISG)は、Oracle Fusion Middleware およびサービス指向アーキテクチャ(SOA)テクノロジーに基づいて構築された、Oracle E-Business Suite Web サービスを提供、消費および管理するための、サービス・インフラストラクチャの完全なセットです。

サービス有効化機能により、Oracle Integration Repository に公開されている統合インタフェースを SOAP ベースおよび REST ベースの Web サービスに変換できます。

Oracle E-Business Suite Integrated SOA Gateway では、Oracle WebLogic Server で実行されている Oracle SOA Suite を利用して、様々な統合インタフェースを Oracle E-Business Suite 内に SOAP Web サービスとして公開できるように高性能の機能およびインフラストラクチャを提供しています。SOAP ベースのサービスは WSDL で記述し、Oracle SOA Suite に配置して利用します。

SOAP サービスと異なり、REST サービスは、Oracle SOA Suite に依存することなく、Oracle E-Business Suite のインフラストラクチャで開発します。WADL で記述した REST サービスは Oracle E-Business Suite WebLogic 環境に直接配置します。これらは、Oracle E-Business Suite モバイル・アプリケーションなどユーザードリブンのアプリケーションで使用できます。

Oracle E-Business Suite Integrated SOA Gateway は、他のアプリケーションから提供される Web サービスを起動して消費するためのサービス起動フレームワークとなります。

主要な機能

Oracle E-Business Suite Integrated SOA Gateway では、次の操作を行うことができます。

- Oracle Integration Repository を介して、Oracle E-Business Suite のすべての統合インタフェース定義を表示します

- Oracle Integration Repository からカスタム統合インタフェースをサポートします
- Oracle E-Business Suite 内のシードおよびカスタム統合インタフェースに対するサービス有効化機能(SOAP サービスおよび REST サービス)を提供します
- 統合リポジトリのユーザー・インタフェースを使用して、Oracle E-Business Suite Web サービスの生成と配置などの設計時アクティビティを実行します
- SOAP ベースの Web サービスに対する同期および非同期(確認なしのコールバックのみ)の相互作用パターンをサポートします

注意: このリリースでは、同期サービス・パターンをサポートしている場合、PL/SQL API のみを有効にできます。
- REST ベースの Web サービスに対する同期相互作用パターンをサポートします

注意: このリリースでは、PL/SQL API、コンカレント・プログラム、Java Bean サービス、アプリケーション・モジュール・サービス、オープン・インタフェース表、オープン・インタフェース・ビューのみを REST サービスとして公開できます。
- インバウンド・サービス要求に対して複数の認証タイプをサポートして、Web サービスのコンテンツを保護します
- 承認済ユーザーのみが管理機能を実行できるように、機能セキュリティおよびロール・ベースのアクセス管理セキュリティを実施します
- Oracle E-Business Suite Integrated SOA Gateway のサービス・プロバイダによって生成された Web サービスに対して集中管理型のわかりやすいログイン構成を提供します
- サービス・モニターから Oracle E-Business Suite インバウンド・サービス操作を監査およびモニターします
- Oracle Workflow ビジネス・イベント・システムを利用して、Oracle E-Business Suite から Web サービス起動を有効にします

主要なコンポーネントの機能および定義

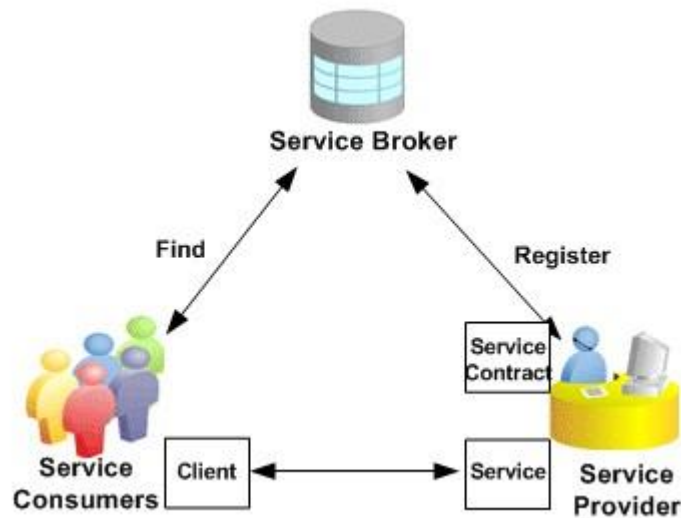
Oracle E-Business Suite Integrated SOA Gateway とそのキー・コンポーネントをさらに深く理解できるように、この項では各コンポーネントのいくつかの主な機能と定義について説明します。

Oracle E-Business Suite Web サービスの有効化

サービス有効化は、Oracle E-Business Suite Integrated SOA Gateway 内の主要な機能です。これは、Oracle Integration Repository に存在している統合インタフェース定義をネイティブにパッケージ化して、Web サービスに変換するためのメカニズムになります。SOAP サービスを統合リポジトリから Oracle SOA Suite に配置することで、Web 経由での消費が増加しま

す。REST サービスは、Oracle E-Business Suite に配置されます。

次の図は、Web サービス・コンポーネントの基本概念を示しています。



- サービス・プロバイダは、Web サービスの基礎となるプライマリ・エンジンです。これは、Oracle E-Business Suite と Oracle SOA Suite との間のブリッジとして機能して、様々なタイプの Oracle E-Business Suite インタフェースのサービス有効化を容易にします。

注意: 以前の Oracle E-Business Suite リリースでは、Oracle E-Business Suite Web サービスの有効化に SOA プロバイダおよび Web サービス・プロバイダを使用していました。リリース 12.2 では、サービス・プロバイダはサービス有効化のエンジンになっています。

サービス・プロバイダは、Oracle SOA Suite を利用して、Oracle E-Business Suite SOAP ベースのサービスをプロビジョニングします。これは、バックグラウンドで実際にサービスの生成と配置を行うエンジンです。

- サービス・コンシューマ(Web サービス・クライアント)は、サービス・プロバイダから提供されるサービスを使用または消費するパーティです。
- サービス・ブローカー(サービス・レジストリ)には、潜在的なサービス・コンシューマがサービス情報を使用できるようにサービスの場所と契約を記述します。

Oracle Integration Repository およびサービス有効化

Oracle Integration Repository は、Oracle E-Business Suite の不可欠な部分であり、Oracle E-Business Suite 内のアプリケーションで公開されている数多くのインタフェース・エンドポイントを含む、一元化されたリポジトリです。これは、次のインタフェース・タイプをサポートしています。

- PL/SQL
- XML ゲートウェイ
- コンカレント・プログラム
- ビジネス・イベント
- インタフェース表/ビュー
- EDI
- ビジネス・サービス・オブジェクト(サービス Bean)
- Java

通常の Java API とは別に、Java インタフェースには次のサブカテゴリがあります。

- アプリケーション・モジュール・サービス

注意: アプリケーション・モジュール実装クラスは、OA フレームワーク・ベースのコンポーネントとページを制御するビジネス・ロジックへのアクセスを提供する Java クラスです。このような Java クラスは、アプリケーション・モジュール・サービスと呼ばれ、Java インタフェースのサブタイプに分類されています。

- Java Bean サービス

注意: Java API のうちメソッドが単純データ型またはシリアライズ可能な Java Bean のパラメータを使用しているものは、Java Bean サービスに分類されています。このような Java API は、REST ベースの Web サービスとして公開できます。

- セキュリティ・サービス

注意: セキュリティ・サービスは、Oracle Application Object Library からの一連の事前定義済および事前配置済の REST サービスです。このようなサービスには、モバイル・アプリケーション向けの認証サービスや認可サービスなどがあります。これらのサービスは Java に基づいて構築されているため、Java インタフェースのサブタイプに分類されています。

フォーム用 Java API は、サービス可能インタフェースではなく、SOAP サービスとして公開できないことに注意してください。

既存の「フォーム用 Java API」インタフェースの推奨される代替方法は、My Oracle Support のナレッジ・ドキュメント 966982.1 を参照してください。

- コンポジット・インタフェース

Oracle E-Business Suite Integrated SOA Gateway では、Oracle Integration Repository を利用して、Web サービスを生成して配置でき、さらにサービス開発ライフサイクルを管理できます。

注意: 統合リポジトリに存在しているすべてのインタフェース・タイプをサービス対応にできるわけではないことに注意してください。サービス有効化をサポートしているインタフェース・タイプは、XML ゲートウェイ、PL/SQL、コンカレント・プログラム、ビジネス・イベント、ビジネス・サービス・オブジェクトおよび Java Bean サービスです。

前述のとおり、セキュリティ・サービスは、Oracle Application Object Library からの事前配置済 REST サービスです。他のサポートされているインタフェース・タイプで要求されているとおりに統合リポジトリからセキュリティ・サービスを有効化する必要はありません。

サービス起動フレームワーク

サービス起動フレームワーク(SIF)は、Oracle Workflow Java ビジネス・イベント・システム (JBES)およびシード Java ルール関数を利用して、Oracle E-Business Suite 内のサービスを起動します。

これは、開発者が WSDL 記述を介して Web サービスと対話するためのインフラストラクチャになります。

サービス起動フレームワークを使用する方法の詳細は、「サービス起動フレームワークを使用した Web サービス起動」(ページ 12-2)を参照してください。

サービス・モニター

SOA モニターと呼ばれるサービス・モニターは、サービスの実行をモニターおよび管理する軽量なツールです。

サービス・モニターは、Web サービスの要求と応答のインスタンスごとにデータおよび統計をフェッチして、Oracle E-Business Suite Web サービスをモニターできます。Oracle E-Business Suite でサービス・モニターのユーザー・インタフェースを使用すると、実行時の要求メッセージおよび応答メッセージを表示できます。

サービス・モニターを介してモニターおよび監査されるのは SOAP サービスのみであることに注意してください。実行時の REST サービスのモニタリングと監査の機能は、このリリースではサポートされていません。

Web サービス・セキュリティ

Oracle E-Business Suite Integrated SOA Gateway は、サブジェクト認証および認可を介してセキュリティ・ルールを実施します。

- Oracle E-Business Suite Web サービスを要求するユーザーを認証するには、選択した認証タイプに基づいて要求メッセージをチェックする必要があります。
 - SOAP メッセージは、UsernameToken または SAML トークン・ベースのセキュリティを使用して認証する必要があります。識別された認証情報は、wsse:security Web セキュリティ・ヘッダーに埋め込まれます。
 - REST メッセージは、HTTP または HTTPS トランスポート・レベルで HTTP Basic 認証またはトークン・ベースの認証を使用して認証されます。
- 特定のサービスまたは操作に対してユーザーを認証するには、セキュリティ付与を介してアクセス権限をユーザーに明示的に付与する必要があります。複数の組織に関連するインタフェース実行を認証できるように、複数組織アクセス管理(MOAC)セキュリティ・ルールも実装されています。

また、入力メッセージ・ヘッダー(SOAP サービス用の SOAHeader や REST サービス用の RESTHeader など)は、サブジェクト認証の一環として Oracle E-Business Suite サービスの起動に必要なアプリケーション・コンテキストを渡すために使用します。

統合インタフェースおよびサービスの検出 および表示

概要

「統合開発者」ロールを持つユーザーは、Oracle Integration Repository からの統合インタフェースのブラウズおよび表示に加え、選択したインタフェース向けの SOAP Web サービスを生成できます。

これらの設計時タスク、および WSDL 要素と WADL 要素の構造についての理解を深めるために、この章には次のトピックが含まれています。

- 統合インタフェースの検索および表示、ページ 2-1
- インタフェース詳細のレビュー、ページ 2-5
- SOAP Web サービスの生成、ページ 2-6
- WSDL 要素詳細のレビュー、ページ 2-8
- WADL 要素詳細のレビュー、ページ 2-18
- SOAP メッセージについて、ページ 2-25
- REST メッセージについて、ページ 2-44

統合インタフェースの検索および表示

統合インタフェースのブラウズ

Oracle Integration Repository に存在している統合インタフェースを見つけて表示するには、様々な方法があります。インタフェースをブラウズするには、「表示別」ドロップダウン・リストから次のいずれかの項目を選択します。

- 製品ファミリ
- インタフェース・タイプ
- 標準

選択した後、これらのビューのいずれかでツリー・ノードを展開して、使用可能なインタフェースのリストを表示します。

インタフェースをブラウズする方法の詳細は、*Oracle E-Business Suite Integrated SOA Gateway ユーザーズ・ガイド*の「統合インタフェースのブラウズ」を参照してください。

統合インタフェースの検索

統合インタフェースを検索するには、「**検索**」をクリックしてメインの「検索」ページにアクセスします。「検索」ページの「**詳細検索オプションの表示**」リンクをクリックして、詳細検索オプションを表示します。たとえば、「範囲」フィールドの場合、ドロップダウンから「**アプリケーション限定で非公開**」、「**Oracle 内部**」、「**公開**」および「**すべて**」の値を選択できます。「範囲」フィールドから「すべて」を選択した場合、「公開」、「アプリケーション限定で非公開」、「Oracle 内部」の各インタフェースを含め、すべての統合インタフェースが「結果」リージョンにリストされます。

注意: 統合アナリストである場合は、「範囲」ドロップダウン・リストに「すべて」(デフォルト)と「公開」の値リストのみが表示され、「範囲」フィールドでデフォルト値「すべて」を選択しても、検索結果として「公開」統合インタフェースのみが取得されます。

「公開」、「アプリケーション限定で非公開」および「Oracle 内部」の詳細は、*Oracle E-Business Suite Integrated SOA Gateway ユーザーズ・ガイド*の「範囲」を参照してください。

配置済 Web サービスの検索

たとえば、コンカレント・プログラム向けに配置されたサービスを見つけるには、「インタフェース」フィールドで「コンカレント・プログラム」を選択し、「**詳細検索オプションの表示**」リンクをクリックすると表示される「Web サービス・ステータス」フィールドで「配置済」を選択します。検索を実行すると、コンカレント・プログラム・インタフェース・タイプのすべての配置済 Web サービスが表示されます。

Java Bean サービス、アプリケーション・モジュール・サービスおよびセキュリティ・サービスの検索

Java Bean サービス、アプリケーション・モジュール・サービスおよびセキュリティ・サービスは、すべて特殊化された Java クラスで、Java インタフェースのサブタイプに分類されており、「統合リポジトリ」の「Java」インタフェース・タイプに表示されます。

アプリケーション・モジュール・サービスの検索

Integration Repository

Search Browse

Interface Name Internal Name

Product Family All Interface Type All

Product All Business Entity

[Hide More Search Options](#)

TIP Select Category before a Category Value

Category Interface Subtype Web Service Type All

Category Value Application Module Services Web Service Status All

Interface Source All Standard All

Scope All Standard Specification

Status All

Export

Name	Internal Name	Product	Type	Source	Status	Description
WF Worklist Service	oracle.apps.fnd.wf.worklist.service.rt.server.WFWorklistServiceAMImpl	Application Object Library	Java	Custom	Active	this is a sample WF AM class

「検索」ページでこれらのインタフェースまたはサービスを簡単に見つけるには、「**詳細検索オプションの表示**」リンクをクリックして詳細検索オプションを表示します。検索基準として、次のキー検索値を必要に応じて製品ファミリまたは範囲とともに入力します。

- カテゴリ: インタフェース・サブタイプ
- カテゴリ値: フォーム用 Java API、Java Bean サービス、アプリケーション・モジュール・サービスまたはセキュリティ・サービス

注意: 「フォーム用 Java API」インタフェースを検索して見つけることができますが、これらはサービス可能インタフェースではなく、SOAP サービスとして公開できません。このタイプのインタフェースを Web サービスとして使用することを計画している場合は、代替のサービス可能インターフェースとして、Web サービスとして配置できる PL/SQL やビジネス・サービス・オブジェクトなどのインタフェースを使用することをお勧めします。既存の「フォーム用 Java API」インタフェースの推奨される代替方法は、My Oracle Support のナレッジ・ドキュメント 966982.1 を参照してください。

インタフェースまたはサービス詳細を表示するには、検索結果リージョンから表示するインタフェースまたはサービス名のリンクをクリックします。インタフェース詳細ページが表示されます。インタフェース詳細については、「インタフェース詳細のレビュー」(ページ 2-5)を参照してください。

カスタム統合インタフェースの検索

注釈付けされたカスタム・インタフェース定義は、統合リポジトリに正常にアップロードされると、これらが属するインタフェース・タイプにマージされ、「統合リポジトリ」ブラウザ・ウィンドウからの Oracle シード・インタフェースとともに表示されます。「インタフェース・ソース」の「カスタム」はそのようなカスタム統合インタフェースを分類するために使用し、一方「インタフェース・ソース」の「Oracle」は Oracle E-Business Suite の Oracle シード・インタフェースに使用します。

カスタム統合インタフェースを見つけるには、まず「**詳細検索オプションの表示**」リンクをクリックして、詳細検索オプションを表示します。

カスタム統合インタフェースの検索

The screenshot shows the 'Integration Repository' search page. The 'Interface Source' dropdown is open, with 'Custom' selected. The search results table is as follows:

Name	Internal Name	Product	Type	Source	Status	Description
ISG_Overload Package	ISG_OVERLOAD_PKG	Application Object Library	PL/SQL	Custom	Active	This is a sample Overload plsql Package to calculate area of different shapes
WF_Worklist Service	oracle.apps.fnd.wf.worklist.service.rt.server.WFWorklistServiceAMImpl	Application Object Library	Java	Custom	Active	this is a sample WF AM class

次に、「インタフェース・ソース」ドロップダウン・リストから「カスタム」を選択し、さらに検索基準として必要に応じてインタフェース・タイプ、製品ファミリまたは範囲も選択します。

「検索」ページの各検索フィールドの詳細は、*Oracle E-Business Suite Integrated SOA Gateway ユーザーズ・ガイド*の「統合インタフェースの検索」を参照してください。

すべての統合インタフェース・タイプを検索するには:

1. 「統合開発者」ロールを持つユーザーとして、Oracle E-Business Suite にログインします。
ナビゲーション・メニューから、「統合 SOA ゲートウェイ」職責を選択します。「統合リポジトリ」リンクを選択して、リポジトリ・ブラウザを開きます。
2. 「検索」をクリックして、メインの「検索」ページを開きます。
3. 製品ファミリ、製品、インタフェース・タイプ、ビジネス・エンティティなど適切な検索情報を入力します。
4. 「**詳細検索オプションの表示**」リンクをクリックして、検索オプションをさらに表示します。
 - カスタム統合インタフェースを検索するには、「インタフェース・ソース」フィールドで「カスタム」を選択します。

- Java Bean サービス、アプリケーション・モジュール・サービスまたはセキュリティ・サービスを検索するには、「カテゴリ」フィールドで「インタフェース・サブタイプ」を選択し、「カテゴリ値」フィールドで「Java Bean サービス」、「アプリケーション・モジュール・サービス」または「セキュリティ・サービス」を選択します。
5. 配置済統合インタフェースを表示するには、「Web サービス・ステータス」フィールド・ドロップダウン・リストから「配置済」を選択します。
 6. すべての統合インタフェースを表示するには、「範囲」フィールドから「すべて」を選択します。「公開」、「Oracle 内部」および「アプリケーション限定で非公開」を含め、すべての統合インタフェースが「結果」リージョンに表示されます。
 7. 様々な範囲の統合インタフェースを表示するには、「範囲」ドロップダウン・リストから「公開」、「Oracle 内部」または「アプリケーション限定で非公開」をそれぞれ選択します。
 8. 「進む」をクリックして、検索を実行します。検索基準に一致するすべてのインタフェースが表示されます。
 9. 検索結果からインタフェースを選択して、インタフェース詳細を表示します。

インタフェース詳細のレビュー

選択したインタフェースの詳細を表示するには、検索の実行後、表示するインタフェース名のリンクをクリックします。インタフェース詳細ページが表示され、インタフェースの一般的な情報、詳細な説明、ソース情報およびインタフェース・メソッド(またはプロシージャおよび関数)のリージョンを参照できます。

また、Web サービス情報も、サービス可能なインタフェースに対してのみ表示できます。

選択したインタフェースに基づき、SOAP ベースや REST ベースのサービスなど関連付けられている Web サービス情報がインタフェース詳細ページにある場合は、それらを参照できます。

注意: このリリースでは、PL/SQL API とコンカレント・プログラムのみを、SOAP サービスと REST サービスの両方として公開できます。Java Bean サービス、アプリケーション・モジュール・サービス、オープン・インタフェース表、オープン・インタフェース・ビューは、REST サービスとしてのみ公開できます。

SOAP ベースのサービスの詳細は、*Oracle E-Business Suite Integrated SOA Gateway ユーザーズ・ガイド*の「SOAP Web サービスの共通情報」を参照してください。

REST ベースのサービスの詳細は、*Oracle E-Business Suite Integrated SOA Gateway ユーザーズ・ガイド*の「REST Web サービスの共通情報」を参照してください。

SOAP または REST Web サービスが配置されたら、それらを実行時に起動できます。これらのサービスを起動する方法の詳細は、このマニュアルで後述する個々の章を参照してください。

SOAP Web サービスの生成

「統合開発者」ロールを持つユーザーは、インタフェース定義を、WSDL 記述で表現される SOAP Web サービスに変換できます。

相互作用パターン表からの必要な相互作用パターンの選択

SOAP サービスは、ニーズに合わせて同期または非同期の相互作用パターン、または同期と非同期の両方のパターンをサポートして生成できます。

サービスを生成する前に、統合管理者または統合開発者は選択したインタフェースまたはそのインタフェースに含まれている特定のメソッド用に相互作用パターン表で少なくとも 1 つの相互作用パターンを選択する必要があります。これは、1 つ以上のメソッドの場合はメソッド・レベルで、またはすべてのメソッドの場合はインタフェース・レベルで行うことができます。

重要: このリリースでは、非同期操作は SOAP ベースのサービスを有効にする際に PL/SQL インタフェースでのみサポートされます。

- 「XML ゲートウェイ」および「コンカレント・プログラム」インタフェース・タイプの場合

各インターフェイスにはメソッドが 1 つだけ含まれており、デフォルトでは同期的にのみサービスを有効にできます。このため、「XML ゲートウェイ」インタフェースの「Web サービス」リージョンにも、「コンカレント・プログラム」インタフェースの「SOAP Web サービス」タブにも、相互作用パターン表が表示されません。

- 「ビジネス・サービス・オブジェクト」インタフェース・タイプの場合

各インタフェースに複数のメソッドが含まれることがあるため、メソッドを選択できるように相互作用パターン表に「同期」列のみが表示されます。

デフォルトではいずれの相互作用パターンも選択されていないことに注意してください。ただし、システムを以前のリリースからアップグレードした場合は、下位互換性のために、サービスに含まれるすべてのメソッドに対して「同期」パターンが選択されます。

サービスの生成

「Web サービス」リージョン(または選択したインタフェースを SOAP サービスと REST サービスの両方として公開できる場合は「SOAP Web サービス」タブ)で、インタフェースの相互作用パターンを選択した後、開発者は「生成」をクリックし、その選択した相互作用パターンを使用して WSDL で表現される Web サービスを生成できます。

サービス生成後

サービスが正常に生成されると、選択された相互作用パターンが表に表示されます。この表は引き続き編集可能ですが、相互作用パターンに加えた変更は、サービスを再生成した後にも適用されます。

インタフェース名のノードを展開し、そのインタフェースに含まれるすべてのメソッドを表示できます。

ビジネス・サービス・オブジェクト・インタフェースの「Web サービス」リージョン

Web Service

Expand All Collapse All		
Display Name	Internal Name	Synchronous
Account Merge Service	/oracle/apps/ar/hz/service/account/AccountMergeService	☒
create Account Merge Request	createAccountMergeRequest	☒
get Account Merge Details	getAccountMergeDetails	☒

 **TIP** To apply any changes in Interaction Pattern, Generate or Regenerate the service.

Web Service Status **Generated** | [View WSDL](#)
Interaction Pattern **Synchronous**

* **Authentication Type**

- ☐ Username Token
☐ SAML Token (Sender Vouches)

サービスが正常に生成された後、SOAP サービス・ステータスが「未生成」から「生成済」に変更され、選択されたインタフェースに使用可能な WSDL 記述があるものの、まだ配置されていないことが示されます。

重要: サービス生成がまだ進行中である場合は、SOAP サービス・ステータスとして「生成中」が表示されます。

「**WSDL の表示**」リンクをクリックすると、生成された WSDL コードが表示されます。WSDL の詳細は、「WSDL 要素詳細のレビュー」(ページ 2-8)を参照してください。

Web サービスの再生成

インタフェース定義が変更された場合や、選択した相互作用パターン情報が変更された場合は、「**再生成**」をクリックして Web サービスを再生成できます。再生成すると、サービス定義と相互作用パターン情報が、インタフェースで行った変更を反映するように変更されます。新しいサービス定義に基づいて該当する Web サービス・クライアントを変更する必要があります。

インタフェース定義が変更されていない場合は、サービスを再生成してもサービス定義は変更されません。引き続き新しいサービス定義で既存の Web サービス・クライアントを使用できます。

サービス生成情報の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「SOAP Web サービスの生成」を参照してください。

WSDL 要素詳細のレビュー

インタフェースが Web サービスとして公開されている場合、インタフェース詳細ページに対応する WSDL ファイルが表示されます。

「WSDL の表示」リンクをクリックすると、新しいウィンドウが起動して関連する WSDL 文書が表示されます。この XML ベースの文書では、選択した Web サービスが記述され、サービスの使用方法が示されます。

たとえば、インタフェース詳細ページから PL/SQL 請求書作成用の配置済の「WSDL の表示」リンクをクリックすると、WSDL 文書が表示されます。

```
infra/services/default/PLSQL_AR_INVOICE_API_PUB/AR_INVOICE_API_PUB_Service/?
XSD=xsd/APPS_ISG_CREATE_SINGLE_INVOICE_AR_INVOICE_API_PUB-24CREATE_SIN.xsd" />
</schema>
- <schema xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  targetNamespace="http://xmlns.oracle.com/apps/ar/soapprovider/plsql/ar_invoice_api_pub/">
- <element name="SOAHeader">
- <complexType>
- <sequence>
  <element name="Responsibility" minOccurs="0" type="string" />
  <element name="RespApplication" minOccurs="0" type="string" />
  <element name="SecurityGroup" minOccurs="0" type="string" />
  <element name="NLSLanguage" minOccurs="0" type="string" />
  <element name="Org_Id" minOccurs="0" type="string" />
</sequence>
</complexType>
</element>
</schema>
</types>
- <message name="CREATE_INVOICE_Input_Msg">
  <part name="header" element="tns:SOAHeader" />
  <part name="body" element="tns1:InputParameters" />
</message>
- <message name="CREATE_INVOICE_Output_Msg">
  <part name="body" element="tns1:OutputParameters" />
</message>
- <message name="CREATE_SINGLE_INVOICE_Input_Msg">
  <part name="header" element="tns:SOAHeader" />
  <part name="body" element="tns2:InputParameters" />
</message>
- <message name="CREATE_SINGLE_INVOICE_Output_Msg">
  <part name="body" element="tns2:OutputParameters" />
</message>
- <portType name="AR_INVOICE_API_PUB_PortType">
- <operation name="CREATE_INVOICE">
  <input message="tns:CREATE_INVOICE_Input_Msg" />
  <output message="tns:CREATE_INVOICE_Output_Msg" />
</operation>
- <operation name="CREATE_SINGLE_INVOICE">
  <input message="tns:CREATE_SINGLE_INVOICE_Input_Msg" />
```

注意: 新しいウィンドウの `http://<hostname>:<port>/soa-infra/services/default/<sid>_PLSQL_AR_INVOICE_API_PUB/AR_INVOICE_API_PUB_Service?wsdl` アドレスは、インタフェース詳細ページに表示された正確な WSDL URL 情報です。このアドレスをコピーして、パートナー・リンクの作成中の BPEL プロセスなど、任意の Web サービス・クライアントで直接使用できます。

WSDL 文書の構造

WSDL 文書は単なる一連の定義です。ルートに **definitions** 要素があり、その中に定義があります。*definitions* 要素は、Web サービスで提供する一連のサービスを定義します。

definitions 要素には、多くの場合、WSDL 文書によるそれ自体への参照を可能にする XML スキーマの規則である、オプションの `TargetNamespace` プロパティを含めます。

たとえば、請求書作成 API (AR_INVOICE_API_PUB)の WSDL 文書の *definitions* 要素は次のようになります。

```
<definitions name="AR_INVOICE_API_PUB" targetNamespace="http://xmlns.oracle.com/apps/ar/soapprovider/plsql/ar_invoice_api_pub/">
```

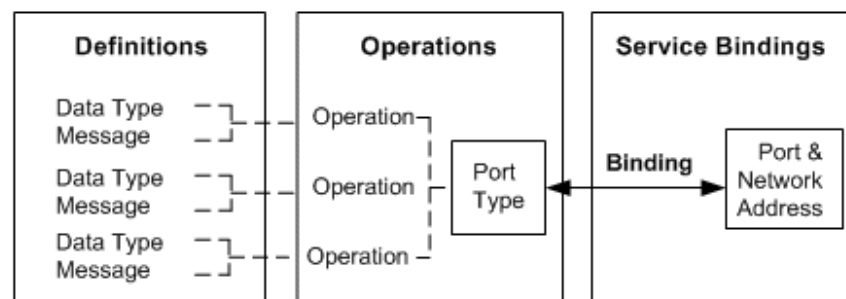
この要素は多数のネームスペースを指定し、これらが文書の残り全体で使用されます。

definitions 要素に加えて、Web サービスは次の 6 つの主要な要素を使用して定義されます。

- **Types:** 交換されるメッセージの記述に使用されるデータ型定義を指定します。
- **Message:** 送信されるデータの抽象定義を表します。
- **PortType:** 一連の抽象操作です。各操作は、入力メッセージと出力メッセージを参照します。
- **Binding:** 特定の portType によって定義されている操作およびメッセージの具体的なプロトコルとデータ形式の仕様を指定します。
- **Port:** バインディングのアドレスを指定し、これにより単一の通信エンドポイントを定義します。
- **Service:** 一連の関連するポートを集約する場合に使用されます。

次の図は、WSDL の基本部分の関係を示しています。

WSDL Basic Parts Relationship Diagram



Types

types 要素には、WSDL で記述されているすべてのメソッド・コールで使用するすべてのデータ型が含まれます。これを使用して、WSDL 部分の構造を記述するために使用される XML スキーマ(xsd:schema)を指定できます。

この Types 要素の構造は、次のようになります。

```

<definitions...>
  <types>
    <xsd:schema.../>*
  </types>
</definitions>

```

たとえば、請求書作成 Web サービスには、次の 2 つの関数が含まれています。

- CREATE_INVOICE
- CREATE_SINGLE_INVOICE

各関数は、データ型定義に記述されています。WSDL では、メッセージ・スキーマでタイプを定義するためにシステム・メカニズムのタイプとして XSD の使用が優先されます。結果として、CREATE_INVOICE 関数のメッセージ・スキーマの場所は、同期操作パターンの場合には APPS_ISG_XX_AR_INVOICE_API_PUB-24CREATE_INV.xsd に定義され、非同期パターンの場合には CREATE_INVOICE_ASYNC.xsd に定義されます。

CREATE_SINGLE_INVOICE 関数のメッセージ・スキーマの場所は、同期操作パターンの場合には APPS_ISG_XX_AR_INVOICE_API_PUB-24CREATE_SIN.xsd に定義され、非同期パターンの場合には CREATE_SINGLE_INVOICE_ASYNC.xsd に定義されます。

注意: 生成されたがまだ配置されていないサービスの場合、抽象 WSDL 記述にはスキーマの場所の一時的な場所が表示されます(<include schemaLocation="http://<hostname>:<port>/isgctx/isgapp/plsql/ar_invoice_api_pub/APPS_ISG_XX_AR_INVOICE_API_PUB-24CREATE_INV.xsd" />など)。

配置済のサービスの場合は、かわりに、soa-infra にサービスがホストされているサービス・エンドポイントの物理的場所が次のように表示されます。

```

<types>
  <schema elementFormDefault="qualified"
    targetNamespace="http://xmlns.oracle.
com/apps/ar/soapprovider/plsql/ar_invoice_api_pub/create_invoice/">
    <include schemaLocation="http://<soa_suite_hostname>:<port>/soa-
infra/services/default/<jndi_name> PLSQL_AR_INVOICE_API_PUB/AR_INVOI
CE_A_PI_PUB_Service?XSD=xsd/APPS_ISG_XX_AR_INVOICE_API_PUB-
24CREATE_INV.xsd" />
  </schema>
  <schema elementFormDefault="qualified"
    targetNamespace="http://xmlns.oracle.
com/apps/ar/soapprovider/plsql/ar_invoice_api_pub/create_single_invoi
ce/" >
    <include schemaLocation="http://<soa_suite_hostname>:<port>/soa-
infra/services/default/<jndi_name> PLSQL_AR_INVOICE_API_PUB/AR_INV
OICE_A_PI_PUB_Service?XSD=xsd/APPS_ISG_XX_AR_INVOICE_API_PUB-
24CREATE_SIN.xsd" />
  </schema>
  <schema elementFormDefault="qualified"
    targetNamespace="http://xmlns.oracle.
com/apps/ar/soapprovider/plsql/ar_invoice_api_pub/">
    <element name="SOAHeader">
      ...
    </element>
  </schema>
  <schema elementFormDefault="qualified"
    targetNamespace="http://xmlns.oracle.
com/apps/ar/soapprovider/plsql/ar_invoice_api_pub/create_invoice/">
    <include schemaLocation="http://<soa_suite_hostname>:<port>/soa-
infra/services/default/<jndi_name> PLSQL_AR_INVOICE_API_PUB/AR_INVOI
CE_A_PI_PUB_Service?XSD=xsd/APPS_ISG_XX_AR_INVOICE_API_PUB-
24CREATE_INV.xsd" />
  </schema>
  <schema elementFormDefault="qualified"
    targetNamespace="http://xmlns.oracle.
com/apps/ar/soapprovider/plsql/ar_invoice_api_pub/create_single_invoi
ce/" >
    <include schemaLocation="http://<soa_suite_hostname>:<port>/soa-
infra/services/default/<jndi_name> PLSQL_AR_INVOICE_API_PUB/AR_INV
OICE_A_PI_PUB_Service?XSD=xsd/APPS_ISG_XX_AR_INVOICE_API_PUB-
24CREATE_SIN.xsd" />
  </schema>
  <schema elementFormDefault="qualified"
    targetNamespace="http://xmlns.oracle.
com/apps/ar/soapprovider/plsql/ar_invoice_api_pub/create_invoice_async"
    >
    <include schemaLocation="http://<soa_suite_hostname>:<port>/soa-
infra/services/default/<jndi_name> PLSQL_AR_INVOICE_API_PUB/AR_INVOI
CE_A_PI_PUB_Service?XSD=xsd/CREATE_INVOICE_ASYNC.xsd" />
  </schema>
  <schema elementFormDefault="qualified"
    targetNamespace="http://xmlns.oracle.
com/apps/ar/soapprovider/plsql/ar_invoice_api_pub/create_single_inv
oice_a_synch/">
    <include schemaLocation="http://<soa_suite_hostname>:<port>/soa-
infra/services/default/<jndi_name> PLSQL_AR_INVOICE_API_PUB/AR_INV
OICE_A_PI_PUB_Service?XSD=xsd/CREATE_SINGLE_INVOICE_ASYNC.xsd"/>
  </schema>
</types>

```

Web メッセージの定義に役立つメッセージ・スキーマの場所およびスキーマ要素に加えて、**Types** 要素は、複合データ型を入力として取ることができます。

たとえば、次のように **SOAHeader** の下にリストされている職責、職責アプリケーション、セキュリティ・グループ、NLS 言語、組織 ID の各複合型は、サービス呼出し中にアプリケーション・コンテキストを設定するための値を渡す際に使用されます。

```

...
<schema elementFormDefault="qualified"
  targetNamespace="http://xmlns.oracle.
com/apps/ar/soapprovider/plsql/ar_invoice_api_pub/">
  <element name="SOAHeader">
    <complexType>
      <sequence>
        <element name="Responsibility" minOccurs="0" type="string"/>
        <element name="RespApplication" minOccurs="0" type="string"/>
        <element name="SecurityGroup" minOccurs="0" type="string"/>
        <element name="NLSLanguage" minOccurs="0" type="string"/>
        <element name="Org_Id" minOccurs="0" type="string" />
      </sequence>
    </complexType>
  </element>
</schema>
...

```

Message

Message 要素は、メッセージの名前を定義します。これは、*Element* 属性または *Type* 属性を使用してメッセージのコンテンツを記述する、1 つ以上の *Part* 要素で構成されています。

パートは、メッセージの論理的な抽象コンテンツを記述します。バインディングは、パートに関するバインディング固有の情報を指定するために、パートの名前を参照できます。

この要素の構造は、次のようになります。

```

<definitions...>
  <message name="nmtoken"> *
    <part name="nmtoken" element="qname"? type="qname"? />
  </message>
</definitions>

```

一般的なドキュメント形式の Web サービスの場合、入力メッセージと出力メッセージにはヘッダーと本文のパートがあります。たとえば、請求書作成 Web サービスの *Message* 要素は次のようになります。

```

<message name="CREATE_INVOICE_Input_Msg">
  <part name="header" element="tns:SOAHeader"/>
  <part name="body" element="tns1:InputParameters"/>
</message>
<message name="CREATE_INVOICE_Output_Msg">
  <part name="body" element="tns1:OutputParameters"/>
</message>
<message name="CREATE_SINGLE_INVOICE_Input_Msg">
  <part name="header" element="tns:SOAHeader"/>
  <part name="body" element="tns2:InputParameters"/>
</message>
<message name="CREATE_SINGLE_INVOICE_Output_Msg">
  <part name="body" element="tns2:InputParameters"/>
</message>
<message name="CREATE_INVOICE_ASYNCH_Input_Msg">
  <part name="header" element="tns:SOAHeader"/>
  <part name="body" element="tns1:InputParameters"/>
</message>
<message name="CREATE_SINGLE_INVOICE_ASYNCH_Input_Msg">
  <part name="header" element="tns:SOAHeader"/>
  <part name="body" element="tns2:InputParameters"/>
</message>
<message name="CREATE_INVOICE_ASYNCH_Output_Msg">
  <part name="body" element="tns1:OutputParameters"/>
</message>
<message name="CREATE_SINGLE_INVOICE_ASYNCH_Output_Msg">
  <part name="body" element="tns2:InputParameters"/>
</message>

```

関連付けられたスキーマによって定義されている各メッセージには、入力メッセージ・パートと出力メッセージ・パートがあります。たとえば、請求書作成 Web サービスには、次の 2 つの関数があります。

- **CREATE_INVOICE**

この関数の入力メッセージは、同期操作パターンの場合には

CREATE_INVOICE_Input_Msg によって定義され、非同期パターン場合には **CREATE_INVOICE_ASYNCH_Input_Msg** によって定義されます。

この関数の結果が返される出力メッセージは、同期操作パターンの場合には

CREATE_INVOICE_Output_Msg によって定義され、非同期パターンの場合には **CREATE_INVOICE_ASYNCH_Output_Msg** によって定義されます。

入力メッセージおよび出力メッセージのスキーマは、同期パターンの場合には

APPS_ISG_XX_AR_INVOICE_API_PUB-24CREATE_INV.xsd に定義され、非同期パターンの場合には CREATE_INVOICE_ASYNCH.xsd に定義されます。

- **CREATE_SINGLE_INVOICE**

この関数の入力メッセージは、同期操作パターンの場合には

CREATE_SINGLE_INVOICE_Input_Msg によって定義され、非同期パターン場合には **CREATE_SINGLE_INVOICE_ASYNCH_Input_Msg** によって定義されます。

この関数の結果が返される出力メッセージは、同期操作パターンの場合には

CREATE_SINGLE_INVOICE_Output_Msg によって定義され、非同期パターン場合には **CREATE_SINGLE_INVOICE_ASYNCH_Output_Msg** によって定義されます。

入力メッセージおよび出力メッセージのスキーマは、同期パターンの場合には

APPS_ISG_XX_AR_INVOICE_API_PUB-24CREATE_SIN.xsd に定義され、非同

期パターンの場合には CREATE_SINGLE_INVOICE_ASYNC.xsd に定義されます。

各メッセージの本文パートの値は SOAP 本文として設定され、ヘッダー・パートの値は Web サービス認証に必要な SOAP ヘッダーに設定されます。

詳細は、「Web サービス入力メッセージ・パートについて」(ページ 12-16)を参照してください。

PortType

portType 要素は、複数のメッセージ要素を結合し、Web サービスでサポートされている完全な一方方向の操作または往復の操作を形成します。

たとえば、*portType* 要素では、1 つの要求(入力メッセージ要素)メッセージと 1 つの応答(出力メッセージ要素)メッセージを、SOAP サービスで最もよく使用される操作である同期要求応答操作に結合できます。

一方方向操作の場合、操作には入力要素のみが含まれます。この要素の構造は、次のようになります。

```
<wsdl:definitions...>
  <wsdl:portType name="nmtoken">*
    <operation name="nmtoken"/>
    <wsdl:input name="nmtoken"? message="qname">?
  </wsdl:input>
  <wsdl:output name="nmtoken"? message="qname">?
  </wsdl:output>
  <wsdl:fault name="nmtoken"? message="qname">?
  </wsdl:fault>
  </wsdl:operation>
</wsdl:portType>
</wsdl:definitions>
```

注意: オプションの Fault 要素は、要求応答操作モデルと請求応答操作モデルの両方のエラー処理に使用できます。この機能は、このリリースではサポートされていません。

この請求書作成 Web サービスの例では、前述の 2 つの関数に対応して、AR_INVOICE_API_PUB_PortType に次の 2 つの操作があります。

- CREATE_INVOICE
 - 入力: CREATE_INVOICE_Input_Msg
 - 出力: CREATE_INVOICE_Output_Msg
- CREATE_SINGLE_INVOICE
 - 入力: CREATE_SINGLE_INVOICE_Input_Msg
 - 出力: CREATE_SINGLE_INVOICE_Output_Msg

同期要求応答操作パターンの場合、入力メッセージおよび出力メッセージが次のように表示されます。

```
<portType name="AR_INVOICE_API_PUB_PortType">
  <operation name="CREATE_INVOICE">
    <input name="tns:CREATE_INVOICE_Input_Msg" />
    <output name="tns:CREATE_INVOICE_Output_Msg" />
  </operation>
  <operation name="CREATE_SINGLE_INVOICE">
    <input name="tns:CREATE_SINGLE_INVOICE_Input_Msg" />
    <output name="tns:CREATE_SINGLE_INVOICE_Output_Msg" />
  </operation>
</portType>
```

非同期要求応答操作パターンの場合、同期的に生成された他の操作と区別するために、**ASYNCH** が入力メッセージと出力(応答)メッセージの両方に表示されます。

たとえば、**CREATE_INVOICE** 操作を非同期に生成した場合、特に **CREATE_INVOICE** 操作用に **ASYNCH** が入力メッセージと出力(応答)メッセージの両方に表示されます。

```
...
<portType name="AR_INVOICE_API_PUB_PortType">
  <operation name="CREATE_INVOICE_**ASYNCH**">
    <input name="tns:CREATE_INVOICE_Input_Msg"/>
  </operation>
</portType>
<portType name="AR_INVOICE_API_PUB_Callback_PortType">
  <operation name="CREATE_INVOICE_**ASYNCH**_RESPONSE">
    <input name="tns:CREATE_INVOICE_Output_Msg"/>
  </operation>
</portType>
<portType name="AR_INVOICE_API_PUB_CallBack_PortType">
  <operation name="CREATE_INVOICE_**ASYNCH**_RESPONSE">
    <input message="tns:CREATE_INVOICE_**ASYNCH**_Output_Msg"/>
  </operation>
  <operation name="CREATE_SINGLE_INVOICE_**ASYNCH**_RESPONSE">
    <input message="tns:CREATE_SINGLE_INVOICE_**ASYNCH**_Output_Msg"/>
  </operation>
</portType>
```

binding

binding は、特定の *portType* によって定義されている操作およびメッセージのメッセージ形式およびプロトコル詳細を定義します。*portType* 操作の Web 経由での実際の送信方法に関して具体的な詳細を指定します。

port は、バインディング用の単一のアドレスを指定して個別のエンドポイントを定義します。

この要素の構造は、次のようになります。

```

<wsdl:definitions...>
  <wsdl:binding name="nmtoken" type="qname">*
    <wsdl:operation name="nmtoken"/>
    <wsdl:input> ?
    </wsdl:input>
    <wsdl:output>?
    </wsdl:output>
    <wsdl:fault name="nmtoken"? message="qname"?>
    </wsdl:fault>
  </wsdl:operation>
</wsdl:binding>
</wsdl:definitions>

```

同じ例の **binding** 要素は、次に示すように、**PortType**

AR_INVOICE_API_PUB_PortType の SOAP バインディングを記述します。

```

<binding name="AR_INVOICE_API_PUB_Binding" type="tns:
AR_INVOICE_API_PUB_PortType">
  <soap:binding style="document" transport="http://schemas.xmlsoap.
org/soap/http"/>
  <wsp:PolicyReference URI="#wss_username_token_service_policy"
wsdl: required="false"/>
  <operation name="CREATE_INVOICE">
    <soap:operation soapAction="CREATE_INVOICE" />
    <input>
      <soap:header message="tns:CREATE_INVOICE_Input_Msg"
part="header" use="literal" />
      <soap:body parts="body" use="literal" />
    </input>
    <output>
      <soap:body use="literal" />
    </output>
  </operation>
  <operation name="CREATE_SINGLE_INVOICE">
    <soap:operation soapAction="CREATE_SINGLE_INVOICE" />
    <input>
      <soap:header message="tns:CREATE_SINGLE_INVOICE_Input_Msg"
part="header" use="literal" />
      <soap:body parts="body" use="literal" />
    </input>
    <output>
      <soap:body use="literal" />
    </output>
  </operation>
  <operation name="CREATE_INVOICE_ASYNC">
    <soap:operation soapAction="CREATE_INVOICE_ASYNC" />
    <input>
      <soap:header message="tns:CREATE_INVOICE_ASYNC_Input_Msg"
part="header" use="literal"/>
      <soap:body parts="body" use="literal" />
    </input>
    <output>
      <soap:body use="literal" />
    </output>
  </operation>
  <operation name="CREATE_SINGLE_INVOICE_ASYNC">
    <soap:operation soapAction="CREATE_SINGLE_INVOICE_ASYNC" />
    <input>
      <soap:header message="tns:CREATE_SINGLE_INVOICE_ASYNC_Input_Msg"
part="header" use="literal"/>
      <soap:body parts="body" use="literal" />
    </input>
    <output>
      <soap:body use="literal" />
    </output>
  </operation>
</binding>

```

非同期要求応答操作パターンの場合、**binding** 要素は、次に示すように、**PortType**

AR_INVOICE_API_PUB_Callback_PortType の SOAP バインディングを記述します。

```

<binding name="AR_INVOICE_API_PUB_Callback_Binding" type="tns:
AR_INVOICE_API_PUB_Callback_PortType">
  <soap:binding style="document" transport="http://schemas.xmlsoap.
org/soap/http"/>
    <operation name="CREATE_INVOICE_ASYNCH_RESPONSE">
      <soap:operation soapAction="CREATE_INVOICE_ASYNCH_RESPONSE"/>
      <input>
        <soap:body use="literal" namespace="http://xmlns.oracle.
com/apps/ar/soapprovider/plsql/ar_invoice_api_pub/CREATE_INVOICE_ASYNCH
"/>
      </input>
    </operation>
    <operation name="CREATE_SINGLE_INVOICE_ASYNCH_RESPONSE">
      <soap:operation soapAction="CREATE_SINGLE_INVOICE_ASYNCH_RESPONSE"
/>
      <input>
        <soap:body use="literal" namespace="http://xmlns.oracle.
com/apps/ar/soapprovider/plsql/ar_invoice_api_pub/CREATE_SINGLE_INVOICE
_ ASYNCH"/>
      </input>
    </operation>
  </binding>

```

binding は、常に SOAP over HTTP バインディングのドキュメント形式になります。また、ここには SOAP ヘッダーと SOAP 本文のコンテンツも定義します。

注意: これはドキュメント形式のサービス(style="document")であるため、要求メッセージおよび応答メッセージは、リモート・プロシージャ・コール(RPC 形式)の Web サービスに必要なラッパー要素を使用するのではなく、単に XML 文書で構成されています。transport 属性は、SOAP メッセージのトランスポートが SOAP HTTP を介して実行されることを示します。

soap:operation 要素は、特定の操作(CREATE_INVOICE など)を特定の SOAP 実装にバインドすることを示します。soapAction 属性は、SOAPAction HTTP ヘッダーをサービスの識別に使用することを指定します。

soap:header 要素は、SOAP エンベロープの Header 要素で送信されるヘッダーを定義できます。SOAHeader は、Types 要素内に Responsibility、RespApplication、SecurityGroup、NLSLanguage、Org_Id の各複合型で構成されます。

soap:body 要素は、特定の操作の入力メッセージおよび出力メッセージの詳細を指定できます。

service

service 要素は、Web サービスを定義します。これは、1 つ以上の **Port** 要素で構成されています。ポートは、バインディング用の単一のアドレスを指定することで個別のエンドポイントを定義します。

サービス・バインディングは、一般に SOAP を使用して作成されます。

この要素の構造は、次のようになります。

```
<wsdl:definitions...>
  <wsdl:service name="nmtoken">*
    <wsdl:port name="nmtoken" binding="qname"> *
      </wsdl:port>
    </wsdl:service>
  </wsdl:definitions>
```

配置済サービスの場合、*Service* 要素 AR_INVOICE_API_PUB_Service は、portType AR_INVOICE_API_PUB_PortType の soa-infra にサービスがホストされているサービス・エンドポイントの物理的場所を定義します。

```
<service name="AR_INVOICE_API_PUB_Service">
  <port name="AR_INVOICE_API_PUB_Port" binding="tns:
AR_INVOICE_API_PUB_Binding">
    <soap:address location="http://<soa_suite_hostname>:<port>/soa-
infra/services/default/<jndi_name>_PLSQL_AR_INVOICE_API_PUB/AR_INVOICE
_A PI_PUB_Service/" />
  </port>
</service>
```

注意: 生成されたがまだ配置されていないサービスの場合、soap:address location 要素に Not_Deployed が表示されます(<soap:address location="#NOT_DEPLOYED#" />など)。

WADL 要素詳細のレビュー

インタフェースを REST サービスとして公開している場合、対応する WADL 記述を別のウィンドウに表示できます。

前述の WSDL 記述を説明した際と同じインタフェース例である PL/SQL API 請求書作成 (AR_INVOICE_API_PUB) を使用して説明します。このインタフェースは、REST サービスとしても公開できます。関連付けられている WADL 情報を表示するには、インタフェース詳細ページの「REST Web サービス」タブにある「**WADL の表示**」リンクをクリックします。WADL 文書が表示されます。

WADL 文書の構造

WADL (Web Application Description Language) は、HTTP ベースの Web アプリケーションを、マシンが処理可能な記述で提供するために設計されています。

application 要素は、WADL 記述のルートを形成します。ここには、次の要素を含めることができます。

- **grammars:** この要素は、WADL 文書で記述されているプロトコルの実行時に交換されるデータの定義用のコンテナとして機能します。
- **resources:** この要素は、アプリケーションから提供されるすべての子リソース要素を含めるためのコンテナとして機能します。

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<application xmlns:tns="http://xmlns.oracle.
com/apps/ar/soapprovider/plsql/rest/ar_invoice_api_pub/" xmlns="http:
//wadl.dev.java.net/2009/02"
xmlns:tns1="http://xmlns.oracle. com/apps/ar/rest/ar/create_invoice/"
name="AR_INVOICE_API_PUB"
targetNamespace="http://xmlns.oracle.
com/apps/ar/soapprovider/plsql/rest/ar_invoice_api_pub/">
  <grammars>
    ...
  </grammars>
  <resources base="http://<hostname>:<port>/webservicess/rest/Invoice/">
    ...
  </resources>
</application>
```

Grammars

この要素は、WADL 文書で記述されているプロトコルの実行時に交換されるデータの定義用のコンテナとして機能します。*include* 要素は、多くの場合、1 つ以上のデータ形式の記述の定義を含めることができるようにするために参照されます。

```
<grammars>
  <include xmlns="http://www.w3.org/2001/XMLSchema" href="https:
//<hostname>:<port>/webservicess/rest/Invoice/?
XSD=CREATE_INVOICE_SYNCH_TYPEDEF.xsd" />
  <include xmlns="http://www.w3.org/2001/XMLSchema" href="https:
//<hostname>:<port>/webservicess/rest/Invoice/?
XSD=CREATE_SINGLE_INVOICE_SYNCH_TYPEDEF.xsd" />
</grammars>
```

この例では、同期パターン CREATE_INVOICE_SYNCH および CREATE_SINGLE_INVOICE_SYNCH をサポートする 2 つのサービス操作が *Include* 要素で参照されています。

ここで強調表示されている **Invoice** は、以前、サービスを配置する前に入力されたサービスの別名です。サービスが正常に配置されたら、指定した別名(**Invoice**)は WADL のサービス・エンドポイントおよび XSD のネームスペースの一部になります。

Resources

resources 要素は、Web アプリケーションから提供されたリソース情報を表します。これには、含まれている子リソース識別子ごとにベース URI を提供する *base* 属性が含まれています。

たとえば、子の *resource* 要素はそれぞれ、選択したインタフェースに含まれている特定のサービス操作(*create_invoice* や *create_single_invoice* など)を表します。

```
<resources base="http://<hostname>:<port>/webservicess/rest/Invoice/">
  <resource path="/create_invoice/">
    ...
  </resource>
  <resource path="/create_single_invoice/">
    ...
  </resource>
</resources>
```

Resource

resources 要素には一連の *resource* 要素を含めることができます。各 *resource* 要素は REST サービス操作を表します。この例では、*create_invoice* および *create_single_invoice* が、子 *resource* 要素として含まれています。

各 *resource* 要素には、次の子要素を含めることができます。

- *method*: この要素は、リソースに適用できる HTTP プロトコル・メソッドに対する入力と出力を表します。

POST は PL/SQL およびコンカレント・プログラム REST サービス用に唯一サポートされているメソッドで、POST および GET は Java Bean サービスおよびアプリケーション・モジュール・サービス用にサポートされているメソッドです。

オープン・インタフェースの場合、サポートされるメソッドはインタフェースの方向に基づいて決まります。Inbound 方向のオープン・インタフェース表の場合、4 つすべての HTTP メソッド(GET、POST、PUT および DELETE)がサポートされています。それ以外の場合、Outbound 方向のオープン・インタフェース表およびオープン・インタフェース・ビューでは GET のみがサポートされています。

- *request*: この要素は、リソースに HTTP メソッドを適用する場合に含める入力を表します。

mediaType 要素は、XML や JSON など、入力パラメータ用にサポートされているメディア・タイプを示します。

- *response*: この要素は、リソースに対して HTTP メソッドを実行して出力された結果を表します。

出力結果用にサポートされているメディア・タイプは、XML および JSON です。

```

<resources base="http://<hostname>:<port>/webservices/rest/Invoice/">
  <resource path="/create_invoice/">
    <method id="CREATE_INVOICE" name="POST">
      <request>
        <representation mediaType="application/xml" type="tns1:
InputParameters" />
        <representation mediaType="application/json" type="tns1:
InputParameters" />
      </request>
      <response>
        <representation mediaType="application/xml" type="tns1:
OutputParameters" />
        <representation mediaType="application/json" type="tns1:
OutputParameters" />
      </response>
    </method>
  </resource>
  <resource path="/create_single_invoice/">
    <method id="CREATE_SINGLE_INVOICE" name="POST">
      <request>
        <representation mediaType="application/xml" type="tns2:
InputParameters" />
        <representation mediaType="application/json" type="tns2:
InputParameters" />
      </request>
      <response>
        <representation mediaType="application/xml" type="tns2:
OutputParameters" />
        <representation mediaType="application/json" type="tns2:
OutputParameters" />
      </response>
    </method>
  </resource>
</resources>

```

この例では、create_invoice および create_single_invoice の各リソース要素に対して HTTP メソッド POST を適用する場合、入力要求メッセージおよび出力応答メッセージはすべて、XML 形式および JSON 形式でサポートされています。

配置済 REST サービスのインタフェース・タイプが Java Bean サービスまたはアプリケーション・モジュール・サービスの場合は、GET と POST の両方を REST サービス操作でサポートされているメソッドとして示すことができます。たとえば、次の WADL 記述は、Rest Service Locator インタフェースに含まれている多数のメソッドのうちの 2 つを示しています。addGrant メソッドは POST HTTP メソッドで実装され、getOperations は GET HTTP メソッドで支援されています。

```

<resources base="http://<hostname>:<port>/webservices/rest/locator/">
  <resource path="/addGrant/">
    <method id="addGrant" name="POST">
      <request>
        <representation mediaType="application/xml" type="tns1:
addGrant_InputParameters" />
        <representation mediaType="application/json" type="tns1:
addGrant_InputParameters" />
      </request>
      <response>
        <representation mediaType="application/xml" type="tns1:
addGrant_OutputParameters" />
        <representation mediaType="application/json" type="tns1:
addGrant_OutputParameters" />
      </response>
    </method>
  </resource>
  ...
  <resource path="/getOperations/{irepClassName}"/>
    <param name="irepClassName" style="template" required="true"
type="xsd:string"/>
    <method id="getOperations" name="GET">
      <request>
        <param name="ctx_responsibility" type="xsd:string" style="query"
required="false" />
        <param name="ctx_resppapplication" type="xsd:string"
style="query" required="false" />
        <param name="ctx_securitygroup" type="xsd:string" style="query"
required="false" />
        <param name="ctx_nlslanguage" type="xsd:string" style="query"
required="false" />
        <param name="ctx_orgid" type="xsd:int" style="query" required="
false" />
      </request>
      <response>
        <representation mediaType="application/xml" type="tns3:
getOperations_OutputParameters" />
        <representation mediaType="application/json" type="tns3:
getOperations_OutputParameters" />
      </response>
    </method>
  </resource>
</resources>

```

前述の例の GET メソッドでは、必要に応じてサービス実行のパス変数およびアプリケーション・コンテキスト情報が含まれるようになっています。

注意: パス変数およびコンテキスト・パラメータは、HTTP GET メソッドに対してのみ適用できます。

- {irepClassName}は、次に指定するように、getOperations 操作のパス変数です。

```

<resource path="/getOperations/{irepClassName}"/>

```

パス変数は、<resource>タグの後かつ<method>タグの前に<param>タグを使用して定義します。HTTP GET メソッドが起動されると、サービス・クライアントは実行時にパス変数を実際の値に置き換えます。

たとえば、irepClassName の値が PLSQL:FND_PROFILE である場合は、PLSQL:FND_PROFILE がこの要求の HTTP URL に渡されます。

HTTP GET が実行される URL は次のとおりです。http://<hostname>:
<port>/webservices/rest/locator/getOperations/PLSQL:
FND_PROFILE

パス変数は、Java API ソース・ファイル内のインライン・アノテーション

@rep:key_param で識別されます。Java Bean サービスのアノテーション・ガイドラインについては、「Java Bean サービスに対するアノテーション」(ページ A-6)を参照してください。アプリケーション・モジュール・サービスのアノテーション・ガイドラインについては、「アプリケーション・モジュール・サービスに対するアノテーション」(ページ A-6)を参照してください。

- コンテキスト・パラメータは、Oracle E-Business Suite アプリケーション・コンテキストの初期化に必要な事前定義済パラメータです。ctx_responsibility、ctx_respapplication、ctx_securitygroup、ctx_nlslanguage、ctx_orgid などのこれらのパラメータは、HTTP GET メソッドおよび DELETE メソッドに対してのみ適用できます(オープン・インタフェース表の例を次に示します)。これらのパラメータは、RESTHeader 要素で要求ペイロード問合せ文字列とともに問合せ文字列として渡されます。

制御パラメータは、リソースまたはコレクション・リソースに使用できる事前定義済問合せパラメータです。たとえば、返されるレコードの数をページ編集用に制限するには、offset パラメータおよび limit パラメータを使用します。

offset=<number>&limit=<number>

- offset=0&limit5

これにより、レコード 0 の後の、応答の最初の 5 つのレコードが返されます。つまり、1 番目、2 番目、3 番目、4 番目および 5 番目のレコードです。

- offset=2&limit4

これにより、レコード 2 の後の、応答の最初の 4 つのレコードが返されます。つまり、3 番目、4 番目、5 番目および 6 番目のレコードです。

- http://<hostname>:
<port>/webservices/rest/empinfo/getAllReports/?
offset=0&limit=5

これにより、ログイン・ユーザーに直属の最初の 5 人の従業員が階層順序で返されます。

配置済 REST サービスが Inbound 方向のオープン・インタフェース表の場合は、サービス操作表に 4 つすべての HTTP メソッドが表示されます。次に示した AR Autoinvoice (関連付けられたコンカレント・プログラム内部名 RAXMTR)オープン・インタフェース表の WADL の例では、RA_INTERFACE_LINES_ALL 操作が 4 つすべての HTTP メソッドとともに実装され、関連付けられたコンカレント・プログラム SUBMIT_CP_RAXMTR が POST メソッドとともに実装されます。

```

    <resources base="http://<hostname>:<port>/webservices/rest/autoinvoice
/"><resource path ="RA_INTERFACE_LINES_ALL/">
    <method id="RA_INTERFACE_LINES_ALL_get" name="GET">
        <request>
            <param name="ctx_responsibility" type="xsd:string" style="query"
required="false"/>
            <param name="ctx_respapplication" type="xsd:string" style="
query" required="false" />
            <param name="ctx_securitygroup" type="xsd:string" style="query"
required="false" />
            <param name="ctx_nlslanguage" type="xsd:string" style="query"
required="false" />
            <param name="ctx_orgid" type="xsd:int" style="query" required="
false" />
            <param name="select" type="xsd:string" style="query" required="
false"/>
            <param name="filter" type="xsd:string" style="query" required="
false"/>
            <param name="sort" type="xsd:string" style="query" required="false"
/>
            <param name="offset" type="xsd:string" style="query" required="
false"/>
            <param name="limit" type="xsd:string" style="query"
required="false" />
        </request>
        <response>
            <representation mediaType="application/xml" type="tns1:
RA_INTERFACE_LINES_ALL_Output"/>
            <representation mediaType="application/json" type="tns1:
RA_INTERFACE_LINES_ALL_Output" />
            <representation mediaType="text/csv" type="tns1:
RA_INTERFACE_LINES_ALL_Output"/>
        </response>
    </method>
    <method id="RA_INTERFACE_LINES_ALL_post" name="POST">
        <request>
            <representation mediaType="application/xml" type="tns1:
RA_INTERFACE_LINES_ALL_Input"/>
            <representation mediaType="application/json" type="tns1:
RA_INTERFACE_LINES_ALL_Input" />
            <representation mediaType="text/csv" type="tns1:
RA_INTERFACE_LINES_ALL_Input"/>
        </request>
        <response>
            <representation mediaType="application/xml" type="tns1:
RA_INTERFACE_LINES_ALL_Output"/>
            <representation mediaType="application/json" type="tns1:
RA_INTERFACE_LINES_ALL_Output" />
            <representation mediaType="text/csv" type="tns1:
RA_INTERFACE_LINES_ALL_Output"/>
        </response>
    </method>
    <method id="RA_INTERFACE_LINES_ALL_put" name="PUT">
        <request>
            <representation mediaType="application/xml" type="tns1:
RA_INTERFACE_LINES_ALL_Input"/>
            <representation mediaType="application/json" type="tns1:
RA_INTERFACE_LINES_ALL_Input" />
            <representation mediaType="text/csv" type="tns1:
RA_INTERFACE_LINES_ALL_Input"/>
        </request>
        <response>
            <representation mediaType="application/xml" type="tns1:
RA_INTERFACE_LINES_ALL_Output"/>
            <representation mediaType="application/json" type="tns1:

```

```

RA_INTERFACE_LINES_ALL_Output" />
<representation mediaType="text/csv" type="tns1:
RA_INTERFACE_LINES_ALL_Output"/>
</response>
</method>
<method id="RA_INTERFACE_LINES_ALL_delete" name="DELETE">
  <request>
    <param name="ctx_responsibility" type="xsd:string" style="query"
required="false"/>
    <param name="ctx_resppapplication" type="xsd:string" style="
query" required="false" />
    <param name="ctx_securitygroup" type="xsd:string" style="query"
required="false" />
    <param name="ctx_nlslanguage" type="xsd:string" style="query"
required="false" />
    <param name="ctx_orgid" type="xsd:int" style="query" required="
false" />
    <param name="filter" type="xsd:string" style="query" required="
false"/>
  </request>
  <response>
    <representation mediaType="application/xml" type="tns1:
RA_INTERFACE_LINES_ALL_Output"/>
    <representation mediaType="application/json" type="tns1:
RA_INTERFACE_LINES_ALL_Output" />
    <representation mediaType="text/csv" type="tns1:
RA_INTERFACE_LINES_ALL_Output"/>
  </response>
</method></resource>

```

- 選択したオープン・インタフェース AR Autoinvoice に含まれている各オープン・インタフェース表名は、選択した HTTP メソッド(複数可)とともに 1 つのリソース・エントリ (<resource path>)に表示されます。この例では、4 つのメソッド(GET、POST、PUT および DELETE)が選択されている表名 RA_INTERFACE_LINES_ALL と、POST のある関連付けられたコンカレント・プログラム SUBMIT_CP_RAXMTR とがそれぞれ別のリソース・エントリに含まれています。
- オープン・インタフェース・ビューの WADL 記述には、GET メソッドのみを持つリソース・エントリが 1 つのみ含まれています。オープン・インタフェース・ビューに関連付けられているコンカレント・プログラム SUBMIT_CP_<internal name of the concurrent program>はありません。

SOAP メッセージについて

SOAP (Simple Object Access Protocol)は、コンピュータ・ネットワークでの Web サービスの実装において、構造化された情報を交換するための軽量な XML ベース・プロトコル仕様です。たとえば、Web サービス・プロバイダは、Web サービス・クライアントから Web サービスの起動を求める SOAP 要求を受信して、対応する SOAP 応答をクライアントに送信します。

SOAP メッセージの構造

SOAP メッセージは、エンベロープと呼ばれる SOAP コンポーネントの 1 つに含まれます。SOAP エンベロープは、メッセージに何を含め、だれがメッセージを処理し、メッセージがオプションか必須かを記述するための全体的なフレームワークを定義します。これは、次の要素で構成されています。

- ヘッダー(オプション)

エンベロープ要素には、オプションでヘッダー要素を含めることができます。エンベロープにヘッダー要素を含める場合、1つのみ含め、それをエンベロープの最初の子にする必要があります。ヘッダー要素の第1レベルの子要素は、ヘッダー・ブロックと呼ばれます。

ヘッダー・ブロックは、次のメカニズムで使用できます。

- 特定の受信者を対象にしたセキュリティ関連の情報を添付するために使用できます。

詳細は、「SOAP セキュリティ・ヘッダー」(ページ 2-27)を参照してください。

- サービスに必要なアプリケーション・コンテキスト値を設定するために使用できます。

詳細は、「アプリケーション・コンテキストの SOAP ヘッダー」(ページ 2-31)を参照してください。

- XML ゲートウェイ・インバウンド取引の正常な完了に必要なヘッダー変数を移入するために使用できます。

詳細は、「XML ゲートウェイ・メッセージの SOA ヘッダー」(ページ 2-34)を参照してください。

- 本文

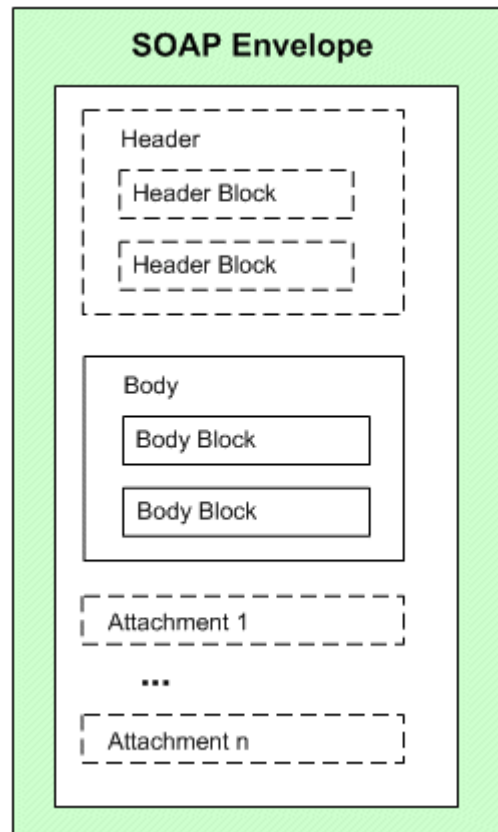
どのエンベロープ要素にも、メッセージを保持する本文要素が1つのみ含まれている必要があります。本文要素の直下の子要素は、本文ブロックまたは本文パートと呼ばれます。

- 添付(オプション)

1つの SOAP メッセージで複数の添付を転送でき、これらの添付はテキスト、バイナリ、画像など任意のタイプにすることができます。

次の図は、SOAP メッセージの構造を示しています。

SOAP メッセージの構造



SOAP メッセージのスケルトンは、次のようになります。

```
<xml version="1.0">
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

  <soap:Header>
    ...
  </soap:Header>

  <soap:Body>
    ...
    <soap:Fault>
      ...
    </soap:Fault>
  </soap:Body>

  </soap:Envelope>
```

SOAP セキュリティ・ヘッダー

Oracle WebLogic 管理対象サーバーの配置済 SOA コンポジットに対する SOAP 要求メッセージを Oracle SOA Suite から受信すると、SOAP メッセージは Oracle E-Business Suite の JAAS (Java Authentication and Authorization Service) ベースのログイン・モジュールによって認証されます。

SOAP メッセージは、配置前にサービス用に対して以前に識別された UsernameToken または SAML トークン・セキュリティ・モデルを使用して認証されます。選択した認証情報は、wsse:security Web セキュリティ・ヘッダーに埋め込まれます。

UsernameToken ベースの SOAP セキュリティ・ヘッダー

UsernameToken ベースの SOAP ヘッダーには、次の wsse:security セクションを含める必要があります。

```
<soapenv:Header>
<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
soapenv:mustUnderstand="1">
  <wsse:UsernameToken>
    <wsse:Username>Username</wsse:Username>
    <wsse:Password>Password</wsse:Password>
  </wsse:UsernameToken>
</wsse:Security>

</soapenv:Header>
```

注意: <wsse:security>ヘッダーに mustUnderstand="1" 属性が含まれているときに、その<wsse:security>ヘッダー・ブロックが含まれているセキュリティ・トークンを対応する WS SOAP メッセージ・セキュリティ・トークン・プロファイルに従って解釈または処理できない場合、受信者は fault を生成する必要があります。

「サンプルの障害 SOAP 応答」(ページ 2-41)を参照してください。

SOAP 要求の一般的な WS セキュリティ・ヘッダーは、次のようになります。

```
<soapenv:Header>
<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
soapenv:mustUnderstand="1">
  <wsse:UsernameToken>
    <wsse:Username>myUser</wsse:Username>
    <wsse:Password
      Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText">password</wsse:Password>
    </wsse:Password>
  </wsse:UsernameToken>
</wsse:Security>

</soapenv:Header>
```

UsernameToken ベースのセキュリティには UsernameToken プロファイルが含まれており、その Web サービス・セキュリティ・ヘッダーにはユーザー名とパスワード情報が指定されています。ユーザー名はクリア・テキストで、パスワードは UsernameToken プロファイルの最も機密性が高い部分です。このセキュリティ・モデルでサポートされているパスワード・タイプは、プレーン・テキスト・パスワード(PasswordText)です。

注意: PasswordText パスワード・タイプは、クリア・テキストで記述されたパスワードです。

Web サービスを起動する SOAP 要求には、ユーザー名とプレーン・テキスト・パスワードで構成されるセキュリティ・ヘッダーを含める必要があります。SOAP 要求の一部として実行時に受信したパスワードは、Oracle E-Business Suite に格納された暗号化パスワードに対して検証されます。検証後に、SOAP 要求からのプレーン・テキスト・パスワードは破棄されます。

SOAP メッセージの SOAP ヘッダーにあるユーザー名/パスワードが渡されて、Web サービスが認証されます。ここで説明する `wsse:security` のユーザー名/パスワードは、Oracle E-Business Suite ユーザー名/パスワード(またはアプリケーション・ユーザーを定義する際に「ユーザー」ウィンドウで作成したユーザー名/パスワード)です。

Oracle E-Business Suite Web サービスを正常に起動するには、SOAP 要求とともにセキュリティ・ヘッダー要素を渡す必要があります。

これらのセキュリティ・ヘッダー値を渡さないと、Web サービスが認証されず、サービスの実行が失敗します。

Oracle E-Business Suite Web サービスを起動するときにセキュリティ・ヘッダーを渡す方法の詳細な手順は、「Web サービス・ポリシーの構成」(ページ 3-34)を参照してください。

SAML トークン・ベースの SOAP セキュリティ・ヘッダー

Security Assertion Markup Language (SAML)は、セキュリティ・ドメイン間、つまりアイデンティティ・プロバイダとサービス・プロバイダとの間で認証データおよび認可データを交換するための XML ベースの標準です。

Web アプリケーションによって SAML を認証メカニズムとして使用するサービスが起動されると、SAML アサーションを含むか参照しているこの SOAP 要求メッセージを Oracle SOA Suite から受信し、それが Oracle E-Business Suite の JAAS ベースのログイン・モジュールに渡されて、`wsse:security` Web セキュリティ・ヘッダーに基づいて認証されます。アサーションの検証および処理の一環として、受信者または Oracle E-Business Suite のログイン・モジュールはサブジェクトと、参照先の SAML アサーションの要求と、文に対して定義された確認メソッドを満たす証拠を提供するエンティティとの間に関係を確立する必要があります。

信頼できるエンティティは、`sender-vouches` 確認メソッドを使用して、SAML 文のサブジェクトの代理として機能できるように、`sender-vouches SubjectConfirmation` 要素を設定します。

次の SOAP の例に示した信頼できるエンティティは、`<ds:Signature>`要素が関連付けられている `sender-vouches` サブジェクト確認メソッドを使用して、アイデンティティを確立し、サブジェクトの代理としてメッセージ本文を送信したことをアサートしています。

```

<soapenv:Envelope
xmlns:fnd="http://xmlns.oracle.
com/apps/fnd/soapprovider/plsql/fnd_user_pkg/" xmlns:soapenv="http:
//schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.
org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <ds:Signature Id="Signature-26598842" xmlns:ds="http://www.w3.
org/2000/09/xmldsig#">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-
exc-c14n#" />
          <ds:SignatureMethod Algorithm="http://www.w3.
org/2000/09/xmldsig#rsa-sha1" />
          <ds:Reference URI="#id-31755621">
            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
            </ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"
/ >
          <ds:DigestValue>hbb/y+b3whhaFakWGO+bnkNm5/Q=</ds:DigestValue>
        </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>

        jiXB+bsTfqd0uYxnaPAJcooCGb9UrKfzqSlGu/lE0nbL+sPkQQzmaB+ZKMFxUAc5pJStyeBu
        3DIg
        6bEXSknB3JeJaHy6UFeGKZz3ROf4WKqRvDLXsa10Ei6Id66go3goqYzYtoUA4J43MjLJbKUw
        5KG/
        LGBImRKABFPRP4qlAlQ=
        </ds:SignatureValue>
        <ds:KeyInfo Id="KeyId-1042529">
          <wsse:SecurityTokenReference wsu:Id="STRId-6382436" xmlns:wsu="http:
//docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd"><wsse:KeyIdentifier
            EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary"
            ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509SubjectKeyIdentifier"
            >ADoNKKuduSTKTwi7jqEzCxd7JU=</wsse:KeyIdentifier></wsse:
SecurityTokenReference>
          </ds:KeyInfo></ds:Signature>
          <Assertion AssertionID="be7d9814c36381c27fefa89d8f27e126"
            IssueInstant="2010-02-27T17:26:21.241Z" Issuer="www.oracle.com"
            MajorVersion="1" MinorVersion="1" xmlns="urn:oasis:names:tc:SAML:1.0:
assertion" xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion" xmlns:
samlp="urn:oasis:names:tc:SAML:1.0:protocol" xmlns:xsd="http://www.w3.
org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"><Conditions NotBefore="2010-02-27T17:26:21.241Z"
NotOnOrAfter="2011-02-27T17:26:21.241Z"/>
          <AuthenticationStatement AuthenticationInstant="2010-02-27T17:26:
21.241Z" AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
            <Subject>
              <NameIdentifier Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:unspecified"
                NameQualifier="notRelevant">SYSADMIN</NameQualifier>
              <SubjectConfirmation>
                <ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:sender-
vouches</ConfirmationMethod>
              </SubjectConfirmation>
            </Subject>
          </AuthenticationStatement>
        </Assertion>
      </wsse:Security>

      <fnd:SOAHeader>

```



```

<!--Optional:-->
    <fnd:Responsibility>UMX</fnd:Responsibility>
    <!--Optional:-->
    <fnd:RespApplication>FND</fnd:RespApplication>

</fnd:SOAHeader>
</soapenv:Header>

<soapenv:Body wsu:Id="id-31755621" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
    <tes:InputParameters xmlns:tes="http://xmlns.oracle.com/apps/fnd/soapprovider/plsql/fnd_user_pkg/testusername/">
        <!--Optional:-->
        <tes:X_USER_NAME>AMILLER</tes:X_USER_NAME>
    </tes:InputParameters>
</soapenv:Body>
</soapenv:Envelope>

```

注意: SAMLトークン・ベースのセキュリティを使用すると、シングル・サインオン (SSO)環境と SSO 非対応の環境の両方でユーザーを認証できます。SAML アサーションの NameIdentifier という形式は、ユーザーを LDAP に照らして認証しているか(SSO ユーザーの場合)、Oracle E-Business Suite FND_USER 表に照らして認証しているか(非 SSO ユーザーの場合)を示しています。

前述の SOAP メッセージの SAML アサーションは、SSO 非対応の環境向けです。NameIdentifier タグのユーザー名が次のように LDAP DN 形式である場合、ユーザー名は SSO ユーザーの登録済 OID で検証されます。

```

<NameIdentifier Format="urn:oasis:names:tc:SAML:1.1:
nameid-format:unspecified"
    NameQualifier="notRelevant"
>orclApplicationCommonName=PROD1,cn=EBusiness,cn=Products,
cn=OracleContext,dc=us,dc=oracle,dc=com</NameIdentifier>

```

SAMLトークン sender-vouches ベースのセキュリティの詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「SAMLsender-vouchesトークン・ベースのセキュリティ」を参照してください。

アプリケーション・コンテキストの SOAP ヘッダー

アプリケーション・コンテキストには、Oracle E-Business Suite で必要な値を渡す場合に使用される多数の重要な要素が含まれています。たとえば、十分な権限を持つ Oracle E-Business Suite ユーザーが API トランザクションまたはコンカレント・プログラムを実行するには、その API トランザクションまたはコンカレント・プログラムのコンテキスト・ヘッダー情報が必要です。

SOAP 要求の SOAHeader 部分のアプリケーション・コンテキスト

PL/SQL サービスおよびコンカレント・プログラム・サービスの SOAP 要求の SOAHeader 部分に定義されているこれらのコンテキスト・ヘッダー要素は次のとおりです。

- Responsibility

これは、Oracle E-Business Suite アプリケーションの職責情報です。これは、responsibility_key (SYSTEM_ADMINISTRATOR など)を値として受け入れます。

- **RespApplication**

これは、職責アプリケーション短縮名の情報です。これは、アプリケーション短縮名(FND など)を値として受け入れます。

- **SecurityGroup**

これは、セキュリティ・グループ・キー(STANDARD など)を値として受け入れます。

- **NLSLanguage (オプション)**

これは、PL/SQL サービスおよびコンカレント・プログラム・サービスの SOAP 要求の SOAHeader 部分で渡されるオプションのパラメータです。

NLS 言語の要素が指定された場合、渡された言語で SOAP 要求を使用できます。すべての対応する SOAP 応答とエラー・メッセージも、同じ言語で戻すことができます。言語が指定されていない場合、ユーザーのデフォルトの言語が使用されます。

- **Org_Id (PL/SQL サービスおよびコンカレント・プログラム・サービスの場合はオプション)**

これは、PL/SQL サービスおよびコンカレント・プログラム・サービスの SOAP 要求の SOAHeader 部分で渡されるオプションのパラメータです。サービスの実行が特定組織に依存する場合は、その SOAP 要求の Org_Id 要素を渡す必要があります。

次の SOAP メッセージでは、SOAHeader 部分を太字で出力しています。

```
<soapenv:Header>
<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd"
soapenv:mustUnderstand="1">
  <wsse:UsernameToken>
    <wsse:Username>sysadmin</wsse:Username>
    <wsse:Password
      Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-1.0#PasswordText">password</wsse:Password>
    </wsse:UsernameToken>
  </wsse:Security><ozf:SOAHeader>
    <ozf:Responsibility>OZF_USER</ozf:Responsibility>
    <ozf:RespApplication>OZF</ozf:RespApplication>
    <ozf:SecurityGroup>STANDARD</ozf:SecurityGroup>
    <ozf:NLSLanguage>AMERICAN</ozf:NLSLanguage>
    <ozf:Org_Id>204</ozf:Org_Id>
  </ozf:SOAHeader>
</soapenv:Header>
```

SOAP 要求の ServiceBean_Header 部分のアプリケーション・コンテキスト

ビジネス・サービス・オブジェクト・サービスの SOAP 要求の ServiceBean_Header 部分に定義されているこれらのコンテキスト・ヘッダー要素は次のとおりです。

- **RESPONSIBILITY_NAME**

これは、Oracle E-Business Suite アプリケーションの職責情報です。これは、名前 (Responsibility_Name、たとえば System Administrator) とキー ({key}responsibility_key の形式、たとえば {key}SYSTEM_ADMINISTRATOR) の両方を値として受け入れることができます。

- **RESPONSIBILITY_APPL_NAME**

これは、職責アプリケーション短縮名の情報です。これは、アプリケーション短縮名(FND など)を値として受け入れます。

- **SECURITY_GROUP_NAME**

これは、セキュリティ・グループ・キー(STANDARD など)を値として受け入れます。

- **NLSLanguage (オプション)**

これは、ビジネス・サービス・オブジェクト・サービスの SOAP 要求の ServiceBean_Header 部分で渡されるオプションのパラメータです。

NLS 言語の要素が指定されている場合(AMERICAN など)、渡された言語で SOAP 要求を使用できます。すべての対応する SOAP 応答とエラー・メッセージも、同じ言語で戻すことができます。言語が指定されていない場合、ユーザーのデフォルトの言語が使用されます。

- **Org_Id (オプション)**

これは、ビジネス・サービス・オブジェクト・サービスの SOAP 要求の ServiceBean_Header 部分で渡されるオプションのパラメータです。

サービスの実行が特定組織に依存する場合は、その SOAP 要求の Org_Id 要素を渡す必要があります。

次の SOAP 要求の例では、ビジネス・サービス・オブジェクト・サービスの ServiceBean_Header 部分を太字で出力しています。

```

<soapenv:Envelope xmlns:ser="http://xmlns.oracle.
com/apps/fnd/ServiceBean"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="
http://xmlns.oracle.com/apps/fnd/rep/ws">
  <soapenv:Header>
    <wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.
oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
1.0.xsd">
      <wsse:UsernameToken wsu:Id="UsernameToken-22948433"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility- 1.0.xsd">
        <wsse:Username>sysadmin</wsse:Username>
        <wsse:Password Type="http://docs.oasis-
open.org/wss/2004/01/oasis- 200401-wss-username-token-profile-
1.0#PasswordText">password</wsse: Password>
      </wsse:UsernameToken>
    </wsse:Security><ser:ServiceBean_Header>
      <ser:RESPONSIBILITY_NAME>System Administrator</ser:
RESPONSIBILITY_NAME>
      <ser:RESPONSIBILITY_APPL_NAME>sysadmin</ser:
RESPONSIBILITY_APPL_NAME>
      <ser:SECURITY_GROUP_NAME>standard</ser:SECURITY_GROUP_NAME>
      <ser:NLS_LANGUAGE>american</ser:NLS_LANGUAGE>
      <ser:ORG_ID>202</ser:ORG_ID>
    </ser:ServiceBean_Header>
  </soapenv:Header>
<soapenv:Body>
  <ws:IntegrationRepositoryService_GetInterfaceByType>
    <interfaceType>XMLGATEWAY</interfaceType>
  </ws:IntegrationRepositoryService_GetInterfaceByType>
</soapenv:Body>
</soapenv:Envelope>

```

XML ゲートウェイ・メッセージの SOA ヘッダー

Oracle XML Gateway では、各取引先に Oracle E-Business Suite ユーザーが構成されています。「取引先の設定」フォームに定義されている承認済ユーザーのみが、XML トランザクションを実行できます。外部クライアントは、このようなユーザー名を SOAP 本文の <ECX:SOAHeader> 要素(一般 XML ゲートウェイ・サービスの場合は <XMLGateway_Header> 要素)内に定義された <USERNAME> 要素および <PASSWORD> 要素で渡すことができます。これらのユーザー名パラメータは、トランザクションを開始する前に、Oracle XML Gateway によって、取引先設定に定義されているユーザー名に照らして検証されます。

このため、XML ゲートウェイ・インタフェース・タイプの場合、取引先レベルと、SOAP 要求の wsse: security ヘッダーで渡されたユーザー名の両方で認可チェックが実行されます。取引先設定およびユーザーを取引先に関連付ける方法の詳細は、*Oracle XML Gateway ユーザーズ・ガイド*を参照してください。

次のコード・スニペットは、XML ゲートウェイ・インバウンド・メッセージの SOAP 要求内にある SOAHeader 要素を示しています。

```

<soapenv: Envelope xmlns:ecx="http://xmlns.oracle.
com/apps/ecx/soapprovider/xmlgateway/ecx_cbodi/"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sys="http://xmlns.oracle.com/xdb/SYSTEM">
  <soapenv:Header>
    <wsse:Security soapenv:mustUnderstand="1"
      xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd">
      <wsse:UsernameToken wsu:Id="UsernameToken-10586449"
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">
        <wsse:Username>SYSADMIN</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.
org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0
#PasswordText">password</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
    <ecx:SOAHeader>
      <sys:ECXMSG>
        <MESSAGE_TYPE></MESSAGE_TYPE>
        <MESSAGE_STANDARD></MESSAGE_STANDARD>
        <TRANSACTION_TYPE></TRANSACTION_TYPE>
        <TRANSACTION_SUBTYPE></TRANSACTION_SUBTYPE>
        <DOCUMENT_NUMBER></DOCUMENT_NUMBER>
        <PARTYID></PARTYID>
        <PARTY_SITE_ID></PARTY_SITE_ID>
        <PARTY_TYPE></PARTY_TYPE>
        <PROTOCOL_TYPE></PROTOCOL_TYPE>
        <PROTOCOL_ADDRESS></PROTOCOL_ADDRESS>
        <USERNAME></USERNAME>
        <PASSWORD></PASSWORD>
        <ATTRIBUTE1></ATTRIBUTE1>
        <ATTRIBUTE2></ATTRIBUTE2>
        <ATTRIBUTE3></ATTRIBUTE3>
        <ATTRIBUTE4></ATTRIBUTE4>
        <ATTRIBUTE5></ATTRIBUTE5>
      </sys:ECXMSG>
    </ecx:SOAHeader>
  </soapenv:Header>

```

次の表では、SOAP 要求内の SOAHeader 要素(一般 XML ゲートウェイ・サービスの場合は XMLGateway_Header 要素)に定義されている XML ゲートウェイ・ヘッダー・パラメータについて説明します。

SOAP 要求内の SOAHeader 要素にある XML ゲートウェイ・ヘッダー情報

属性	説明
MESSAGE_TYPE	ペイロード・メッセージの形式。これは、XML にデフォルト設定されます。現在、Oracle XML Gateway では XML のみがサポートされています。
MESSAGE_STANDARD	「XML 標準フォームの定義」に入力されて「取引フォームの定義」に表示されるメッセージ形式の標準。これは、OAG にデフォルト設定されます。インバウンド XML 文書に入力されたメッセージ標準は、取引先の設定のメッセージ標準と同じである必要があります。

属性	説明
TRANSACTION_TYPE	取引先の表からのビジネス文書の外部取引タイプ。インバウンド XML 文書の取引タイプは、取引先フォームで定義された取引タイプと同じである必要があります。
TRANSACTION_SUBTYPE	取引先の表からのビジネス文書の外部取引サブタイプ。インバウンド XML 文書の取引サブタイプは、取引先フォームで定義された取引サブタイプと同じである必要があります。
DOCUMENT_NUMBER	発注書や請求書番号などの、取引を識別するために使用される文書識別子。このフィールドは、XML ゲートウェイでは使用されませんが、インバウンド・メッセージで渡すことができます。
PROTOCOL_TYPE	伝送プロトコルは、取引先表に定義されています。
PROTOCOL_ADDRESS	伝送アドレスは、取引先表に定義されています。
USERNAME	USERNAME は、取引先表に定義されています。
PASSWORD	USERNAME に関連付けられているパスワードは、取引先表に定義されています。
PARTY_SITE_ID	インバウンド XML 文書のパーティ・サイト識別子は、取引先フォームで定義されたソース取引先の場所と同じである必要があります。
ATTRIBUTE1	このパラメータは、ベース・アプリケーションによって定義されることがあります。
ATTRIBUTE2	このパラメータは、ベース・アプリケーションによって定義されることがあります。

属性	説明
ATTRIBUTE3	アウトバウンド・メッセージの場合、このフィールドには取引先表の目的取引先ロケーション・コードからの値が設定されます。インバウンド・メッセージの場合、この値が存在していると、別の XML メッセージが生成されて、取引先表の目的取引先ロケーション・コードで識別される取引先に送信されます。XML メッセージを XML メッセージの最終受信者に転送するには、この値がハブで認識される必要があります。 注意: 詳細は、<i>Oracle XML Gateway ユーザーズ・ガイド</i>の <i>目的取引先ロケーション・コード</i>を参照してください。
ATTRIBUTE4	このパラメータは、ベース・アプリケーションによって定義されることがあります。
ATTRIBUTE5	このパラメータは、ベース・アプリケーションによって定義されることがあります。

注意: PARTYID パラメータおよび PARTY_TYPE パラメータは使用しません。

次のコード・スニペットは、一般 XML ゲートウェイ・サービスの SOAP 要求内にある XMLGateway_Header 要素を示しています。

```
<soap:Envelope>
  <soap:Header>
    ...
    <ns1:XMLGateway_Header
      xmlns:ns1="http://xmlns.oracle.com/apps/fnd/XMLGateway
      soapenv:mustUnderstand="0">
      <ns1:MESSAGE_TYPE>XML</ns1:MESSAGE_TYPE>
      <ns1:MESSAGE_STANDARD>OAG</ns1:MESSAGE_STANDARD>
      <ns1:TRANSACTION_TYPE>PO</ns1:TRANSACTION_TYPE>
      <ns1:TRANSACTION_SUBTYPE>PROCESS</ns1:TRANSACTION_SUBTYPE>
      <ns1:DOCUMENT_NUMBER>123</ns1:DOCUMENT_NUMBER>
      <ns1:PARTY_SITE_ID>4444</ns1:PARTY_SITE_ID>
    </ns1:XMLGateway_Header>
  </soap:Header>
  ...
</soap:Envelope>
```

次の表では、SOAP 要求の XMLGateway_Header 部分にある XML ゲートウェイ・ヘッダー情報について説明します。

SOAP 要求の XMLGateway_Header 部分

パラメータ名	説明
MESSAGE_TYPE	ペイロード・メッセージの形式。これは、XML にデフォルト設定されます。現在、Oracle XML Gateway では XML のみがサポートされています。
MESSAGE_STANDARD	「XML 標準フォームの定義」に入力されて「取引フォームの定義」に表示されるメッセージ形式の標準。これは、OAG にデフォルト設定されます。インバウンド XML 文書に入力されたメッセージ標準は、取引先の設定のメッセージ標準と同じである必要があります。
TRANSACTION_TYPE	取引先の表からのビジネス文書の外部取引タイプ。インバウンド XML 文書の取引タイプは、取引先フォームで定義された取引タイプと同じである必要があります。
TRANSACTION_SUBTYPE	取引先の表からのビジネス文書の外部取引サブタイプ。インバウンド XML 文書の取引サブタイプは、取引先フォームで定義された取引サブタイプと同じである必要があります。
DOCUMENT_NUMBER	発注書や請求書番号などの、取引を識別するために使用される文書識別子。このパラメータは、XML ゲートウェイでは使用されませんが、インバウンド・メッセージで渡すことができます。
PARTY_SITE_ID	インバウンド XML 文書のパーティ・サイト識別子は、取引先フォームで定義されたソース取引先の場所と同じである必要があります。

SOAP メッセージの例

Oracle SOA Suite から受信する SOAP 要求メッセージおよび応答メッセージについての理解を深めるために、ここでは次のサンプルの SOAP メッセージについて説明します。

同期 SOAP メッセージの詳細は、次を参照してください。

- サンプルの同期 SOAP 要求、ページ 2-39
- サンプルの同期 SOAP 応答、ページ 2-41
- サンプルの障害同期 SOAP 応答、ページ 2-41

非同期 SOAP メッセージの詳細は、次を参照してください。

- サンプルの非同期 SOAP 要求、ページ 2-41
- サンプルの非同期 SOAP 応答、ページ 2-43
- サンプルの障害非同期 SOAP 応答、ページ 2-44

サンプルの同期 SOAP 要求

次の例は、PL/SQL サービスに対する同期 SOAP 要求を示しています。

```

<soapenv:Envelope xmlns:ser="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"
xmlns:ozf="http://xmlns.oracle.
com/apps/ozf/soapprovider/plsql/ozf_sd_request_pub/"
xmlns:cre="http://xmlns.oracle.
com/apps/ozf/soapprovider/plsql/ozf_sd_request_pub/create_sd_request/">
  <soapenv:Header>
    <wsse:Security soapenv:mustUnderstand="1">
      <wsse:UsernameToken>
        <wsse:Username>trademgr</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-username-token-profile-1.0#PasswordText">password</wsse:
Password>
      </wsse:UsernameToken>
    </wsse:Security>
    <ozf:SOAHeader>
      <ozf:Responsibility>OZF_USER</ozf:Responsibility>
      <ozf:RespApplication>OZF</ozf:RespApplication>
      <ozf:SecurityGroupE>STANDARD</ozf:SecurityGroup>
      <ozf:NLSLanguage>AMERICAN</ozf:NLSLanguage>
      <ozf:Org_Id>204</ozf:Org_Id>
    </ozf:SOAHeader>
  </soapenv:Header>
  <soapenv:Body>
    <cre:InputParameters>
      <cre:P_API_VERSION_NUMBER>1.0</cre:P_API_VERSION_NUMBER>
      <cre:P_INIT_MSG_LIST>T</cre:P_INIT_MSG_LIST>
      <cre:P_COMMIT>F</cre:P_COMMIT>
      <cre:P_VALIDATION_LEVEL>100</cre:P_VALIDATION_LEVEL>
      <cre:P_SDR_HDR_REC>
        <cre:REQUEST_NUMBER>SDR-CREATE-A1</cre:REQUEST_NUMBER>
        <cre:REQUEST_START_DATE>2008-08-18T12:00:00</cre:
REQUEST_START_DATE>
        <cre:REQUEST_END_DATE>2008-10-18T12:00:00</cre:REQUEST_END_DATE>>
        <cre:USER_STATUS_ID>1701</cre:USER_STATUS_ID>
        <cre:REQUEST_OUTCOME>IN PROGRESS</cre:REQUEST_OUTCOME>
        <cre:REQUEST_CURRENCY_CODE>USD</cre:REQUEST_CURRENCY_CODE>
        <cre:SUPPLIER_ID>601</cre:SUPPLIER_ID>
        <cre:SUPPLIER_SITE_ID>1415</cre:SUPPLIER_SITE_ID>
        <cre:REQUESTOR_ID>100001499</cre:REQUESTOR_ID>
        <cre:ASSIGNEE_RESOURCE_ID>100001499</cre:ASSIGNEE_RESOURCE_ID>
        <cre:ORG_ID>204</cre:ORG_ID>
        <cre:ACCRUAL_TYPE>SUPPLIER</cre:ACCRUAL_TYPE>
        <cre:REQUEST_DESCRIPTION>Create</cre:REQUEST_DESCRIPTION>
        <cre:SUPPLIER_CONTACT_EMAIL_ADDRESS>sdr.supplier@example.
com</cre:SUPPLIER_CONTACT_EMAIL_ADDRESS>
        <cre:SUPPLIER_CONTACT_PHONE_NUMBER>2255</cre:
SUPPLIER_CONTACT_PHONE_NUMBER>
        <cre:REQUEST_TYPE_SETUP_ID>400</cre:REQUEST_TYPE_SETUP_ID>
        <cre:REQUEST_BASIS>Y</cre:REQUEST_BASIS>
        <cre:USER_ID>1002795</cre:USER_ID>
      </cre:P_SDR_HDR_REC>
      <cre:P_SDR_LINES_TBL>
        <cre:P_SDR_LINES_TBL_ITEM>
          <cre:PRODUCT_CONTEXT>PRODUCT</cre:PRODUCT_CONTEXT>
          ...
        </cre:P_SDR_LINES_TBL_ITEM>
      </cre:P_SDR_LINES_TBL>
      <cre:P_SDR_CUST_TBL>
        ...
      </cre:P_SDR_CUST_TBL>
    </cre:InputParameters>
  </soapenv:Body>
</soapenv:Envelope>

```

サンプルの同期 SOAP 応答

次の例は、PL/SQL サービスの同期 SOAP 応答を示しています。

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header/>
  <env:Body>
    <OutputParameters xmlns="http://xmlns.oracle.com/apps/ozf/soapprovider/plsql/ozf_sd_request_pub/create_sd_request/">
      <X_RETURN_STATUS>S</X_RETURN_STATUS>
      <X_MSG_COUNT>23</X_MSG_COUNT>
      <X_MSG_DATA xsi:nil="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
      <X_REQUEST_HEADER_ID>162</X_REQUEST_HEADER_ID>
    </OutputParameters>
  </env:Body>
</env:Envelope>
```

サンプルの障害同期 SOAP 応答

SOAP Fault 要素は、SOAP メッセージでエラーとステータスの情報を転送するために使用されます。

たとえば、次の障害同期応答メッセージは、サービスが配置されていないことを示しています。

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header/>
  <env:Body>
    <Fault xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
      <faultcode xmlns="">SOAP-ENV:Server</faultcode>
      <faultstring xmlns="">Service is not deployed.</faultstring>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

サンプルの非同期 SOAP 要求

次の例は、PL/SQL サービス OZF_SD_REQUEST_PUB に含まれている CREATE_SD_REQUEST_ASYNCH 操作に対する非同期 SOAP 要求を示しています。

```

<soapenv:Envelope "http://xmlns.oracle.
com/isg/ozf_sd_request_pub/CREATE_SD_REQUEST_ASYNC" xmlns:crel="http:
//xmlns.oracle.
com/apps/ozf/soapprovider/plsql/ozf_sd_request_pub/create_sd_request/"
xmlns:ozf="http://xmlns.oracle.
com/apps/ozf/soapprovider/plsql/ozf_sd_request_pub/" xmlns:soapenv="http:
//schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.
oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <wsse:UsernameToken wsu:Id="UsernameToken-3" xmlns:wsu="http://docs.
oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
        <wsse:Username>trademgr</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-username-token-profile-1.0#PasswordText">password</wsse:
Password>
        <wsse:Nonce EncodingType="http://docs.oasis-open.
org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
>JtcpwUGEcUyy09YgwaPzSA==</wsse:Nonce>
        <wsu:Created>2011-09-21T08:17:10.656Z</wsu:Created>
      </wsse:UsernameToken>
    </wsse:Security>
    <ozf:SOAHeader>
      <!--Optional:-->
      <ozf:Responsibility>OZF_USER</ozf:Responsibility>
      <!--Optional:-->
      <ozf:RespApplication>OZF</ozf:RespApplication>
      <!--Optional:-->
      <ozf:SecurityGroupE>STANDARD</ozf:SecurityGroup>
      <!--Optional:-->
      <ozf:NLSLanguage>AMERICAN</ozf:NLSLanguage>
      <!--Optional:-->
      <ozf:Org_Id>204</ozf:Org_Id>
    </ozf:SOAHeader>
  </soapenv:Header>
  <soapenv:Body>
    <cre:InputParameters>
      <cre:P_API_VERSION_NUMBER>1.0</cre:P_API_VERSION_NUMBER>
      <cre:P_INIT_MSG_LIST>T</cre:P_INIT_MSG_LIST>
      <cre:P_COMMIT>F</cre:P_COMMIT>
      <cre:P_VALIDATION_LEVEL>100</cre:P_VALIDATION_LEVEL>
      <cre:P_SDR_HDR_REC>
        <crel:REQUEST_NUMBER>SDR-CREATE-BPEL001</crel:REQUEST_NUMBER>
        <crel:REQUEST_START_DATE>2011-08-18T12:00:00</crel:
REQUEST_START_DATE>
        <crel:REQUEST_END_DATE>2012-10-18T12:00:00</crel:
REQUEST_END_DATE>>
        <crel:USER_STATUS_ID>1712</crel:USER_STATUS_ID>
        <crel:REQUEST_OUTCOME>IN_PROGRESS</crel:REQUEST_OUTCOME>
        <crel:REQUEST_CURRENCY_CODE>USD</crel:REQUEST_CURRENCY_CODE>
        <crel:SUPPLIER_ID>1718</crel:SUPPLIER_ID>
        <crel:SUPPLIER_SITE_ID>2854</crel:SUPPLIER_SITE_ID>
        <crel:REQUESTOR_ID>100001499</crel:REQUESTOR_ID>
        <crel:ASSIGNEE_RESOURCE_ID>100001499</crel:ASSIGNEE_RESOURCE_ID>
        <crel:ORG_ID>204</crel:ORG_ID>
        <crel:ACCRUAL_TYPE>SUPPLIER</crel:ACCRUAL_TYPE>
        <crel:REQUEST_DESCRIPTION>Create</crel:REQUEST_DESCRIPTION>

        <crel:SUPPLIER_CONTACT_EMAIL_ADDRESS>sdr.supplier@example.
com</crel:SUPPLIER_CONTACT_EMAIL_ADDRESS>
        <crel:SUPPLIER_CONTACT_PHONE_NUMBER>2255</crel:
SUPPLIER_CONTACT_PHONE_NUMBER>
        <crel:REQUEST_TYPE_SETUP_ID>400</crel:REQUEST_TYPE_SETUP_ID>
        <crel:REQUEST_BASIS>Y</crel:REQUEST_BASIS>
        <crel:USER_ID>1002795</crel:USER_ID>
      </cre:P_SDR_HDR_REC>
    </cre:InputParameters>
  </soapenv:Body>
</soapenv:Envelope>

```

```

<cre:P_SDR_LINES_TBL>
  <cre1:P_SDR_LINES_TBL_ITEM>
    <cre1:PRODUCT_CONTEXT>PRODUCT</cre:PRODUCT_CONTEXT>
    <cre1:INVENTORY_ITEM_ID>149</cre1:INVENTORY_ITEM_ID>
    <cre1:ITEM_UOM>Ea</cre1:ITEM_UOM>
    <cre1:REQUESTED_DISCOUNT_TYPE>%</cre1:REQUESTED_DISCOUNT_TYPE>
    <cre1:REQUESTED_DISCOUNT_VALUE>15.5</cre1:REQUESTED_DISCOUNT_VALUE>
    <!--<cre1:COST_BASIS>200</cre1:COST_BASIS>-->
    <cre1:MAX_QTY>200</cre1:MAX_QTY>
    <cre1:DESIGN_WIN>200</cre1:DESIGN_WIN>
    <cre1:APPROVED_DISCOUNT_TYPE>%</cre1:APPROVED_DISCOUNT_TYPE>
    <cre1:APPROVED_DISCOUNT_VALUE>15.5</cre1:APPROVED_DISCOUNT_VALUE>
    <cre1:APPROVED_MAX_QTY>200</cre1:APPROVED_MAX_QTY>
    <cre1:VENDOR_APPROVED_FLAG>Y</cre1:VENDOR_APPROVED_FLAG>
    <cre1:PRODUCT_COST_CURRENCY>USD</cre1:PRODUCT_COST_CURRENCY>
    <cre1:END_CUSTOMER_CURRENCY>USD</cre1:END_CUSTOMER_CURRENCY>
  </cre1:P_SDR_LINES_TBL_ITEM>
</cre:P_SDR_LINES_TBL>
<cre:P_SDR_CUST_TBL>
  <cre1:P_SDR_CUST_TBL_ITEM>
    <cre1:CUST_ACCOUNT_ID>1290</cre1:CUST_ACCOUNT_ID>
    <cre1:PARTY_ID>1290</cre1:PARTY_ID>
    <cre1:SITE_USE_ID>10479</cre1:SITE_USE_ID>
    <cre1:CUST_USAGE_CODE>BILL_TO</cre1:CUST_USAGE_CODE>
    <cre1:END_CUSTOMER_FLAG>N</cre1:END_CUSTOMER_FLAG>
  </cre1:P_SDR_CUST_TBL_ITEM>
  <cre1:P_SDR_CUST_TBL_ITEM>
    <cre1:CUST_ACCOUNT_ID>1287</cre1:CUST_ACCOUNT_ID>
    <cre1:PARTY_ID>1287</cre1:PARTY_ID>
    <cre1:SITE_USE_ID>1418</cre1:SITE_USE_ID>
    <cre1:CUST_USAGE_CODE>CUSTOMER</cre1:CUST_USAGE_CODE>
    <cre1:END_CUSTOMER_FLAG>Y</cre1:END_CUSTOMER_FLAG>
  </cre1:P_SDR_CUST_TBL_ITEM>
</cre:P_SDR_CUST_TBL>
</cre:InputParameters>
</soapenv:Body>
</soapenv:Envelope>

```

サンプルの非同期 SOAP 応答

次の例は、PL/SQL サービス OZF_SD_REQUEST_PUB に含まれている
CREATE_SD_REQUEST_ASYNC 操作に対する非同期応答を示しています。

```

<?xml version="1.0" encoding="UTF-8" ?>
<outputParameters
  xmlns:client="http://xmlns.oracle.
com/isg/ozf_sd_request_pub/CREATE_SD_REQUEST_ASYNC"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns="http://xmlns.oracle.
com/isg/ozf_sd_request_pub/CREATE_SD_REQUEST_ASYNC">
  <OutputParameters xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns="http://xmlns.oracle.
com/apps/ozf/soapprovider/plsql/ozf_sd_request_pub/create_sd_request/">
    <X_RETURN_STATUS>S</X_RETURN_STATUS>
    <X_MSG_COUNT>23</X_MSG_COUNT>
    <X_MSG_DATA xsi:nil="true"/>
    <X_REQUEST_HEADER_ID>31</X_REQUEST_HEADER_ID>
  </OutputParameters>
</outputParameters>

```

サンプルの障害非同期 SOAP 応答

たとえば、次のサンプルは、ヘッダーに誤りがある soapUI からの非同期応答メッセージを示しています。

```
<?xml version="1.0" encoding="UTF-8" ?>
<outputParameters
  xmlns:client="http://xmlns.oracle.
com/isg/ozf_sd_request_pub/CREATE_SD_REQUEST_ASYNC"
  xmlns:wsa="http://www.w3.org/2005/08/addressing"
  xmlns="http://xmlns.oracle.
com/isg/ozf_sd_request_pub/CREATE_SD_REQUEST_ASYNC">
  <OutputParameters
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://xmlns.oracle.
com/apps/ozf/soapprovider/plsql/ozf_sd_request_pub/create_sd_request/">
    <X_RETURN_STATUS>E</X_RETURN_STATUS>
    <X_MSG_COUNT>1</X_MSG_COUNT>
    <X_MSG_DATA>The Organization Id provided is invalid, please
provide a valid Organization Id.</X_MSG_DATA>
    <X_REQUEST_HEADER_ID xsi:nil="true"/>
  </OutputParameters>
</outputParameters>
```

REST メッセージについて

REST アーキテクチャに基づいて、REST メッセージは HTTP ヘッダーおよび POST メソッドを使用して、サービス・プロバイダ経由で Oracle E-Business Suite データを作成または更新します。

XML および JSON メッセージ形式のサポート

完全に XML 形式ベースの SOAP メッセージとは異なり、REST メッセージは XML 形式と JSON などの非 XML 形式の両方を処理できます。

注意: このリリースでは Jackson JSON 形式のみがサポートされます。Google GSON などのその他の JSON 形式はサポートされません。

- **XML** は、HTML と同様に、山カッコのタグ・ペア(<または>)をネストして情報を編成します。
- **JSON** は XML よりも軽量であり、解析結果が得られるまでの時間が短縮されます。REST サービスでよく使用される簡単なテキストベースのメッセージ形式です。

これは、中カッコ({または})を使用して、情報を階層的に構造化します。

REST メッセージの構造

REST 要求は、次の要素が含まれている簡単な HTTP 要求です。

- ヘッダー

この要素は、HTTP トランザクションの操作パラメータを定義します。

REST サービスのユーザー資格証明を HTTP ヘッダーで渡すことができます。REST サービス・セキュリティの詳細は、「REST セキュリティ・ヘッダー」(ページ 2-45)を参照してください。

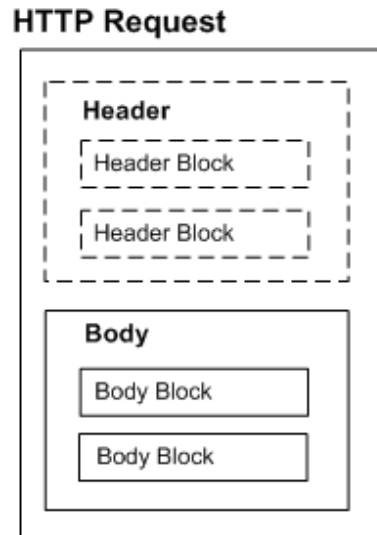
- 本文

この要素は、メイン・メッセージまたはメイン・リソースを定義します。

REST サービスの起動にアプリケーション・コンテキスト値が必要な場合には、**RESTHeader** 要素を HTTP 本文に含めてその値を設定できます。

アプリケーション・コンテキストの設定の詳細は、「アプリケーション・コンテキストの REST ヘッダー」(ページ 2-46)を参照してください。

次の図は、REST メッセージの構造を示しています。



REST セキュリティ・ヘッダー

次のいずれかのメソッドに基づいて、ユーザー資格証明を認証する必要があります。

- **HTTP Basic 認証**

このセキュリティ・モデルでは、**ユーザー名とパスワード**を REST 要求メッセージの一部として HTTP ヘッダーに入力データとして指定する必要があります。REST サービスが要求を受信すると、ユーザー資格証明(ユーザー名とパスワード)が認証および認可のために **LoginModule** に転送されます。続いて、**LoginModule** が HTTP ヘッダーから資格証明を抽出し、Oracle E-Business Suite ユーザー表に照らしてユーザーを認証し、認証したユーザーのアイデンティティを確立します。

ユーザー資格証明を検証し、REST サービスの起動に必要なアプリケーション・コンテキストを初期化できたら、REST サービスを起動できます。

HTTP Basic 認証セキュリティの詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「HTTP Basic 認証」を参照してください。

- **トークン・ベース認証**

ユーザーに関連付けられているパスワードを渡すのではなく、**セキュリティ・トークン**をパスワードのかわりにユーザー資格証明として渡すことができます。

複数の要求でユーザーがサーバーにログオンしようとする場合、その都度ユーザー名とパスワードでユーザーを認証するのではなく、HTTP ヘッダーでユーザー名とともに一意のアクセス・トークン(Oracle E-Business Suite セッション ID など)を送信できます。

Oracle E-Business Suite セッション ID は、ログイン・サービスをコールすることによって取得できます。LoginModule は HTTP ヘッダーを解釈してトークンを抽出し、後続の要求では、トークンを使用してサブジェクトまたはユーザー名を認証のために検証します。

ユーザー資格証明を検証し、REST サービスの起動に必要なアプリケーション・コンテキストを初期化できたら、サービスを起動できます。

アプリケーション・コンテキストの設定の詳細は、「アプリケーション・コンテキストの REST ヘッダー」(ページ 2-46)を参照してください。

トークン・ベースのセキュリティの詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「トークン・ベース認証」を参照してください。

アプリケーション・コンテキストの REST ヘッダー

一部の Oracle E-Business Suite API は、アプリケーション・コンテキスト値を渡さないと起動できません。このようなコンテキスト値には Responsibility、RespApplication、SecurityGroup、NLSLanguage、Org_Id などがあり、いずれも HTTP 本文の一部として RESTHeader 要素に含めることができます。

トークン・ベースのセキュリティのオプションのコンテキスト値

コンテキスト・ヘッダー値はオプションです。トークン・ベースのセキュリティを使用している場合にコンテキスト値を渡さなかったときには、以前に渡した値が使用されます。コンテキスト値を渡すと、指定したトークンに以前に設定した値が新しく渡した値で上書きされます。

次の XML 形式の REST メッセージでは、RESTHeader 要素を太字で出力しています。

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<TESTUSERNAME_Input xmlns="http://xmlns.oracle.
com/apps/fnd/rest/FndUserSvc/testusername/">
  <RESTHeader xmlns="http://xmlns.oracle.
com/apps/fnd/rest/FndUserSvc/header">
    <Responsibility>SYSTEM_ADMINISTRATOR</Responsibility>
    <RespApplication></RespApplication>
    <SecurityGroup></SecurityGroup>
    <NLSLanguage>AMERICAN</NLSLanguage>
    <Org_Id>/Org_Id>
  </RESTHeader>
  <InputParameters>
    <X_USER_NAME>sysadmin</X_USER_NAME>
  </InputParameters>
</TESTUSERNAME_Input>
```

オプションの言語パラメータ

<NLSLanguage>パラメータで言語作業環境を設定するときと同様に、Web サービス・クライアントが Oracle E-Business Suite で使用されている<NLSLanguage>パラメータを認識し

ない場合には、かわりに ISO 言語パラメータ<Language>を使用して、RFC 5646 形式の言語作業環境を REST ヘッダーに含めて送信してセッション言語を設定できます。

注意: <Language>パラメータは、REST サービスでのみ使用します。

たとえば、標準の RFC 5646 形式のドイツ語は de-DE または de です。次のように <Language>パラメータで言語をドイツ語に設定します。

<Language>de-DE</Language>または<Language>de</Language>

<NLSLanguage>パラメータと<Language>パラメータは、どちらもオプションであることに注意してください。<NLSLanguage>パラメータが<Language>パラメータよりも優先される場合があります、それは<NLSLanguage>パラメータ値 (<NLSLanguage>GERMAN</NLSLanguage>など)が渡されて有効になっている場合です。

- <NLSLanguage>値を指定しない場合は、次のようになります。
 - <Language>パラメータを有効な値とともに渡した場合は、<Language>値が使用されます。
 - <Language>パラメータを無効な値とともに渡した場合は、エラー・メッセージが返されます。

たとえば、de-DE を<Language>値として渡したものの、ドイツ語がまだ有効でないかインストールされていない Oracle E-Business Suite 言語である場合は、言語がインストールされていないことを示すエラーが表示されます。

- <NLSLanguage>値が無効であるかインストールされていない場合は、エラー・メッセージが返されます。

<NLSLanguage>値と<Language>値の両方を指定していないときにのみ、HTTP ヘッダーにある Accept-Language ヘッダー(Accept-Language = de-DE など)が考慮されます。Accept-Language ヘッダーを渡していないか、または渡した値が無効であるかインストールされていない場合は、ユーザーのデフォルト言語が使用されます。

WADL 記述からのペイロードの構築

WADL 記述のリソース情報に基づいて、REST サービスを起動する前に入力ペイロードをコンパイルできます。

次のステップを使用して、入力ペイロードをコンパイルします。

1. 「統合リポジトリ」で、使用する配置済 REST サービスを検索して見つけます。
2. 「REST Web サービス」タブの「**WADL の表示**」リンクをクリックします。次の WADL 記述が表示されます。

```

<xml version="1.0" encoding="UTF-8" standalone="no" ?>
<application xmlns:tns="http://xmlns.oracle.
com/apps/fnd/soapprovider/plsql/rest/fnd_user_pkg/" xmlns="http:
//wadl.dev.java.net/2009/02" xmlns:tns1="http://xmlns.oracle.
com/apps/fnd/rest/FndUserSvc/testusername/" name="FND_USER_PKG"
targetNamespace="http://xmlns.oracle.
com/apps/fnd/soapprovider/plsql/rest/fnd_user_pkg/">
<grammars>
  <include xmlns="http://www.w3.org/2001/XMLSchema" href="https:
//<hostname>:<port>/webservices/rest/FndUserSvc/?
XSD=TESTUSERNAME_SYNCH_TYPEDEF.xsd" />
</grammars><resources base="http://<hostname>:
<port>/webservices/rest/FndUserSvc/"><resource
path="/testusername/" >
  <method id="GET" name="POST">
    <request>
      <representation mediaType="application/xml" type="tns1:
InputParameters" />
      <representation mediaType="application/json" type="tns1:
InputParameters" />
    </request>
    <response>
      <representation mediaType="application/xml" type="tns1:
OutputParameters" />
      <representation mediaType="application/json" type="tns1:
OutputParameters" />
    </response>
  </method>
</resource>
</resources>
</application>

```

3. WADL 記述からテスト・ユーザー名(TESTUSERNAME)サービス操作のスキーマ情報 (.XSD)を探します。WADL での操作 TESTUSERNAME の XSD は、次のようになります。

```

http://<hostname>:<port>/webservices/rest/FndUserSvc/?
XSD=TESTUSERNAME_SYNCH_TYPEDEF.xsd

```

注意: サービス操作のスキーマ情報は、WADL 記述にある次の要素の値を連結することによっても構築できます。

- <resources base="http://<hostname>:<port>/webservices/rest/FndUserSvc/">
- <resource path="/testusername/">

4. XSD を XML に変換する任意のツールを使用してペイロード情報を取得することによって、サービスのペイロードを構築します。

ペイロードをコンパイルしたら、それを使用して TESTUSERNAME REST サービス操作を起動できます。次の表に要求、応答、障害の各メッセージを XML 形式と JSON 形式の両方でリストします。

XML と JSON の形式の REST メッセージ

入力ペイロード XML ベースの REST メッセージ

(要求メッセージ)

```
<?xml version="1.0" encoding="UTF-8" ?>
<TESTUSERNAME_Input xmlns="http://xmlns.oracle.
com/apps/fnd/rest/FndUserSvc/testusername/">
  <RESTHeader xmlns="http://xmlns.oracle.
com/apps/fnd/rest/FndUserSvc/header">
    <Responsibility>SYSTEM_ADMINISTRATOR</Responsibility>
    <RespApplication></RespApplication>
    <SecurityGroupE></SecurityGroup>
    <NLSLanguage></NLSLanguage>
    <Org_Id></Org_Id>
  </RESTHeader>
  <InputParameters>
    <X_USER_NAME>sysadmin</X_USER_NAME>
  </InputParameters>
</TESTUSERNAME_Input>
```

JSON ベースの REST メッセージ

```
{ "TESTUSERNAME_Input": {
  "@xmlns": "http://xmlns.oracle.
com/apps/fnd/rest/FndUserSvc/testusername/",
  "RESTHeader": {
    "@xmlns": "http://xmlns.oracle.
com/apps/fnd/rest/FndUserSvc/header",
    "Responsibility": "SYSTEM_ADMINISTRATOR",
    "RespApplication": "SYSADMIN",
    "SecurityGroup": "STANDARD",
    "NLSLanguage": "AMERICAN",
    "Org_Id": "202"
  },
  "InputParameters": {
    "X_USER_NAME": "operations"
  }
}}
```

応答

XML ベースの REST メッセージ

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<OutputParameters xmlns:xsi="http://www.w3.
org/2001/XMLSchema-instance"
xmlns="http://xmlns.oracle.
com/apps/fnd/rest/FndUserSvc/testusername/">
  <X_USER_NAME>2</X_USER_NAME>
</OutputParameters>
```

JSON ベースの REST メッセージ

```
{
  "OutputParameters" : {
    "@xmlns:xsi" : "http://www.w3.org/2001/XMLSchema-
instance",
    "@xmlns" : "http://xmlns.oracle.
com/apps/fnd/rest/fndGlobalSvc/user_id/",
    "TESTUSERNAME" : "2"
  }
}
```

XML と JSON の形式の REST メッセージ

エラー応答 XML ベースの REST メッセージ

```
<ISGServiceFault>
  <Code>IRepAccessError</Code>
  <Message>This is a sample Fault Message.Message will
vary depending on fault condition</Message>
  <Resolution>Check the server logs for
details</Resolution>
  <ServiceDetails>
    <ServiceName>FndUserSvc</ServiceName>
    <OperationName>testusername</OperationName>
    <InstanceId>0</InstanceId>
  </ServiceDetails>
</ISGServiceFault>
```

JSON ベースの REST メッセージ

```
{
  "ISGServiceFault": {
    "Code": "IRepAccessError",
    "Message": "Sample Fault Message.Will vary depending
on fault condition",
    "Resolution": "Check the server logs for details",
    "ServiceDetails": {
      "ServiceName": "FndUserSvc",
      "OperationName": "testusername",
      "InstanceId": "0"
    }
  }
}
```

OZF_SD_REQUEST_PUB サービス起動に使用されている REST メッセージのその他の例の詳細は、「REST メッセージの例」(ページ 2-50)を参照してください。

REST メッセージの例

Oracle E-Business Suite から受信する REST 要求メッセージおよび応答メッセージについての理解を深めるために、ここでは次のサンプルの REST メッセージについて説明します。

- サンプルの XML ベースの REST 要求、ページ 2-50
- サンプルの JSON ベースの REST 要求、ページ 2-55
- サンプルの XML ベースの REST 応答、ページ 2-57
- サンプルの JSON ベースの REST 応答、ページ 2-59
- XML ベースの障害応答のサンプル、ページ 2-59

サンプルの XML ベースの REST 要求

この項では、PL/SQL サービスを使用して出荷およびデビット要求を作成するためのサンプルの REST 要求と、オープン・インタフェース表に対してデータを更新、挿入または削除するための要求について説明します。

- **PL/SQL サービスの同期要求**

次の例は、PL/SQL サービス(OZF_SD_REQUEST_PUB API)に対する同期 XML ベースの REST 要求を示しています。

```

<?xml version="1.0" encoding="UTF-8" ?>
<CREATE_SD_REQUEST_Input xmlns:xsi="http://www.w3.
org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.oracle.
com/apps/ozf/rest/ozfsdrequestpubsvc/create_sd_request/xsd/OZF_SD_
RE_QUEST_PUB_CREATEREQUEST.xsd"
    xmlns="http://xmlns.oracle.
com/apps/ozf/rest/ozfsdrequestpubsvc/create_sd_request/">
  <RESTHeader xmlns="http://xmlns.oracle.
com/apps/fnd/rest/ozfsdrequestpubsvc/header">
    <Responsibility>SYSTEM ADMINISTRATOR</Responsibility>
    <RespApplication></RespApplication>
    <SecurityGroupE></SecurityGroup>
    <NLSLanguage></NLSLanguage>
    <Org_Id></Org_Id>
  </RESTHeader>
  <InputParameters>
    <P_API_VERSION_NUMBER>1.0</P_API_VERSION_NUMBER>
    <P_INIT_MSG_LIST>T</P_INIT_MSG_LIST>
    <P_COMMIT>F</P_COMMIT>
    <P_VALIDATION_LEVEL>100</P_VALIDATION_LEVEL>
    <P_SDR_HDR_REC>
      <REQUEST_NUMBER>SDR-CREATE-BPEL1</REQUEST_NUMBER>
      <REQUEST_START_DATE>2008-08-18T12:00:00</REQUEST_START_DATE>
      <REQUEST_END_DATE>2008-10-18T12:00:00</REQUEST_END_DATE>
      <USER_STATUS_ID>1701</USER_STATUS_ID>
      <REQUEST_OUTCOME>IN_PROGRESS</REQUEST_OUTCOME>
      <REQUEST_CURRENCY_CODE>USD</REQUEST_CURRENCY_CODE>
      <SUPPLIER_ID>601</SUPPLIER_ID>
      <SUPPLIER_SITE_ID>1415</SUPPLIER_SITE_ID>
      <REQUESTOR_ID>100001499</REQUESTOR_ID>
      <ASSIGNEE_RESOURCE_ID>100001499</ASSIGNEE_RESOURCE_ID>
      <ORG_ID>204</ORG_ID>
      <ACCRUAL_TYPE>SUPPLIER</ACCRUAL_TYPE>
      <REQUEST_DESCRIPTION>Create</REQUEST_DESCRIPTION>
      <SUPPLIER_CONTACT_EMAIL_ADDRESS>sdr.supplier@example.
com</SUPPLIER_CONTACT_EMAIL_ADDRESS>

      <SUPPLIER_CONTACT_PHONE_NUMBER>2255</SUPPLIER_CONTACT_PHONE_NUMBER>
      <REQUEST_TYPE_SETUP_ID>400</REQUEST_TYPE_SETUP_ID>
      <REQUEST_BASIS>Y</REQUEST_BASIS>
      <USER_ID>1002795</USER_ID>
    </P_SDR_HDR_REC>
    <P_SDR_LINES_TBL>
      <P_SDR_LINES_TBL_ITEM>
        <PRODUCT_CONTEXT>PRODUCT</PRODUCT_CONTEXT>
        <INVENTORY_ITEM_ID>2155</INVENTORY_ITEM_ID>
        <ITEM_UOM>Ea</ITEM_UOM>
        <REQUESTED_DISCOUNT_TYPE>%</REQUESTED_DISCOUNT_TYPE>
        <REQUESTED_DISCOUNT_VALUE>20</REQUESTED_DISCOUNT_VALUE>
        <COST_BASIS>200</COST_BASIS>
        <MAX_QTY>200</MAX_QTY>
        <DESIGN_WIN>200</DESIGN_WIN>
        <APPROVED_DISCOUNT_TYPE>%</APPROVED_DISCOUNT_TYPE>
        <APPROVED_DISCOUNT_VALUE>20</APPROVED_DISCOUNT_VALUE>
        <APPROVED_MAX_QTY>200</APPROVED_MAX_QTY>
        <VENDOR_APPROVED_FLAG>Y</VENDOR_APPROVED_FLAG>
        <PRODUCT_COST_CURRENCY>USD</PRODUCT_COST_CURRENCY>
        <END_CUSTOMER_CURRENCY>USD</END_CUSTOMER_CURRENCY>
      </P_SDR_LINES_TBL_ITEM>
    </P_SDR_LINES_TBL>
    <P_SDR_CUST_TBL>
      <P_SDR_CUST_TBL_ITEM>
        <CUST_ACCOUNT_ID>1290</CUST_ACCOUNT_ID>
        <PARTY_ID>1290</PARTY_ID>
      </P_SDR_CUST_TBL_ITEM>
    </P_SDR_CUST_TBL>
  </InputParameters>
</CREATE_SD_REQUEST_Input>

```

```

<SITE_USE_ID>10479</SITE_USE_ID>
  <CUST_USAGE_CODE>BILL_TO</CUST_USAGE_CODE>
  <END_CUSTOMER_FLAG>N</END_CUSTOMER_FLAG>
</P_SDR_CUST_TBL_ITEM>
<P_SDR_CUST_TBL_ITEM>
  <CUST_ACCOUNT_ID>1287</CUST_ACCOUNT_ID>
  <PARTY_ID>1287</PARTY_ID>
  <SITE_USE_ID>1418</SITE_USE_ID>
  <CUST_USAGE_CODE>CUSTOMER</CUST_USAGE_CODE>
  <END_CUSTOMER_FLAG>Y</END_CUSTOMER_FLAG>
</P_SDR_CUST_TBL_ITEM>
</P_SDR_CUST_TBL>
</InputParameters>
</CREATE_SD_REQUEST_Input>

```

- オープン・インタフェース表のレコードを更新するための要求

次の例は、インバウンド方向の AR Autoinvoice オープン・インタフェース(RAXMTR)に含まれているオープン・インタフェース表 RA_INTERFACE_LINES_ALL を更新(PUT HTTP メソッド)するための要求メッセージを示しています。

この例では、HTTP 要求は複数の更新要求で構成されています。各更新要求の応答メッセージには、ステータス、メッセージおよび更新レコード数を含める必要があります。この要求には Select も含まれています。このため、応答メッセージによって、Select 文の指定した列 INTERFACE_LINE_ID および BATCH_SOURCE_NAME に値が返されます。

```

PUT http://host:
port/webservices/rest/autoinvoice/RA_INTERFACE_LINES_ALL
Content-Type: application/xml

```

```

<?xml version="1.0" encoding="UTF-8" ?>
<RA_INTERFACE_LINES_ALL_Input xmlns="http://xmlns.oracle.
com/apps/ar/rest/autoinvoice/RA_INTERFACE_LINES_ALL">
  <RESTHeader xmlns="http://xmlns.oracle.com/apps/fnd/rest/header">
    <Responsibility>RECEIVABLES_VISION_OPERATIONS</Responsibility>
    <RespApplication>AR</RespApplication>
    <SecurityGroupE>STANDARD</SecurityGroup>
    <NLSLanguage>AMERICAN</NLSLanguage>
    <Org_Id>204</Org_Id>
  </RESTHeader><Select>INTERFACE_LINE_ID,
BATCH_SOURCE_NAME</Select>
  <InputParameters>
    <Update>
      <Filter>BATCH_SOURCE_NAME==ICS-01;ERROR_FLAG==T;
QUANTITY=le=100</Filter>
      <RA_INTERFACE_LINES_ALL_REC>
        <REQUEST_ID></REQUEST_ID>
        <ERROR_FLAG />
        <PROCESS_FLAG>0</PROCESS_FLAG>
      </Update>
      <Update>
        <Filter>BATCH_SOURCE_NAME==ICS-01;ERROR_FLAG==T;
QUANTITY=gt=100</Filter>
        <RA_INTERFACE_LINES_ALL_REC>
          <PROCESS_FLAG>4</PROCESS_FLAG>
        </RA_INTERFACE_LINES_ALL_REC>
      </Update>
    </InputParameters>
  </RA_INTERFACE_LINES_ALL_Input>

```

- オープン・インタフェース表にレコードを挿入するための要求

この例では、インバウンド方向の同じ AR Autoinvoice (RAXMTR)に含まれている同じオープン・インタフェース表 RA_INTERFACE_LINES_ALL にデータを挿入(POST HTTP メソッド)するためのサンプルの要求について説明します。

この要求は、Select パラメータで次のように選択した列名 INTERFACE_LINE_ID および BATCH_SOURCE_NAME に挿入する 2 つのレコードで構成されています。前述した update サンプル要求と同様に、応答メッセージによって、同じ選択した列の値が返されます。

```
PUT http://host:
port/webservices/rest/autoinvoice/RA_INTERFACE_LINES_ALL
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" ?>
<RA_INTERFACE_LINES_ALL_Input xmlns="http://xmlns.oracle.
com/apps/ar/rest/autoinvoice/RA_INTERFACE_LINES_ALL">
  <RESTHeader xmlns="http://xmlns.oracle.com/apps/fnd/rest/header">
    <Responsibility>RECEIVABLES_VISION_OPERATIONS</Responsibility>
    <RespApplication>AR</RespApplication>
    <SecurityGroupE>STANDARD</SecurityGroup>
    <NLSLanguage>AMERICAN</NLSLanguage>
    <Org_Id>204</Org_Id>
  </RESTHeader><Select>INTERFACE_LINE_ID,
BATCH_SOURCE_NAME</Select>
  <InputParameters>
    <RA_INTERFACE_LINES_ALL_REC>
      <REQUEST_ID></REQUEST_ID>
      <BATCH_SOURCE_NAME>ICS-1</BATCH_SOURCE_NAME>
      <RELATED_TRX_NUMBER>101</RELATED_TRX_NUMBER>
      <INTERFACE_LINE_ID>11</INTERFACE_LINE_ID>
      ...
    </RA_INTERFACE_LINES_ALL_REC>
    <RA_INTERFACE_LINES_ALL_REC>
      <REQUEST_ID></REQUEST_ID>
      <BATCH_SOURCE_NAME>ICS-1</BATCH_SOURCE_NAME>
      <RELATED_TRX_NUMBER>102</RELATED_TRX_NUMBER>
      <INTERFACE_LINE_ID>12</INTERFACE_LINE_ID>
      ...
    </RA_INTERFACE_LINES_ALL_REC>
  </InputParameters>
</RA_INTERFACE_LINES_ALL_Input>
```

- オープン・インタフェース表のレコードを削除するための要求

削除操作(DELETE HTTP メソッド)に対するこのサンプル要求では、AR Autoinvoice (RAXMTR)に含まれている同じオープン・インタフェース表 RA_INTERFACE_LINES_ALL を使用します。この操作では、フィルタ基準を問合せパラメータとして受け入れる必要があります。

たとえば、次の要求では、レコードが条件 ORG_ID = 204、TRX_DATE < 27/04/2016 および PROCESS_FLAG = 4 を満たした場合に、RA_INTERFACE_LINES_ALL のレコードが削除されます。

問合せは、Feed Item Query Language (FIQL)または Resource Query Language (RQL)に基づいていることに注意してください。

```
http://host:
port/webservices/rest/autoinvoice/RA_INTERFACE_LINES_ALL? filter
=ORG_ID==204;TRX_DATE=lt=2016-04-27T00:00:00.000%2B00:00;
PROCESS_FLAG==4
```


応答メッセージには、ステータス、メッセージおよび削除するレコード数 (DeleteCount)が含まれています。

サンプルの JSON ベースの REST 要求

次の例は、同じ PL/SQL サービス(OZF_SD_REQUEST_PUB API)に対する同期 JSON ベースの POST 要求を示しています。

注意: このリリースでは Jackson JSON 形式のみがサポートされます。Google GSON などのその他の JSON 形式はサポートされません。

```

{
  "CREATE_SD_REQUEST_Input": {
    "@xmlns": "http://xmlns.oracle.
com/apps/ozf/rest/ozfsdrequestpubsvc/create_sd_request/",
    "RESTHeader": {
      "@xmlns": "http://xmlns.oracle.
com/apps/fnd/rest/ozfsdrequestpubsvc/header",
      "Responsibility": "SYSTEM_ADMINISTRATOR"
    },
    "InputParameters": {
      "P_API_VERSION_NUMBER": "1.0",
      "P_INIT_MSG_LIST": "T",
      "P_COMMIT": "F",
      "P_VALIDATION_LEVEL": "100",
      "P_SDR_HDR_REC": {
        "REQUEST_NUMBER": "SDR-CREATE-BPEL1",
        "REQUEST_START_DATE": "2008-08-18T12:00:00", "REQUEST_END_DATE":
        "2008-10-18T12:00:00",
        "USER_STATUS_ID": "1701",
        "REQUEST_OUTCOME": "IN_PROGRESS",
        "REQUEST_CURRENCY_CODE": "USD",
        "SUPPLIER_ID": "601",
        "SUPPLIER_SITE_ID": "1415",
        "REQUESTOR_ID": "100001499",
        "ASSIGNEE_RESOURCE_ID": "100001499",
        "ORG_ID": "204",
        "ACCRUAL_TYPE": "SUPPLIER",
        "REQUEST_DESCRIPTION": "Create",
        "SUPPLIER_CONTACT_EMAIL_ADDRESS": "sdr.supplier@example.com",
        "SUPPLIER_CONTACT_PHONE_NUMBER": "2255",
        "REQUEST_TYPE_SETUP_ID": "400",
        "REQUEST_BASIS": "Y",
        "USER_ID": "1002795"
      },
      "P_SDR_LINES_TBL": {
        "P_SDR_LINES_TBL_ITEM": {
          "PRODUCT_CONTEXT": "PRODUCT",
          "INVENTORY_ITEM_ID": "2155",
          "ITEM_UOM": "Ea",
          "REQUESTED_DISCOUNT_TYPE": "%",
          "REQUESTED_DISCOUNT_VALUE": "20",
          "COST_BASIS": "200",
          "MAX_QTY": "200",
          "DESIGN_WIN": "200",
          "APPROVED_DISCOUNT_TYPE": "%",
          "APPROVED_DISCOUNT_VALUE": "20",
          "APPROVED_MAX_QTY": "200",
          "VENDOR_APPROVED_FLAG": "Y",
          "PRODUCT_COST_CURRENCY": "USD",
          "END_CUSTOMER_CURRENCY": "USD"
        }
      },
      "P_SDR_CUST_TBL": {
        "P_SDR_CUST_TBL_ITEM": [
          {
            "CUST_ACCOUNT_ID": "1290",
            "PARTY_ID": "1290",
            "SITE_USE_ID": "10479",
            "CUST_USAGE_CODE": "BILL_TO",
            "END_CUSTOMER_FLAG": "N"
          },
          {
            "CUST_ACCOUNT_ID": "1287",
            "PARTY_ID": "1287",
            "SITE_USE_ID": "1418",

```

次の例は、REST Service Locator という Java Bean サービスに対する JSON ベースの GET 要求を示しています。

注意: GET 要求では、JSON がデフォルトの出力応答形式です。Accept ヘッダー `application/xml` を使用して、XML 形式の応答を受け入れます。GET HTTP 要求で `Content-Type` ヘッダーを送信した場合、それは無視されます。

次の例は、OZF_SD_REQUEST_PUB API サービスに対する XML ベースの REST 応答を示しています。

また、次の例は、オープン・インタフェース表 AR Autoinvoice に対する前述の HTTP 要求への応答メッセージを示しています。RA_INTERFACE_LINES_ALL 表の INTERFACE_LINE_ID 列および BATCH_SOURCE_NAME 列への**更新**要求に応じて、応答メッセージには索引で識別された各更新要求のステータス、メッセージおよび更新レコード数(<UpdateCount>)が要求の発生順に含まれています。更新が正常に完了すると、この応答メッセージでも、要求の Select 文に示された INTERFACE_LINE_ID および BATCH_SOURCE_NAME の値が返されます。

```

<?xml version = '1.0' encoding = 'UTF-8'?>
<OutputParameters xmlns="http://xmlns.oracle.
com/apps/ar/rest/autoinvoice/RA_INTERFACE_LINES_ALL">
  <Summary>
    <SuccessCount>5</SuccessCount>
    <ErrorCount>1</ErrorCount>
  </Summary>
  <Result><Index>1</Index> <Status>SUCCESS</Status>
  <Message></Message>
  <UpdateCount>2</UpdateCount>
  <RA_INTERFACE_LINES_ALL_REC>
    <INTERFACE_LINE_ID>11</INTERFACE_LINE_ID>
    <BATCH_SOURCE_NAME>ICS-1</BATCH_SOURCE_NAME>
  </RA_INTERFACE_LINES_ALL_REC>
  <RA_INTERFACE_LINES_ALL_REC>
    <INTERFACE_LINE_ID>12</INTERFACE_LINE_ID>
    <BATCH_SOURCE_NAME>ICS-1</BATCH_SOURCE_NAME>
  </RA_INTERFACE_LINES_ALL_REC>
</Output>
<Output><Index>2</Index><Status>ERROR</Status>
<Message>Invalid date ..</Message>
<UpdateCount>0</UpdateCount>
  <RA_INTERFACE_LINES_ALL_REC>
</Output>
</Result>
</OutputParameters>

```

同様に、前述の同じオープン・インタフェース表 RA_INTERFACE_LINES_ALL に 2 つのレコードを挿入するという要求の場合、次に示すように、応答メッセージによって同じ選択した列 INTERFACE_LINE_ID および BATCH_SOURCE_NAME の値が返されます。

```

<?xml version = '1.0' encoding = 'UTF-8'?>
<OutputParameters xmlns="http://xmlns.oracle.
com/apps/ar/rest/autoinvoice/RA_INTERFACE_LINES_ALL">
  <Summary>
    <SuccessCount>15</SuccessCount>
    <ErrorCount>2</ErrorCount>
  </Summary>
  <Result>
    <Output><Index>1</Index>
      <Status>SUCCESS</Status>
      <Message></Message>
      <RA_INTERFACE_LINES_ALL_REC>
        <INTERFACE_LINE_ID>11</INTERFACE_LINE_ID>
        <BATCH_SOURCE_NAME>ICS-1</BATCH_SOURCE_NAME>
      </RA_INTERFACE_LINES_ALL_REC>
    </Output>
    <Output><Index>2</Index>
      <Status>ERROR</Status>
      <Message>Invalid date ..</Message>
      <RA_INTERFACE_LINES_ALL_REC>
        <INTERFACE_LINE_ID>12</INTERFACE_LINE_ID>
        <BATCH_SOURCE_NAME>ICS-1</BATCH_SOURCE_NAME>
      </RA_INTERFACE_LINES_ALL_REC>
    </Output>
  </Result>
</OutputParameters>

```

前述の削除要求に応じて、応答メッセージにはステータス、メッセージおよび削除レコード数(DeleteCount)が含まれています。

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<OutputParameters xmlns="http://xmlns.oracle.
com/apps/ar/rest/autoinvoice/RA_INTERFACE_LINES_ALL">
  <Result>
    <Status>SUCCESS</Status>
    <Message></Message>
    <DeleteCount>2</DeleteCount>
  </Result>
</OutputParameters>
```

サンプルの JSON ベースの REST 応答

次の例は、POST メソッドを使用した OZF_SD_REQUEST_PUB API サービスに対する JSON ベースの REST 応答を示しています。

```
{
  "OutputParameters" : {
    "@xmlns:xsi" : "http://www.w3.org/2001/XMLSchema-instance",
    "@xmlns" : "http://xmlns.oracle.
com/apps/ozf/rest/ozfsdrequestpubsvc/create_sd_request/",
    "X_RETURN_STATUS" : "E",
    "X_MSG_COUNT" : "1",
    "X_MSG_DATA" : "The Organization Id provided is invalid, please
provide a valid Organization Id.",
    "X_REQUEST_HEADER_ID" : {
      "@xsi:nil" : "true"
    }
  }
}
```

次の例は、GET メソッドを使用した Rest Service Locator (getRestInterface サービス操作) サービスに対する JSON ベースの REST 応答を示しています。

```
{
  "OutputParameters" : {
    "EbsRestServiceBean" : [ {
      "alternateAlias" : "plssql/PLSQL:FND_PROFILE",
      "serviceAlias" : "NotAnything",
      "serviceName" : "PLSQL:FND_PROFILE",
      "wadlUrl" : "http://<hostname>:
<port>/webservices/rest/profile?WADL"
    } ],
    "ControlBean" : [ {
      "fields" : "",
      "filter" : "",
      "limit" : "",
      "offset" : ""
    } ]
  }
}
```

XML ベースの障害応答のサンプル

次のサンプルは、XML の形式が正しくない場合の XML ベースの REST 応答メッセージを示しています。

```

<ISGServiceFault>
  <Code>RequestParsingError</Code>
  <Message>SAXException in XmlRequestObject, while parsing XML request
The request could not be parsed correctly</Message>
  <Resolution>This may be due to malformed construction of the payload
or incorrectContent-Type header.Please check the wellformed-ness of
payload, matching Content-Type header of the http request and retry.
</Resolution>
  <ServiceDetails>
    <ServiceName>ozfsdrequestpubsvc</ServiceName>
    <OperationName>create_sd_request</OperationName>
    <InstanceId>0</InstanceId>
  </ServiceDetails>
</ISGServiceFault>

```

次のサンプルは、XML ベースの REST 応答メッセージであり、RespApplication(職責アプリケーション(Responsibility Application)の短縮名)が無効である場合の例を示しています。

```

<ISGServiceFault>
  <Code>InvalidResponsibilityApplicationShortCode</Code>
  <Message>Responsibility short code is invalid System error while
processing the request</Message>
  <Resolution>Check the server logs for details</Resolution>
  <ServiceDetails>
    <ServiceName>ozfsdrequestpubsvc</ServiceName>
    <OperationName>get_text_number</OperationName>
    <InstanceId>0</InstanceId>
  </ServiceDetails>
</ISGServiceFault>

```

WebサービスとしてのPL/SQL APIの使用

概要

Oracle E-Business Suite Integrated SOA Gateway を使用すると、PL/SQL アプリケーション・プログラミング・インタフェース(API)を使用して Oracle E-Business Suite でデータを挿入または更新できます。API は、Oracle E-Business Suite のデータを更新または取得できるストア・プロセスです。

PL/SQL API インタフェース定義が、WSDL で表現される SOAP Web サービスとして公開された後、配置済サービスをサービス・エンドポイントを持つ SOA コンポジット・アプリケーション内の有用な BPEL プロセスにオーケストレーションできます。実行時に、soa-infra アプリケーションが実行されている WebLogic 管理対象サーバーの SOA コンポジットを顧客に公開し、Web サービス・クライアントまたはオーケストレーション・ツール(Oracle JDeveloper、Apache Axis、.NET Web サービス・クライアント、Oracle BPEL Process Manager、Oracle Enterprise Service Bus (ESB)など)によって起動できます。

SOAP サービスに加えて、PL/SQL API は REST サービスとして公開できます。アプリケーション・データの挿入または更新で各 Web サービスを使用する方法についての理解を深めるために、この章では、設計時タスクおよび実行時タスクの詳細について説明します。以降の項で説明する例では、BPEL プロセスによる SOA コンポジット・アプリケーションを作成するための設計時ツールとして Oracle JDeveloper 11g (11.1.1.6.0)を使用し、プロセスの配置には Oracle SOA Suite 11g (11.1.1.6.0)を使用します。

注意: Oracle JDeveloper とその他の Oracle Fusion Middleware コンポーネント (Oracle SOA Suite など)を使用している場合、SOA テクノロジを有効にするには、SOA テクノロジ用の JDeveloper の拡張機能である Oracle SOA Suite コンポジット・エディタを手動でダウンロードする必要があります。追加の Oracle Fusion Middleware 設計時コンポーネントのインストールの詳細は、*Oracle Fusion Middleware Oracle JDeveloper インストレーション・ガイド*を参照してください。

この章では、次のトピックを取り上げます。

- PL/SQL SOAP サービスの使用、ページ 3-2

- BPEL プロセスによる SOA コンポジット・アプリケーションからの同期 SOAP Web サービスの起動、ページ 3-6
- BPEL プロセスによる SOA コンポジット・アプリケーションからの非同期 SOAP Web サービスの起動、ページ 3-46
- PL/SQL REST サービスの使用、ページ 3-72
 - HTTP Basic 認証および REST ヘッダー付き XML ペイロードを使用した REST サービスの起動、ページ 3-73
 - トークン・ベース認証および JSON ペイロードを使用した REST サービスの起動、ページ 3-83

PL/SQL SOAP サービスの使用

BPEL プロセスによる SOA コンポジット・アプリケーションのシナリオ

PL/SQL の「仕入先出荷およびデビット要求」API `OZF_SD_REQUEST_PUB` を例として、BPEL プロセスによる SOA コンポジット・アプリケーションの作成について説明します。

出荷およびデビット要求の作成が受信されると、入力された出荷およびデビット・ペイロードを含む作成情報が読み取られて、出荷およびデビット要求を作成するために渡されます。

• 同期 SOAP Web サービスの起動

同期サービスは、必要なすべての情報が即時に提供される場合に要求を処理するために使用されます。要求番号が要求者に同期的に戻されます。

BPEL プロセスによる SOA コンポジット・アプリケーションが配置後に正常に実行されると、Oracle Order Management で出荷およびデビット要求が作成されます。要求番号は、ペイロード入力値と同じである必要があります。

参照: BPEL プロセスによる SOA コンポジット・アプリケーションからの同期 SOAP Web サービスの起動、ページ 3-6。

• 非同期 SOAP Web サービスの起動

応答メッセージが要求者に即時に返されない場合、または要求の処理に長時間かかる場合は、かわりに非同期 Web サービスを使用できます。要求が完了したときに、応答情報が受信されます。

BPEL プロセスによる SOA コンポジット・アプリケーションが作成されます。非同期操作の応答が入力になる BPEL プロセス。この BPEL プロセスにより、受信された応答は SOA コンポジットが配置されているサーバー上の出力ファイルに書き込まれます。

参照: BPEL プロセスによる SOA コンポジット・アプリケーションからの非同期 SOAP Web サービスの起動、ページ 3-46。

PL/SQL Web サービスを使用して BPEL プロセスによる SOA コンポジット・アプリケーションを作成するための前提条件

PL/SQL Web サービスの設計時タスクを実行する前に、次のタスクを実行しておく必要があります。

注意: 選択したインタフェースの Web サービスを生成する前に、必要に応じて特定のユーザー(TRADEMGR など)またはユーザー・グループ用にセキュリティ付与を作成し、ユーザーがそのインタフェースへのアクセス権限を持つようにすることができます。

- 統合管理者または統合開発者が、最初に Web サービスを生成する必要があります。管理者が、Oracle SOA Suite WebLogic 管理対象サーバーに、生成されたサービスをデプロイします。
- 統合開発者が、Web サービスとして公開される PL/SQL インタフェースのデプロイ済 WSDL URL を特定して記録する必要があります。
- Web サービス承認用に SOAHeader 変数を設定する必要があります。

Oracle E-Business Suite Integrated SOA Gateway から公開される特定の PL/SQL API は、レコード・タイプを入力として取得することに注意してください。そのような API は、正常な実行のために、これらのレコード・タイプ内のパラメータにデフォルト値が移入されると想定しています。

デフォルト値は、FND_API.G_MISS_CHAR (文字の場合)、FND_API.G_MISS_DATE (日付の場合)、FND_API.G_MISS_NUM (数値の場合)です。Oracle E-Business Suite Integrated SOA Gateway では、たとえば次の例に示すように、レコード・タイプ内のパラメータが nil 値として渡された場合に、これらの値をデフォルト設定できます。

```
<PRICE_LIST_REC>
<ATTRIBUTE1 xsi:nil="true"/>
<ATTRIBUTE2 xsi:nil="true"/>
<ATTRIBUTE3 xsi:nil="true"/>
...
</PRICE_LIST_REC>
```

PL/SQL Web サービス・コンポジットのデプロイ

統合管理者または統合開発者は、最初に、選択したインタフェース定義の Web サービスを作成する必要があり、その後、管理者は Oracle Integration Repository から Oracle SOA Suite WebLogic 管理対象サーバーにサービスをデプロイできます。

たとえば、統合開発者がデプロイ済 WSDL を使用して BPEL を作成する前に、まず次のステップを実行する必要があります。

1. Web サービスを生成するには、統合管理者または統合開発者は最初にインタフェース定義を特定し(PL/SQL インタフェース OZF_SD_REQUEST_PUB など)、必要な相互作用パターン情報(同期または非同期、あるいは両方のパターン)を相互作用パターン表から選択します。

これは、インタフェース・レベルまたはメソッド・レベルで、インタフェースの詳細ページの「生成」をクリックする前に選択できます。

サービスが正常に生成されると、「SOAP Web サービス」タブで「SOAP サービス・ステータス」フィールドが「未生成」から「生成済」に変わります。Web サービスの生成方法の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「SOAP Web サービスの生成」を参照してください。

2. 生成された Web サービスを配置するには、「配置」をクリックする前に、管理者が 1 つの認証タイプを選択します。Oracle SOA Suite に配置されたサービスは、有効なサービスであり、新しい SOAP 要求を受け入れる準備が整っています。

サービスが正常に配置されたら、「SOAP サービス・ステータス」フィールドが「有効」の状態となり、「配置済」とともに、選択した認証タイプが表示されます。認証タイプによる Web サービスの保護の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「Web サービス・セキュリティの管理」を参照してください。

Web サービスの配置方法の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「SOAP Web サービスの配置および配置解除」を参照してください。

WSDL URL の検索および記録

前述の必須タスク以外に、統合開発者は、Oracle JDeveloper で有用な BPEL プロセスにオーケストレーションする必要があるインタフェースの配置済 Web サービスの WSDL URL を特定して記録する必要があります。

これを行うには、インタフェースの詳細ページの「WSDL の表示」リンクをクリックします。新しいポップアップ・ウィンドウから WSDL URL をコピーします。この URL は、後で BPEL プロセスでパートナ・リンク・サービスを作成するときに使用されます。

配置済 WSDL URL の表示および記録

Integration Repository > PLSQL Interface : OZF_SD_REQUEST Public API > PLSQL Method Details : Create SDR >

PLSQL Interface : OZF_SD_REQUEST Public API [Browse](#) [Search](#) [Printable Page](#)

Log Configuration Disabled

Internal Name: OZF_SD_REQUEST_PUB
Type: PL/SQL
Product: Trade Management
Status: Active
Business Entity: [Supplier Ship and Debit Request](#)

Scope: Public
Interface Source: Oracle

Overview **SOAP Web Service** REST Web Service

SOAP Service Status: Deployed | Active | [View WSDL](#)

Service Operations

Display Name	Internal Name	Synchronous	Asynchronous	Grant
▲ OZF_SD_REQUEST Public API	OZF_SD_REQUEST_PUB	☒	☐	
Create SDR	CREATE_SD_REQUEST	☒	☒	
Update SD Request	UPDATE_SD_REQUEST	☒	☐	
copy_sd_request	COPY_SD_REQUEST	☒	☐	

TIP To apply any changes in Interaction Pattern, Generate or Regenerate the service.

Web Service Security

* Authentication Type: ☒ Username Token ☐ SAML Token (Sender Vouches)

Copyright (c) 1998, 2015, Oracle and/or its affiliates. All rights reserved. [Privacy Statement](#)

インタフェースを検索してインタフェースの詳細を表示する方法の詳細は、「統合インタフェースの検索および表示」(ページ 2-1)を参照してください。

SOAP 要求の SOAHeader の変数の設定

サービスの実行中に、アプリケーション・コンテキストを設定するために要求される値を SOAHeader 要素で渡すには、特定の変数を設定する必要があります。PL/SQL インタフェース・タイプでのこれらの SOAHeader の要素は、*Responsibility*、*RespApplication*、*SecurityGroup*、*NLSLanguage* および *Org_Id* です。

注意: ユーザー名とパスワード情報は、Web サービス・セキュリティ・ポリシー (oracle/wss_username_token_service_policy など)によって定義されます。SOAP 要求とともにセキュリティ・ヘッダーを渡す方法の詳細な手順は、「Web サービス・ポリシーの構成」(ページ 3-34)を参照してください。

これらの要素の予期される値を次の表に示します。

PL/SQL インタフェース・タイプのヘッダー変数と予期される値

要素名	予期される値
Responsibility	responsibility_key (OZF_USER など)
RespApplication	アプリケーション短縮名(OZF など)
SecurityGroup	セキュリティ・グループ・キー(STANDARD など)
NLSLanguage	NLS 言語(AMERICAN など)
Org_Id	組織 ID(204 など)

注意: NLS 言語と組織 ID は、オプションで渡される値です。

- NLS 言語の要素が指定された場合、渡された言語で SOAP 要求を使用できます。すべての対応する SOAP 応答とエラー・メッセージも、同じ言語で戻すことができます。言語が指定されていない場合、ユーザーのデフォルトの言語が使用されます。
- サービスの実行が特定組織に依存する場合は、その SOAP 要求の Org_Id 要素を渡す必要があります。

BPEL プロセスで Invoke アクティビティの前に Assign アクティビティを構成することによって、コンテキスト情報を指定できます。

SOAP 要求の SOAHeader を設定する方法の詳細は、「SOAHeader パラメータの割当て」(ページ 3-23)を参照してください。

BPEL プロセスによる SOA コンポジット・アプリケーションからの同期 Web サービスの起動

単一請求書作成シナリオに基づき、必要なすべての情報が即時に提供される場合は、要求を処理するために同期 Web サービスが使用されます。要求番号が要求者に同期的に戻されます。

この章では、次の設計時タスクについて説明します。

1. 同期 BPEL プロセスによる SOA コンポジット・アプリケーションの作成、ページ 3-7 このステップを使用して、ShipDebitRequest.bpel という BPEL プロジェクトによる新しい SOA コンポジット・アプリケーションを「同期 BPEL プロセス」テンプレートを使用して作成します。

これにより、サード・パーティ・アプリケーションからの入力を受け入れる受入と、要求アプリケーションに BPEL プロセスの出力を返信する返信の 2 つのダミー・アクティビティが自動的に作成されます。

2. パートナ・リンクの作成、ページ 3-10

このステップを使用して、Web サービスとして公開される「仕入先出荷およびデビット要求」API OZF_SD_REQUEST_PUB を使用することによって、Oracle Order Management で出荷およびデビット要求を作成します。

3. ファイル・アダプタのパートナ・リンクの追加、ページ 3-11

このステップを使用して、最初の Assign アクティビティから渡される請求書ヘッダー詳細を同期的に読み込みます。

4. Invoke アクティビティの追加、ページ 3-19

このステップを使用して、次の目的の 2 つの Invoke アクティビティを構成します。

- ファイル・アダプタを指して、最初の Assign アクティビティから渡される請求書ヘッダー詳細を同期的に読み込みます。
- OZF_SD_REQUEST_PUB パートナ・リンクを指して、Assign アクティビティから受信したペイロードおよび取引詳細による要求作成を開始します。

5. Assign アクティビティの追加、ページ 3-23

このステップを使用して、要求ヘッダー詳細、ペイロード情報および要求番号を適切な Invoke アクティビティに渡して要求作成を促進するために、Assign アクティビティを構成します。最後に、ダミーの返信アクティビティを介して、要求アプリケーションに要求番号を渡します。

BPEL プロセスのサービス・コンポーネントを使用した SOA コンポジット・アプリケーションの一般情報と作成方法の詳細は、*Oracle Fusion Middleware Oracle SOA Suite 開発者ガイド* を参照してください。

BPEL プロセスによる SOA コンポジット・アプリケーションの作成

このステップを使用して、様々な BPEL プロセス・アクティビティによる新規 SOA コンポジット・アプリケーションを作成します。

BPEL プロジェクトを使用して新規 SOA コンポジット・アプリケーションを作成するには:

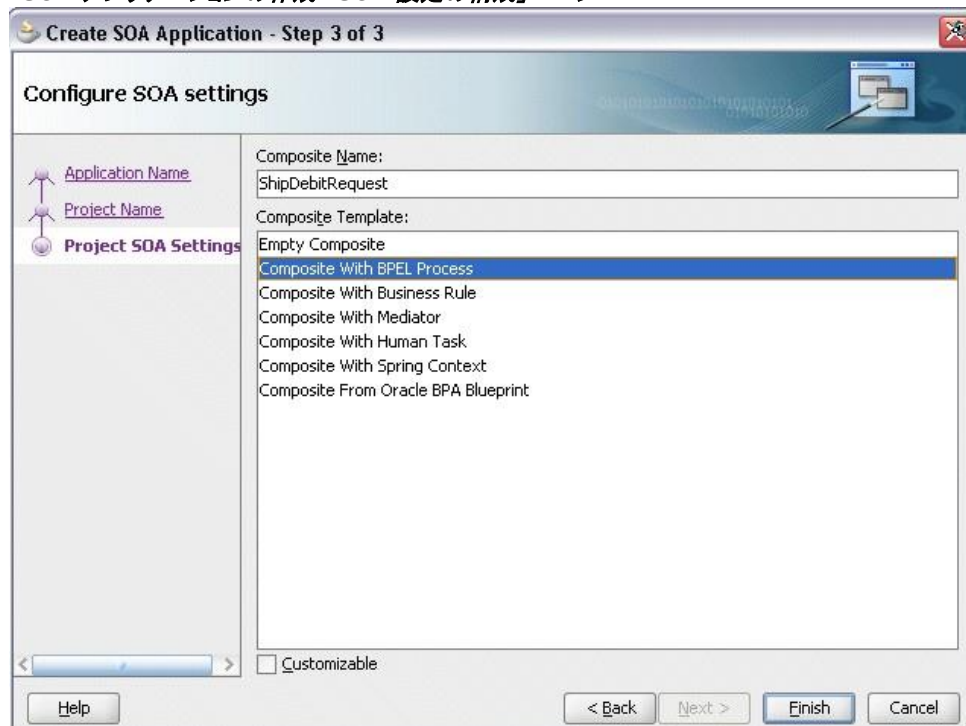
1. Oracle JDeveloper を開きます。
2. アプリケーション・ナビゲータで「**新規アプリケーション**」をクリックします。
「SOA アプリケーションの作成 - アプリケーションの名前付け」ページが表示されます。
3. 「アプリケーション名」フィールドにアプリケーションの適切な名前を入力して、「アプリケーション・テンプレート」リストから「**SOA アプリケーション**」を選択します。

「次へ」をクリックします。「SOA アプリケーションの作成 - プロジェクトの名前付け」ページが表示されます。

4. 「プロジェクト名」フィールドにプロジェクトの適切な名前を入力します (ShipDebitRequest など)。
5. 「プロジェクト・テクノロジー」タブで、「Web サービス」を選択して、「使用可能」のテクノロジー・リストから「選択済」のテクノロジー・リストに「SOA」が選択されるようにします。

「次へ」をクリックします。「SOA アプリケーションの作成 - SOA 設定の構成」ページが表示されます。

「SOA アプリケーションの作成 - SOA 設定の構成」ページ



6. 「コンポジット・テンプレート」リストから**コンポジット BPEL プロセス**を選択して、「終了」をクリックします。新規アプリケーションと SOA プロジェクトを作成しました。これにより、SOA コンポジットが自動的に作成されます。
「BPEL プロセスの作成」ページが表示されます。

「BPEL プロセスの作成」ページ

7. デフォルトの「**BPEL 1.1 Specification**」の選択を変更せずに維持します。これにより、BPEL 1.1 Specification をサポートする BPEL プロジェクトが作成されます。

「名前」フィールドに BPEL プロセスの適切な名前を入力します(ShipDebitRequest など)。

「テンプレート」フィールドで「同期 BPEL プロセス」を選択します。

「トランザクション」ドロップダウン・リストから「required」を選択します。「OK」をクリックします。

受入アクティビティと返信アクティビティのある同期 BPEL プロセスが作成されます。指定した名前を使用した、bpel および wsdl を含む必須のソース・ファイル(たとえば、ShipDebitRequest.bpel、ShipDebitRequest.wsdl)と composite.xml も生成されます。

注意: サービス・プロバイダでは、パラメータ・タイプ名に\$の文字がある PL/SQL ストアド・プロシージャまたはパッケージのサービス作成はサポートされません。名前に\$があると、XSD 生成に失敗します。

8. 「SOA コンテンツ」→「ビジネス・ルール」に移動し、composite.xml をクリックしてコンポジット・ダイアグラムを表示します。

ShipDebitRequest コンポーネントをダブルクリックして BPEL プロセスを開きます。

Web サービスのパートナ・リンクの作成

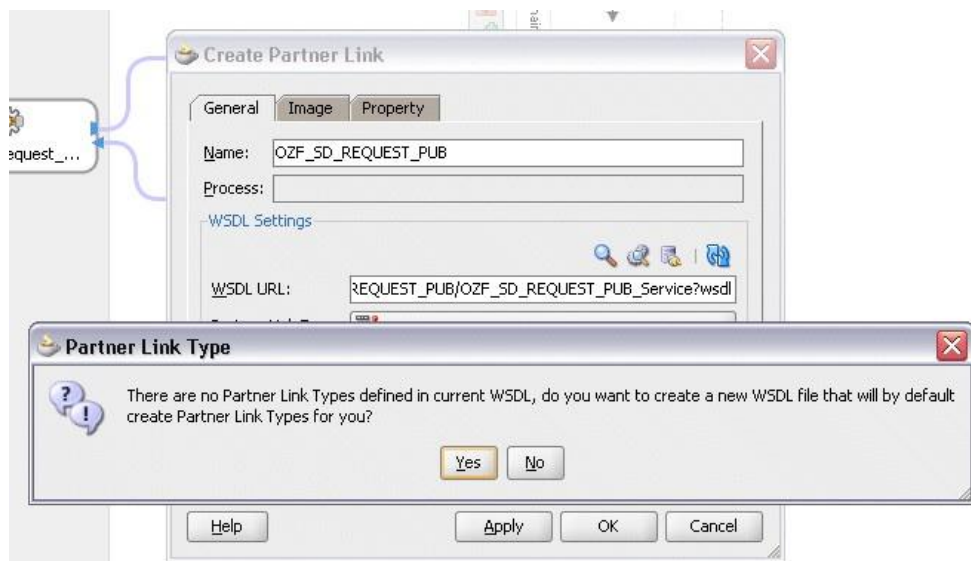
このステップを使用して、OZF_SD_REQUEST_PUB というパートナ・リンクを作成します。

OZF_SD_REQUEST_PUB Web サービスのパートナリンクを作成するには:

1. Oracle JDeveloper で、「パートナ・リンク」領域にマウスを置いて右クリックして、プルダウン・メニューから「パートナ・リンクの作成」を選択します。または、「BPEL コンストラクト」リストから「パートナ・リンク」を、プロセス・ダイアグラムの右側の「パートナ・リンク」スイムレーンにドラッグ・アンド・ドロップできます。

「パートナ・リンクの作成」ウィンドウが表示されます。

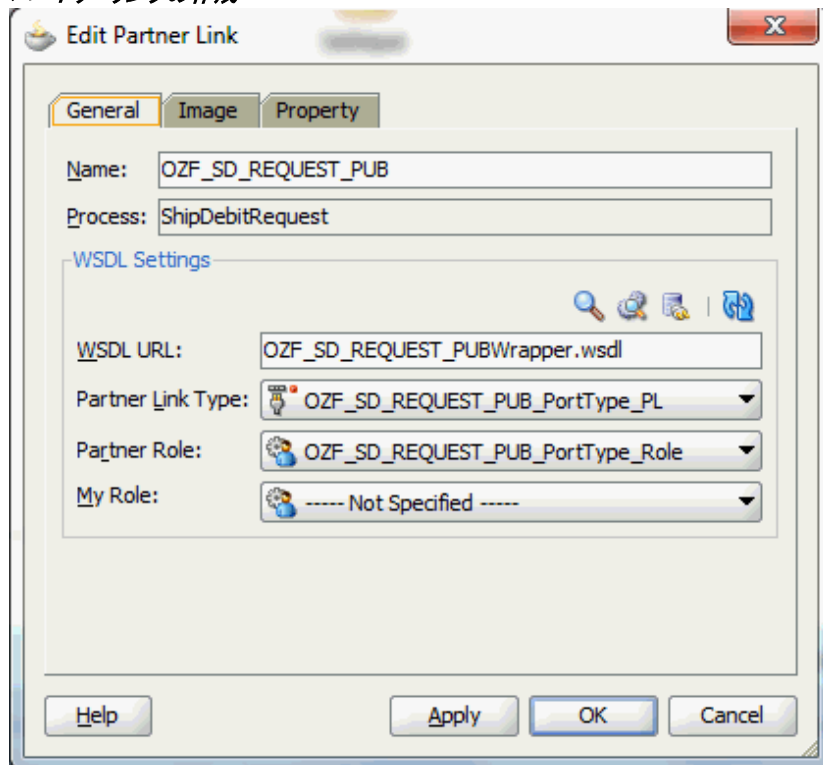
2. 以前に記録した OZF_SD_REQUEST_PUB サービスに対応する WSDL URL を統合リポジトリからコピーし、「WSDL ファイル」フィールドに貼り付けます。[Tab]キーを押します。
3. デフォルトでパートナ・リンク・タイプを作成する新しい WSDL ファイルをシステムで作成するかどうかを尋ねる「パートナ・リンク・タイプ」メッセージ・ダイアログ・ボックスが表示されます。



パートナ名の値が自動的に移入されるように、「はい」をクリックします。パートナリンク名を OZF_SD_REQUEST_PUB として入力します。

ドロップダウン・リストから「パートナ・リンク・タイプ」フィールドと「パートナ・ロール」フィールドを選択します。

パートナ・リンクの作成



「適用」をクリックします。

パートナ・リンクは、必要な WSDL 設定で作成され、BPEL プロジェクトでは、プロセス・ダイアグラムの枠線の領域の新しいアイコンによって表現されます。

4. 「OK」をクリックして、パートナ・リンクの構成を完了します。

パートナ・リンク OZF_SD_REQUEST_PUB が BPEL プロセス・ダイアグラムの「パートナ・リンク」セクションに追加されます。

ファイル・アダプタのパートナ・リンクの追加

このステップを使用して、ファイルの現在のコンテンツを読み込むことで BPEL プロセスを構成します。

ファイル・アダプタのパートナ・リンクをペイロードの読取りに追加するには:

1. Oracle JDeveloper で、「BPEL サービス」リストから「ファイル・アダプタ」サービスを、プロセス・ダイアグラムの右側の「パートナ・リンク」スイムレーンにドラッグ・アンド・ドロップします。ファイル・アダプタ構成ウィザードへようこそページが表示されます。
2. 「次へ」をクリックします。「サービス名」ダイアログ・ボックスが表示されます。
3. ファイル・アダプタ・サービス名を入力します(ReadPayload など)。

4. 「次へ」をクリックします。「アダプタ・インタフェース」ダイアログ・ボックスが表示されます。
5. 「操作およびスキーマから定義(後で指定)」ラジオ・ボタンを選択して、「次へ」をクリックします。「操作」ダイアログ・ボックスが表示されます。

操作の指定

Operation

The File Adapter supports four operations. There is a Read File operation that polls for incoming files in your local file system, a Write File operation that creates outgoing files, a Synchronous Read File operation that reads the current contents of a file, and a List Files operation that lists file names in specified locations. Specify the Operation type and Operation Name. Only one operation per Adapter Service may be defined using this wizard.

Operation Type: ☐ Read File ☐ Write File ☒ Synchronous Read File ☐ List Files

Operation Name:

Help < Back Next > Finish Cancel

6. 「同期 Read File」などの操作タイプを指定します。これにより、「操作名」フィールドが自動的に設定されます。
「次へ」をクリックして、「ファイル・ディレクトリ」ダイアログ・ボックスにアクセスします。

入力ファイル・ディレクトリの指定

Adapter Configuration Wizard - Step 5 of 8

File Directories

Enter directory information for the incoming file of the Synchronous Read File operation.

Directory names are specified as: ☒ Physical Path ☐ Logical Name

Directory for Incoming Files (physical path):

☐ Archive processed files
Archive Directory for Processed Files (physical path):

☒ Delete files after successful retrieval

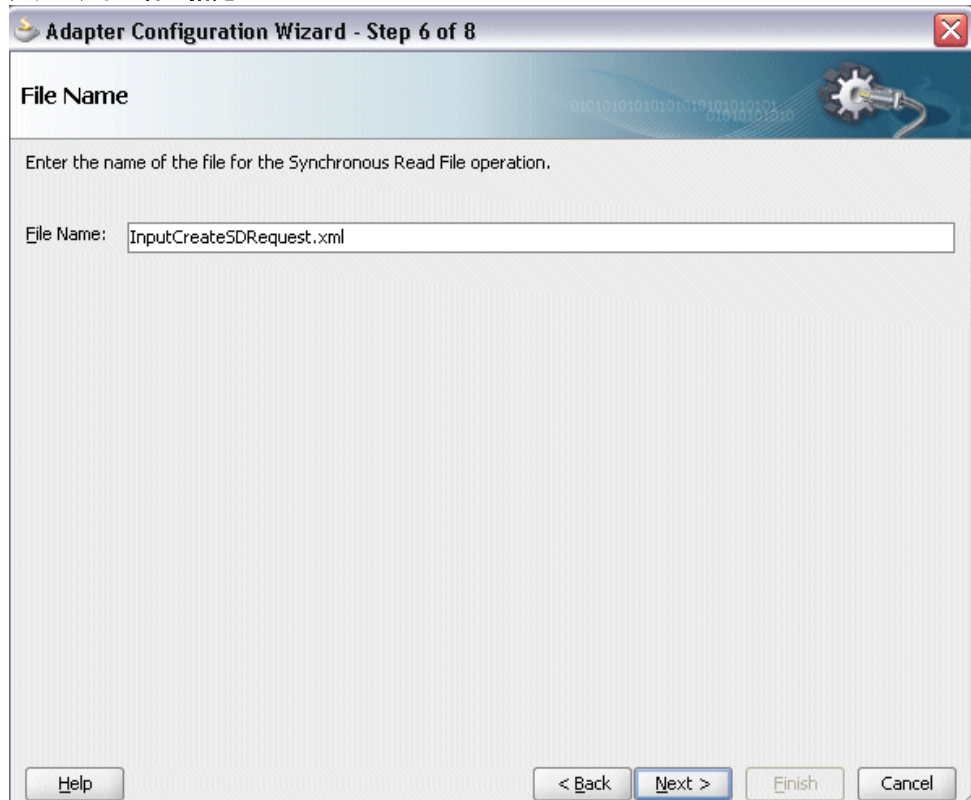
7. 「物理パス」ラジオ・ボタンを選択して、入力ペイロードのファイル・ディレクトリの情報を入力します。たとえば、ディレクトリ名として /usr/tmp/ と入力します。

注意: 入力ペイロード・ファイル `InputCreateSDRequest.xml` が Oracle SOA Suite サーバーのディレクトリ /usr/tmp/ フォルダで使用可能であることを確認する必要があります。

「次へ」をクリックして、「ファイル名」ダイアログ・ボックスを開きます。

8. 同期 Read File 操作のファイル名を入力します。たとえば、`InputCreateSDRequest.xml` と入力します。

入力ファイル名の指定



Adapter Configuration Wizard - Step 6 of 8

File Name

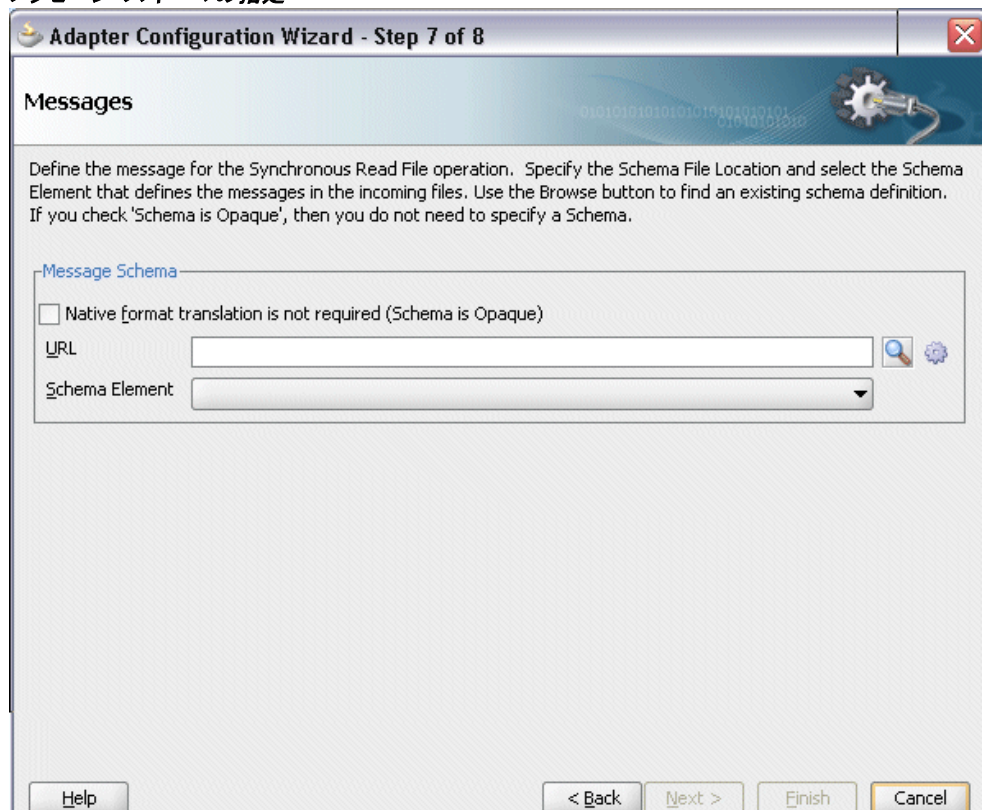
Enter the name of the file for the Synchronous Read File operation.

File Name:

Help < Back Next > Finish Cancel

「次へ」をクリックします。「メッセージ」ダイアログ・ボックスが表示されます。

メッセージ・スキーマの指定



9. 「URL」フィールドの右にある「スキーマ・ファイルを参照」を選択します。

「タイプ・チューザ」ウィンドウが表示されます。

「タイプ・チューザ」ウィンドウの右上隅にある「スキーマ・ファイルのインポート」ボタンをクリックします。

サービスのスキーマの場所を入力します。例: http:

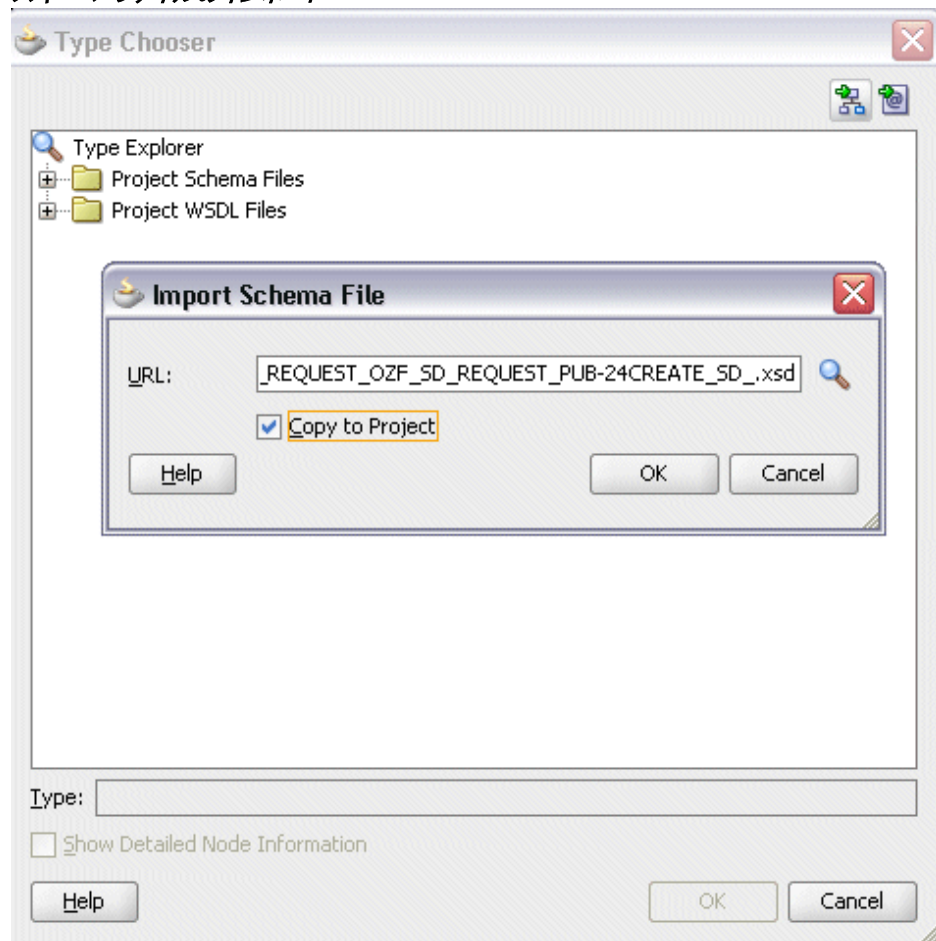
```
//<soa_suite_hostname>:<port>/soa-  
infra/services/default/<jndi_name>_PLSQL_OZF_SD_REQUEST_PUB  
/OZ_F_SD_REQUEST_PUB_Service?  
XSD=xsd/APPS_ISG_CREATE_SD_REQUEST_OZF_SD_REQUEST_PUB-  
24CREATE_SD_.xsd。
```

サービスのスキーマの場所は、サービスの WSDL URL (たとえば、

```
http://<soa_suite_hostname>:<port>/soa-  
infra/services/default/<jndi_name>_PLSQL_OZF_SD_REQUEST_PUB  
/OZ_F_SD_REQUEST_PUB_Service/?wsdl)から見つけることができます。
```

「プロジェクトにコピー」チェック・ボックスを選択して、「OK」をクリックします。

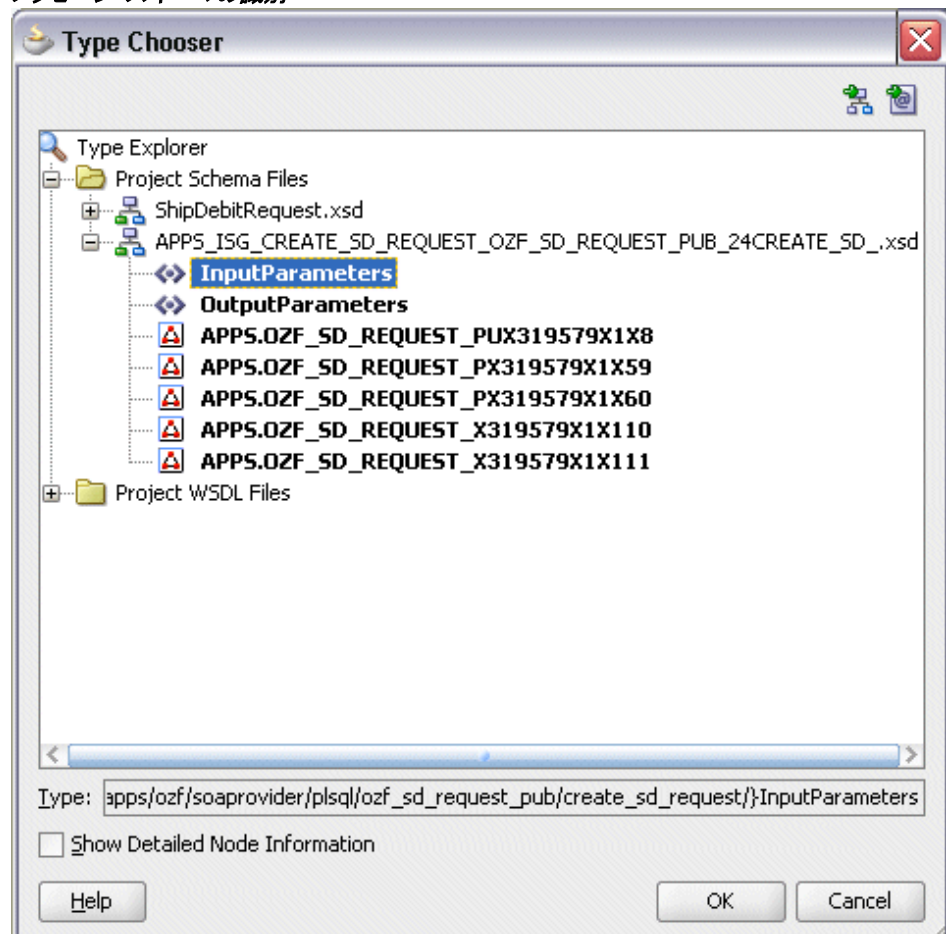
スキーマ・ファイルのインポート



「オプションをコピー」の「インポートされたファイルの元のディレクトリ構造を保持」チェック・ボックスを選択し、「OK」をクリックします。

「タイプ・チューザ」ウィンドウに、「インポートしたスキーマ」フォルダが自動的に追加されます。

メッセージスキーマの識別



APPS_ISG_CREATE_SD_REQUEST_OZF_SD_REQUEST_PUB-24CREATE_SD_.xsd の InputParameters メッセージを選択します。「OK」をクリックします。

選択した.xsd が URL として表示されて、InputParameters がスキーマ要素として選択されます。

選択したメッセージ・スキーマおよび要素の表示

Adapter Configuration Wizard - Step 7 of 8

Messages

Define the message for the Synchronous Read File operation. Specify the Schema File Location and select the Schema Element that defines the messages in the incoming files. Use the Browse button to find an existing schema definition. If you check 'Schema is Opaque', then you do not need to specify a Schema.

Message Schema

☐ Native format translation is not required (Schema is Opaque)

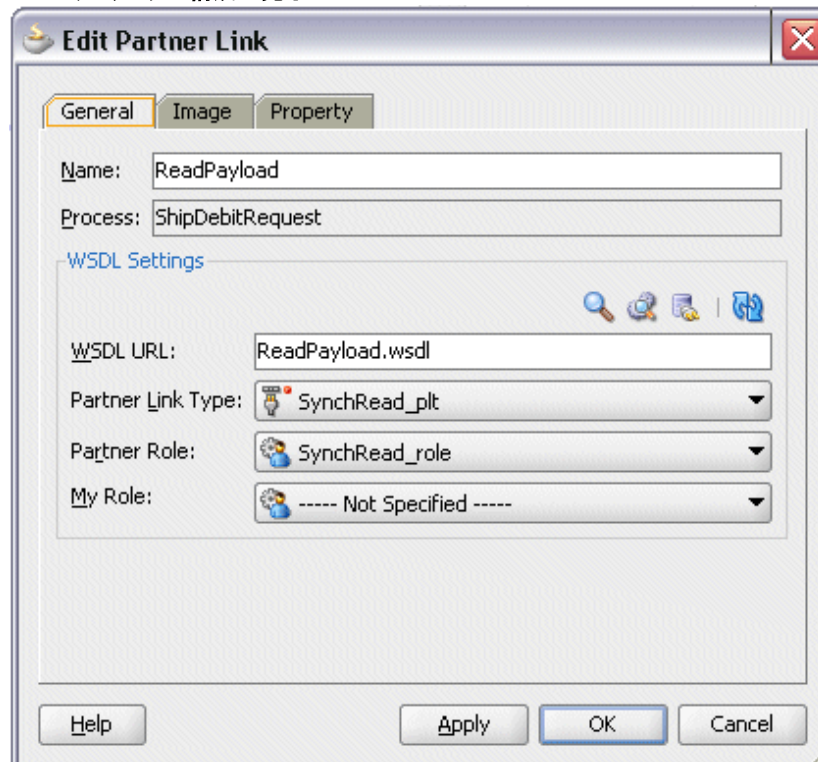
URL: xsd/APPS_ISG_CREATE_SD_REQUEST_OZF_SD_REQUEST_PUB_24CREATE_SD_.xsd

Schema Element: InputParameters

Buttons: Help, < Back, Next >, Finish, Cancel

10. 「次へ」→「終了」をクリックします。ウィザードによって、パートナ・リンクに対応する WSDL ファイルが生成されます。メインの「パートナ・リンクの作成」ダイアログ・ボックスが表示され、新しい WSDL ファイル ReadPayload.wsdl が指定されています。

パートナ・リンクの構成の完了



「適用」→「OK」をクリックして構成を完了し、ファイル・アダプタ・サービスに必要な WSDL 設定を指定してパートナ・リンクを作成します。

BPEL プロセス・ダイアグラムに ReadPayload パートナ・リンクが表示されます。

11. アプリケーション・ウィンドウで、ファイル ReadPayload_file.jca に移動します。DeleteFile プロパティの値を false に設定します。

プロパティ値の設定

```
1 <adapter-config name="ReadPayload" adapter="File Adapter" wsdlLocation="ReadPayload.wsdl" xmlns="http://platform.integration.
2
3 <connection-factory location="eis/FileAdapter" adapterRef="" />
4 <endpoint-interaction portType="SynchRead_ptt" operation="SynchRead">
5 <interaction-spec className="oracle.tip.adapter.file.outbound.FileReadInteractionSpec">
6 <property name="PhysicalDirectory" value="/usr/tmp"/>
7 <property name="DeleteFile" value="false"/>
8 <property name="FileName" value="InputCreateSDRequest.xml"/>
9 </interaction-spec>
10 </endpoint-interaction>
11
12 </adapter-config>
```

Invoke アクティビティの追加

このステップでは、次の 2 つの Invoke アクティビティを構成します。

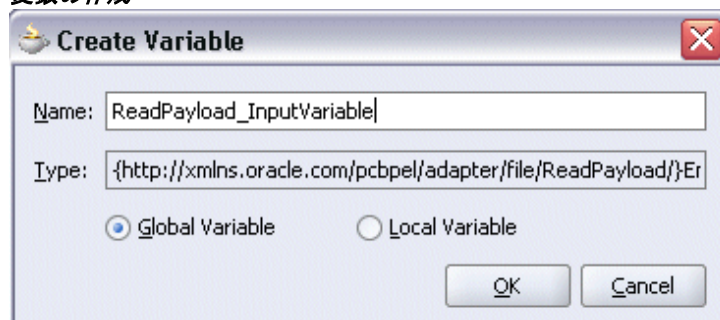
- ファイル・アダプタの ReadPayload パートナ・リンクを使用して最初の **Assign** アクティビティから渡される要求作成詳細を読み取ります。
- **Assign** アクティビティから受信したペイロードおよび要求詳細を送信して、OZF_SD_REQUEST_PUB パートナ・リンクを使用することで出荷およびデビット要求を作成します。

ReadPayload パートナ・リンクの Invoke アクティビティを追加するには:

1. Oracle JDeveloper で、コンポーネント・パレットから「**BPEL コンストラクト**」を展開します。**Invoke** アクティビティを、プロセス・ダイアグラムの中央のスイムレーンの、**receiveInput** と **replyOutput** のアクティビティの間にドラッグ・アンド・ドロップします。
2. **Invoke** アクティビティを ReadPayload サービスにリンクします。「編集 Invoke」ダイアログ・ボックスが表示されます。
3. **Invoke** アクティビティの名前を入力し、「**入力変数**」フィールドの横にある「**作成**」アイコンをクリックして、新しい変数を作成します。「変数の作成」ダイアログ・ボックスが表示されます。

入力変数名として ReadPayload_InputVariable と入力します。デフォルト名を受け入れることもできます。

変数の作成



「**グローバル変数**」を選択して、「**OK**」をクリックします。

4. 「**出力変数**」フィールドの横にある「**入力変数の作成**」アイコンをクリックして、新しい変数を作成します。「変数の作成」ダイアログ・ボックスが表示されます。

出力変数名として ReadPayload_OutputVariable と入力します。デフォルト名を受け入れることもできます。

「**グローバル変数**」を選択して、変数の名前を入力します。「**OK**」をクリックします。

Invoke アクティビティの編集

Edit Invoke

Annotations Assertions Skip Condition Headers

General Correlations Properties

Name: InvokeReadPayload

Conversation ID:

Detail Label:

☐ Invoke as Detail

Interaction Type: Partner Link

Partner Role Web Service Interface

Partner Link: ReadPayload

Operation: SynchRead

Variables

Input: ReadPayload_InputVariable

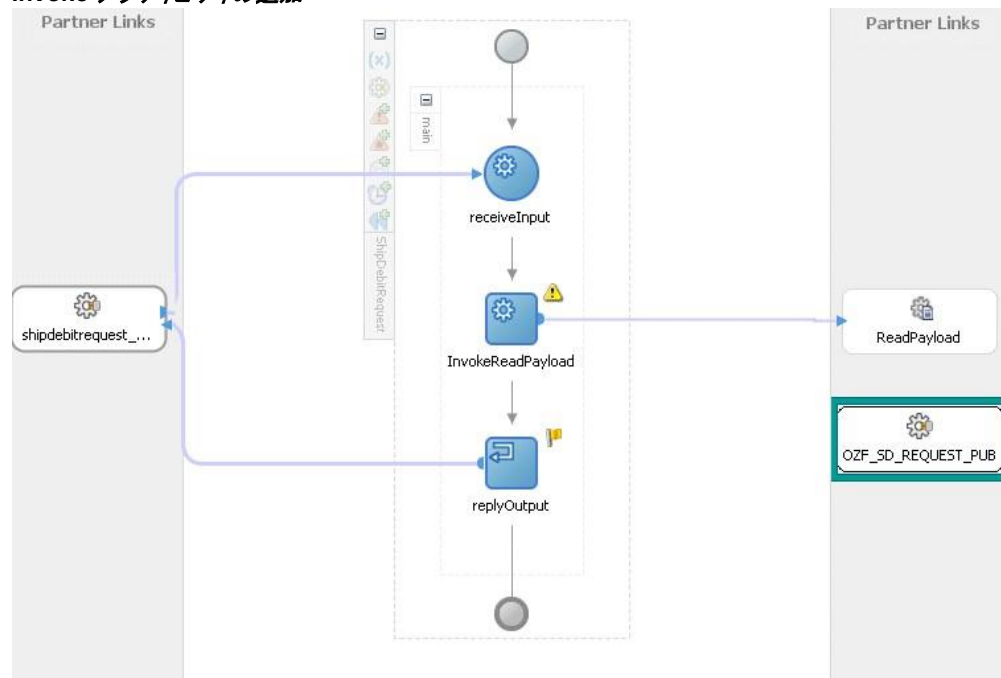
Output: ReadPayload_OutputVariable

Help Apply OK Cancel

5. 「編集 Invoke」ダイアログ・ボックスで「適用」→「OK」をクリックして、**Invoke** アクティビティの構成を終了します。

プロセス・ダイアグラムに **Invoke** アクティビティが表示されます。

Invoke アクティビティの追加



OZF_SD_REQUEST_PUB パートナ・リンクの Invoke アクティビティを追加するには:

1. Oracle JDeveloper で、コンポーネント・パレットから「**BPEL コンストラクト**」を展開します。
Invoke アクティビティを、プロセス・ダイアグラムの中央のスイムレーンの、最初の **Invoke** アクティビティと **replyOutput** のアクティビティの後にドラッグ・アンド・ドロップします。
2. **Invoke** アクティビティを OZF_SD_REQUEST_PUB サービスにリンクします。「編集 Invoke」ダイアログ・ボックスが表示されます。
3. **Invoke** アクティビティの名前を入力します(Invoke_EBS_SDR_Service など)。
「操作」フィールドで、ドロップダウン・リストから CREATE_SD_REQUEST を選択します。
4. グローバルな入力変数および出力変数を CREATE_SD_REQUEST_InputVariable および CREATE_SD_REQUEST_OutputVariable として作成します。
「起動の編集」で「OK」をクリックします。
「編集 Invoke」ダイアログ・ボックスで「適用」→「OK」をクリックして、**Invoke** アクティビティの構成を終了します。
プロセス・ダイアグラムに **Invoke** アクティビティが表示されます。

Assign アクティビティの追加

このステップでは、次の目的の 4 つの **Assign** アクティビティを構成します。

1. 出荷およびデビット SOAP 要求の SOAHeader 詳細の設定。

注意: サービスの実行中に、アプリケーション・コンテキストを設定するために要求される値を SOAHeader 要素で渡すには、BPEL プロセスの特定の変数を設定する必要があります。これらの SOAHeader の要素は、*Responsibility*、*RespApplication*、*SecurityGroup*、*NLSLanguage* および *Org_Id* です。

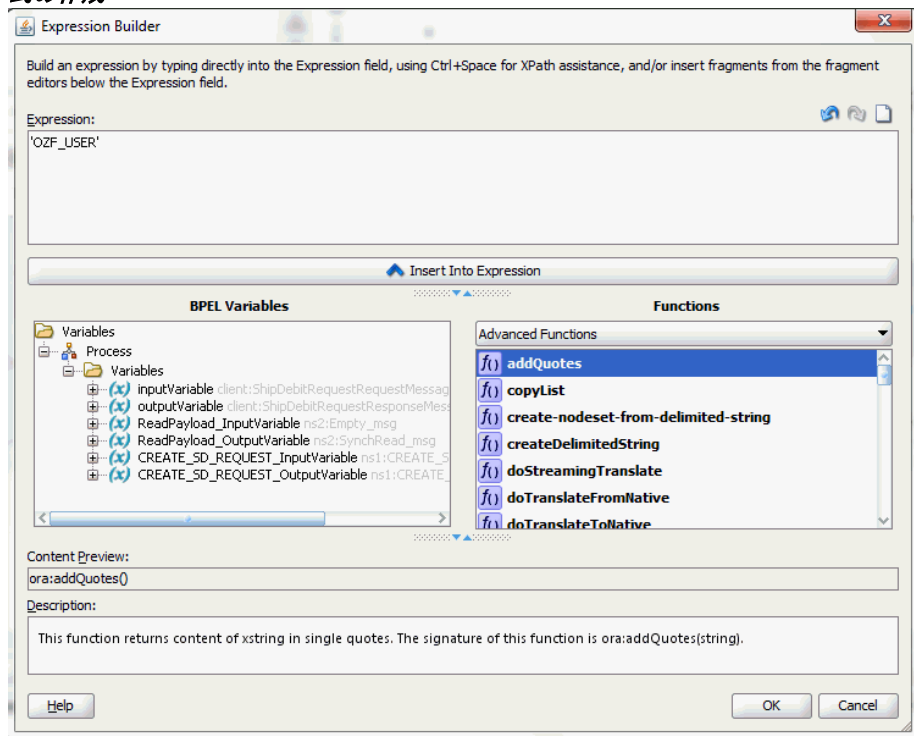
2. SOAP 要求の入力ペイロードの設定。
3. SOAP 要求の入力の設定。
4. 出力する SOAP 応答の設定。

SOAHeader 詳細を設定する最初の Assign アクティビティを追加するには:

SOAHeader パラメータの割当て:

1. Oracle JDeveloper で、コンポーネント・パレットから「**BPEL コンストラクト**」を展開します。**Assign** アクティビティを、プロセス・ダイアグラムの中央のスイムレーンの、先ほど作成したばかりの 2 つの **Invoke** アクティビティの間にドラッグ・アンド・ドロップします。
2. **Assign** アクティビティをダブルクリックして、「編集 Assign」ダイアログ・ボックスにアクセスします。
3. 「一般」タブをクリックして、Assign アクティビティの名前を入力します(SetSOAHeader など)。
4. 「コピー・ルール」タブを選択して、ターゲット・ツリーを展開します。
 - 「式」アイコンをクリックして、「式ビルダー」ダイアログを起動します。

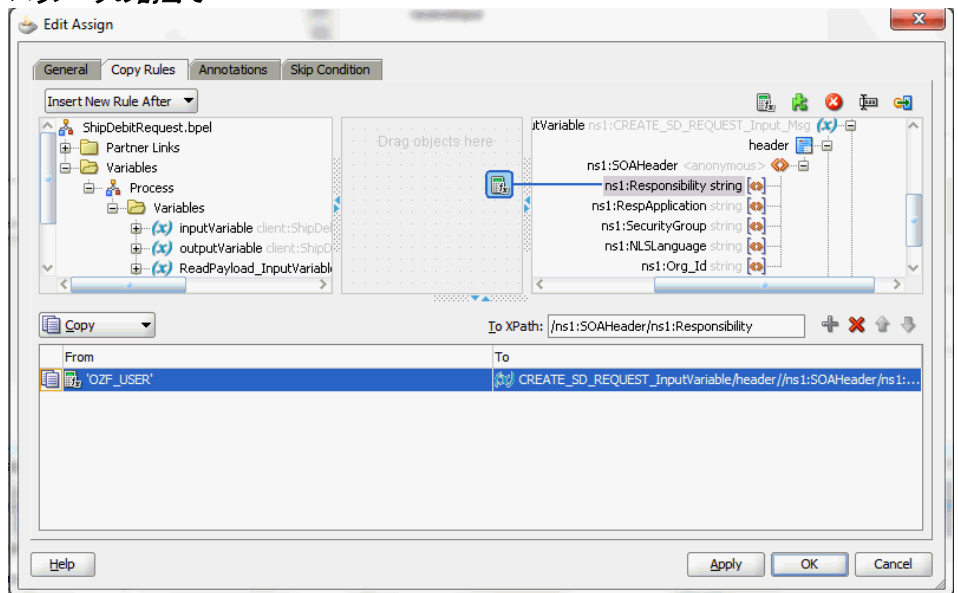
式の作成



「式」ボックスに OZF_USER と入力します。「OK」をクリックします。「割当ての編集」ダイアログの中央の、「元」と「先」のナビゲーション・ツリー・ノードの間に、式の値 (OZF_USER) を持つ「式」アイコンが表示されます。

- 「先」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「CREATE_SD_REQUEST_InputVariable」→「header」→「ns1:SOAHeader」にナビゲートし、「ns1:Responsibility」を選択します。「送信先 XPath」値が表示されます。
- 「式」アイコンをドラッグして、識別したばかりのターゲット・ノード(ns1:Responsibility)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

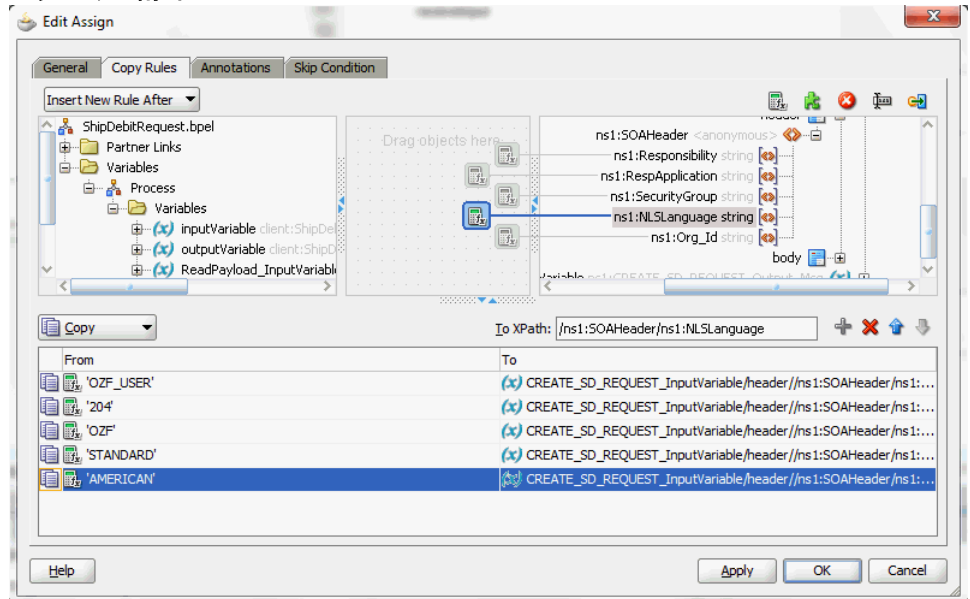
パラメータの割当て



5. 「式」アイコンをクリックして「式ビルダー」ダイアログを起動して、2 目目のパラメータのペアを入力します。
 - 「式」ボックスに **OZF** と入力します。「**OK**」をクリックします。「割当ての編集」ダイアログの中央の、「元」と「先」のナビゲーション・ツリー・ノードの間に、式の値(**OZF**)を持つ「式」アイコンが表示されます。
 - 「先」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「**CREATE_SD_REQUEST_InputVariable**」→「header」→「**ns1:SOAHeader**」にナビゲートし、「**ns1:RespApplication**」を選択します。「送信先 XPath」フィールドに、選択したエントリが含まれる必要があります。
 - 「式」アイコンをドラッグして、識別したばかりのターゲット・ノード (**ns1:RespApplication**) に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。
6. 「式」アイコンをクリックして「式ビルダー」ダイアログを起動して、3 目目のパラメータのペアを入力します。
 - 「式」ボックスに **STANDARD** と入力します。「**OK**」をクリックします。「割当ての編集」ダイアログの中央の、「元」と「先」のナビゲーション・ツリー・ノードの間に、式の値(**STANDARD**)を持つ「式」アイコンが表示されます。
 - 「先」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「**CREATE_SD_REQUEST_InputVariable**」→「header」→「**ns1:SOAHeader**」にナビゲートし、「**ns1:SecurityGroup**」を選択します。「XPath」フィールドに、選択したエントリが含まれる必要があります。

- 「式」アイコンをドラッグして、識別したばかりのターゲット・ノード(ns1:SecurityGroup)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。
- 7. 「式」アイコンをクリックして「式ビルダー」ダイアログを起動して、4 つ目のパラメータのペアを入力します。
 - 「式」ボックスに AMERICAN と入力します。「OK」をクリックします。「割当ての編集」ダイアログの中央の、「元」と「先」のナビゲーション・ツリー・ノードの間に、式の値 (AMERICAN) を持つ「式」アイコンが表示されます。
 - 「先」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「CREATE_SD_REQUEST_InputVariable」→「header」→「ns1:SOAHeader」にナビゲートし、「ns1:NLSLanguage」を選択します。「XPath」フィールドに、選択したエントリが含まれる必要があります。
 - 「式」アイコンをドラッグして、識別したばかりのターゲット・ノード(ns1:NLSLanguage)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。
- 8. 「式」アイコンをクリックして「式ビルダー」ダイアログを起動して、5 つ目のパラメータのペアを入力します。
 - 「式」ボックスに 204 と入力します。「OK」をクリックします。「割当ての編集」ダイアログの中央の、「元」と「先」のナビゲーション・ツリー・ノードの間に、式の値(204)を持つ「式」アイコンが表示されます。
 - 「先」のナビゲーション・ツリーで「変数」タイプを選択します。「変数」→「プロセス」→「変数」→「CREATE_SD_REQUEST_InputVariable」→「header」→「ns1:SOAHeader」にナビゲートし、「ns1:Org_Id」を選択します。「XPath」フィールドに、選択したエントリが含まれる必要があります。
 - 「式」アイコンをドラッグして、識別したばかりのターゲット・ノード(ns1:Org_Id)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。
- 9. 「編集 Assign」ダイアログ・ボックスが表示されます。

パラメータの割当て



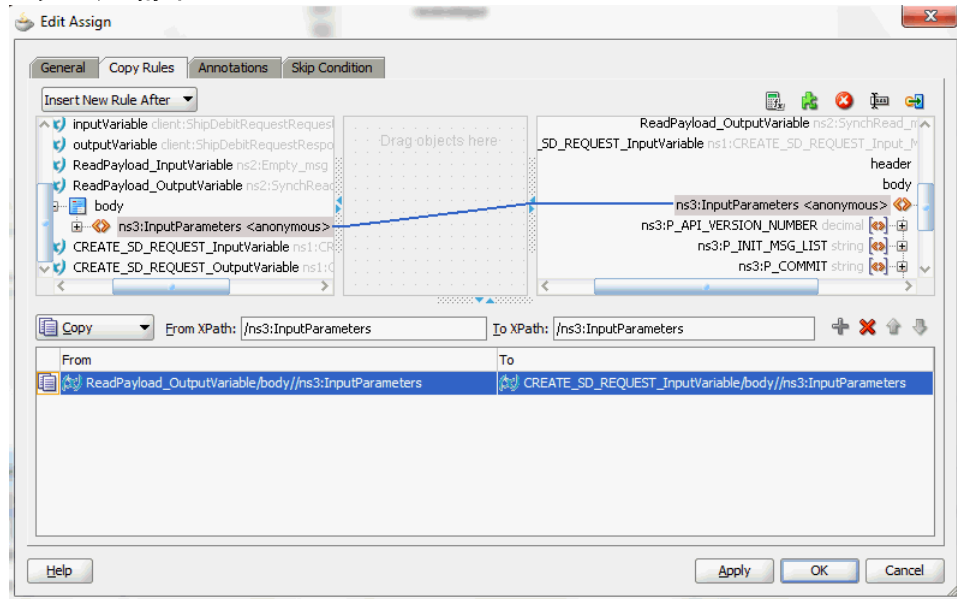
10. 「適用」→「OK」をクリックして、**Assign** アクティビティの構成を完了します。

ペイロード情報を **Invoke_EBS_SDR_ServiceInvoke** アクティビティに渡す 2 つ目の **Assign** アクティビティを入力するには:

1. **Assign** アクティビティをコンポーネント・パレットの「**BPEL コンストラクト**」からプロセス・ダイアグラムの中央のスイムレーンの、**SetSOAHeaderAssign** アクティビティと **Invoke_EBS_SDR_Service Invoke** アクティビティの間にドラッグ・アンド・ドロップすることによって、2 つ目の **Assign** アクティビティを追加します。
2. 1 つ目の **Assign** アクティビティの作成で説明したステップ 2 とステップ 3 を繰り返して、**SetPayload** という 2 つ目の **Assign** アクティビティを追加します。
3. 「コピー・ルール」タブを選択して、ソース・ツリーとターゲット・ツリーを展開します。
 - 「元」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「**InputVariable**」→「**ReadPayload_OutVariable**」→「**body**」にナビゲートし、「**ns3:InputParameters**」を選択します。「送信元 XPath」フィールドも表示されます。
 - 「先」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「**CREATE_SD_REQUEST_InputVariable**」→「**body**」にナビゲートし、「**ns3:InputParameters**」を選択します。「送信先 XPath」フィールドも表示されます。

ソース・ノード(**InputParameters**)をドラッグして、識別したばかりのターゲット・ノード(**InputParameters**)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

パラメータの割当て



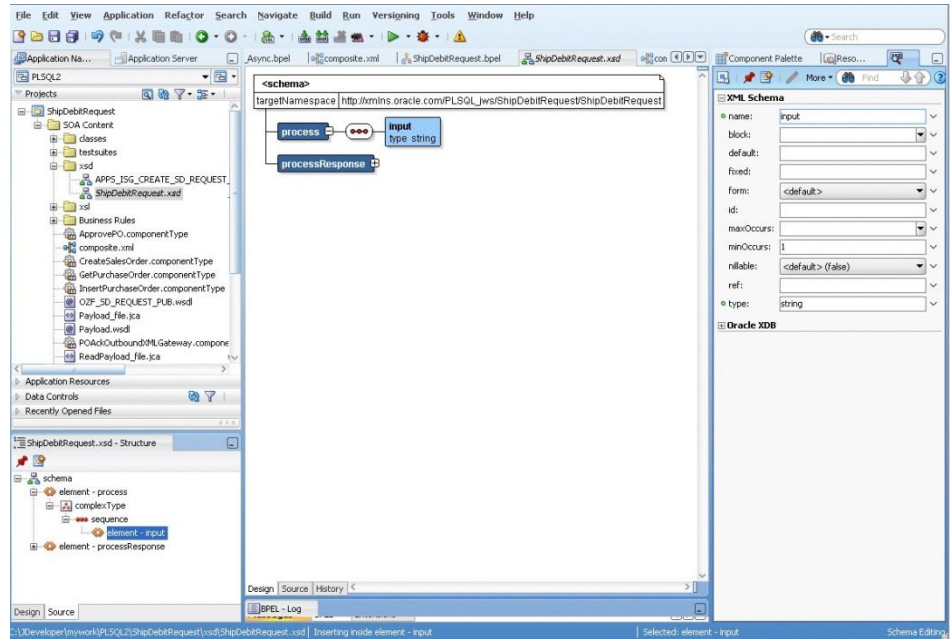
4. 「適用」→「OK」をクリックして、2 つ目の割当てアクティビティの構成を完了します。

BPEL プロセス入力要求のスキーマの定義

SOAP 要求の入力要求を設定する前に、BPEL プロセス要求に必要なスキーマを定義する必要があります。

1. アプリケーション・ナビゲータ・ウィンドウから、「ShipDebitRequest」→「SOA コンテンツ」→「xsd」フォルダを展開して ShipDebitRequest.xsd ファイルを開きます。
2. 設計モードで、process を展開してプロセス要求内の要素を表示します。

BPEL プロセス要求のスキーマの定義



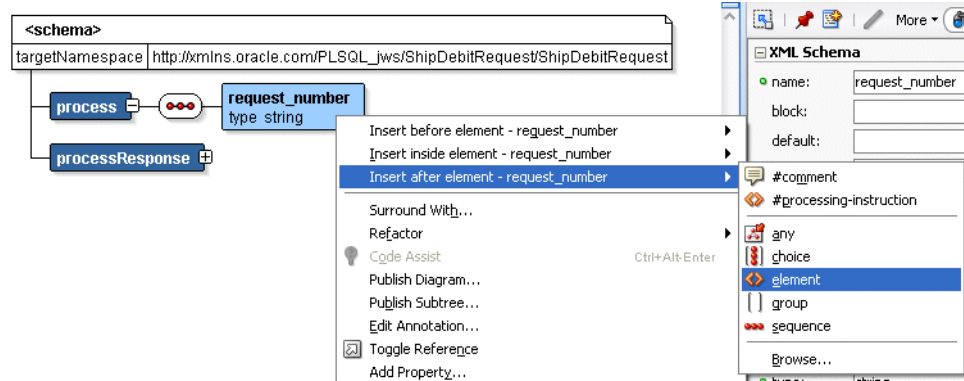
3. 要素 input をクリックし、「XML スキーマ」ウィンドウでプロパティ名を input から request_number に変更します。

4. request_number 要素を選択して右クリックし、ポップアップ・メニューを開きます。

「element - request_number の後ろに挿入」→「element」オプションを選択します。新しい要素 element1 が、スキーマ設計ウィンドウの request_number 要素の下に表示されます。

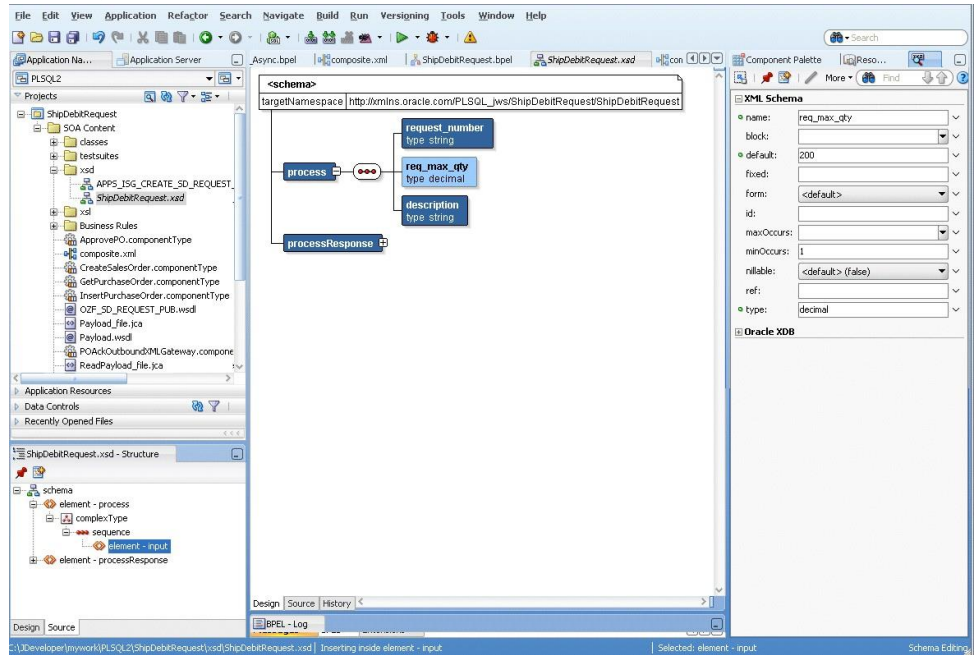
要素のプロパティ・セクションから、名前を element1 から description に変更し、タイプを string として入力します。

スキーマ要素の追加



5. 同様に、`req_max_qty` という別の要素を要素 `description` の後に挿入します。デフォルト値を 200、タイプを `decimal` として入力します。
マウスを右クリックして、「再ビルド」オプションを選択します。

スキーマ要素の再ビルド



ログでコンパイル・メッセージを探し、コンパイルが成功したことを確認します。

入力要求を **Invoke_EBS_SDR_ServiceInvoke** アクティビティに渡す 3 つ目の **Assign** アクティビティを設定するには:

1. **Assign** アクティビティをコンポーネント・パレットの「**BPEL コンストラクト**」からプロセス・ダイアグラムの中央のスイムレーンの、2 つ目の **Assign** アクティビティ **SetPayload** と **Invoke_EBS_SDR_Service Invoke** アクティビティの間にドラッグ・アンド・ドロップすることによって、3 つ目の **Assign** アクティビティを追加します。
2. 1 つ目の **Assign** アクティビティの作成で説明したステップ 2 とステップ 3 を繰り返して、**SetInput** という 3 つ目の **Assign** アクティビティを追加します。
3. 「コピー・ルール」タブを選択して、ソース・ツリーとターゲット・ツリーを展開します。
 - 「元」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「inputVariable」→「Payload」→「client:ShipDebitRequestProcessRequest」にナビゲートし、「client:request_number」を選択します。「送信元 XPath」フィールドも表示されます。
 - 「先」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「Create_SD_REQUEST_InputVariable」→「Body」→「ns4:InputParameters」→「ns4: P_SDR_HDR_REC」にナビゲートし、「ns4:REQUEST_NUMBER」を選択します。「送信先 XPath」フィールドも表示されます。

ソース・ノード(client:request_number)をドラッグして、識別したばかりのターゲット・ノード

(ns4: REQUEST_NUMBER)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

4. 次の値を使用して、2 つ目のパラメータのペアを入力します。

- 「元」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「inputVariable」→「Payload」→「client:ShipDebitRequestProcessRequest」にナビゲートし、「client:description」を選択します。「XPath」フィールドに、選択したエントリが含まれる必要があります。
- 「先」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「Create_SD_REQUEST_InputVariable」→「Body」→「ns4:InputParameters」→「ns4: P_SDR_HDR_REC」にナビゲートし、「ns4:REQUEST_DESCRIPTION」を選択します。「XPath」フィールドに、選択したエントリが含まれる必要があります。

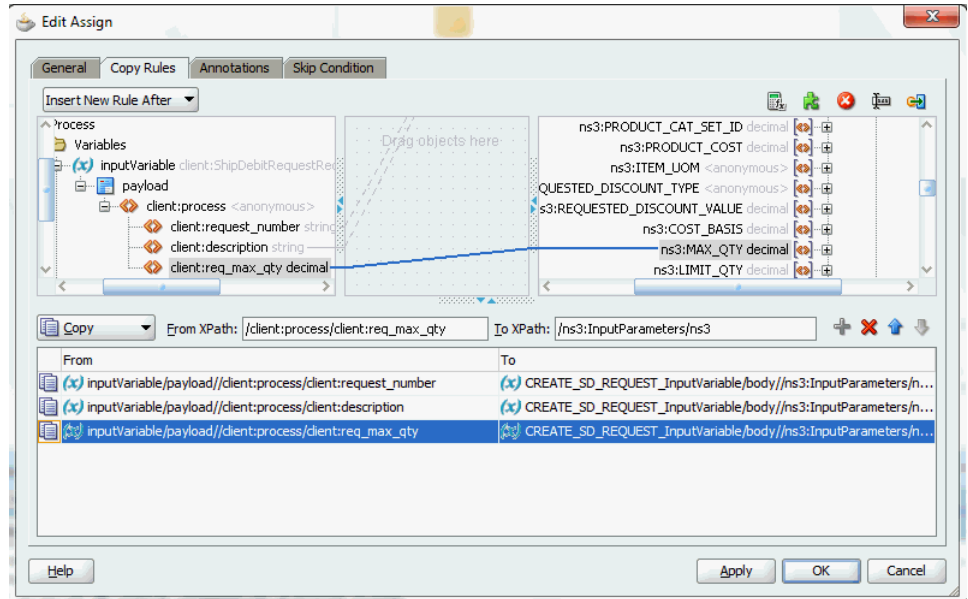
ソース・ノード(client:description)をドラッグして、識別したばかりのターゲット・ノード(ns4: REQUEST_DESCRIPTION)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

5. 次の値を使用して、3 つ目のパラメータのペアを入力します。

- 「元」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「inputVariable」→「Payload」→「client:ShipDebitRequestProcessRequest」にナビゲートし、「client:req_max_qty」を選択します。「XPath」フィールドに、選択したエントリが含まれる必要があります。
- 「先」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「Create_SD_REQUEST_InputVariable」→「Body」→「ns4:InputParameters」→「ns4: P_SDR_LINES_TBL」→「ns4:P_SDR_LINES_TBL_ITEM」にナビゲートし、「ns4: MAX_QTY」を選択します。「XPath」フィールドに、選択したエントリが含まれる必要があります。

ソース・ノード(client:req_max_qty)をドラッグして、識別したばかりのターゲット・ノード(ns4: MAX_QTY)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

パラメータの割当て



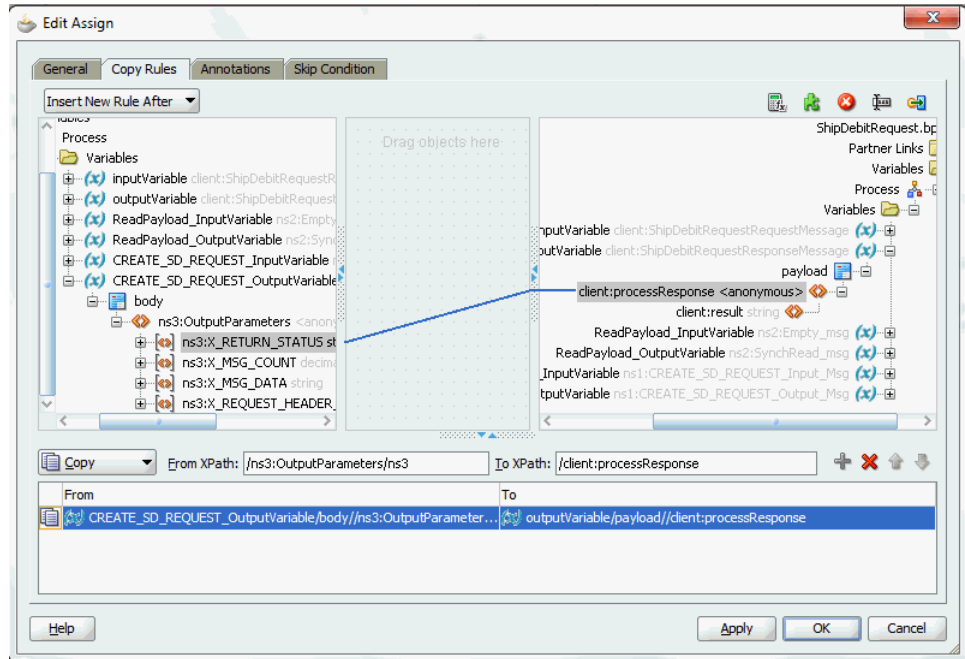
6. 「適用」→「OK」をクリックして、**Assign** アクティビティの構成を完了します。

SOAP 応答を出力に設定する 4 つ目の **Assign** アクティビティを追加するには:

1. **Assign** アクティビティをコンポーネント・パレットの「**BPEL コンストラクト**」からプロセス・ダイアグラムの中央のスイムレーンの、**Invoke_EBS_SDR_Service Invoke** アクティビティと **replyOutput** アクティビティの間にドラッグ・アンド・ドロップすることによって、4 つ目の **Assign** アクティビティを追加します。
2. 1 つ目の **Assign** アクティビティの作成で説明したステップ 2 とステップ 3 を繰り返して、**SetResponse** という 4 つ目の **Assign** アクティビティを追加します。
3. 「コピー・ルール」タブを選択して、ソース・ツリーとターゲット・ツリーを展開します。
 - 「元」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「**CREATE_SD_REQUEST_OutputVariable**」→「**body**」→「**ns3:OutputParameters**」にナビゲートし、「**ns3:X_RETURN_STATUS**」を選択します。
 - 「先」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「**outputVariable**」→「**payload**」にナビゲートし、「**client:processResponse**」を選択します。

ソース・ノード(**ns3:X_RETURN_STATUS**)をドラッグして、識別したばかりのターゲット・ノード(**client:processResponse**)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 **Assign**」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

パラメータの割当て



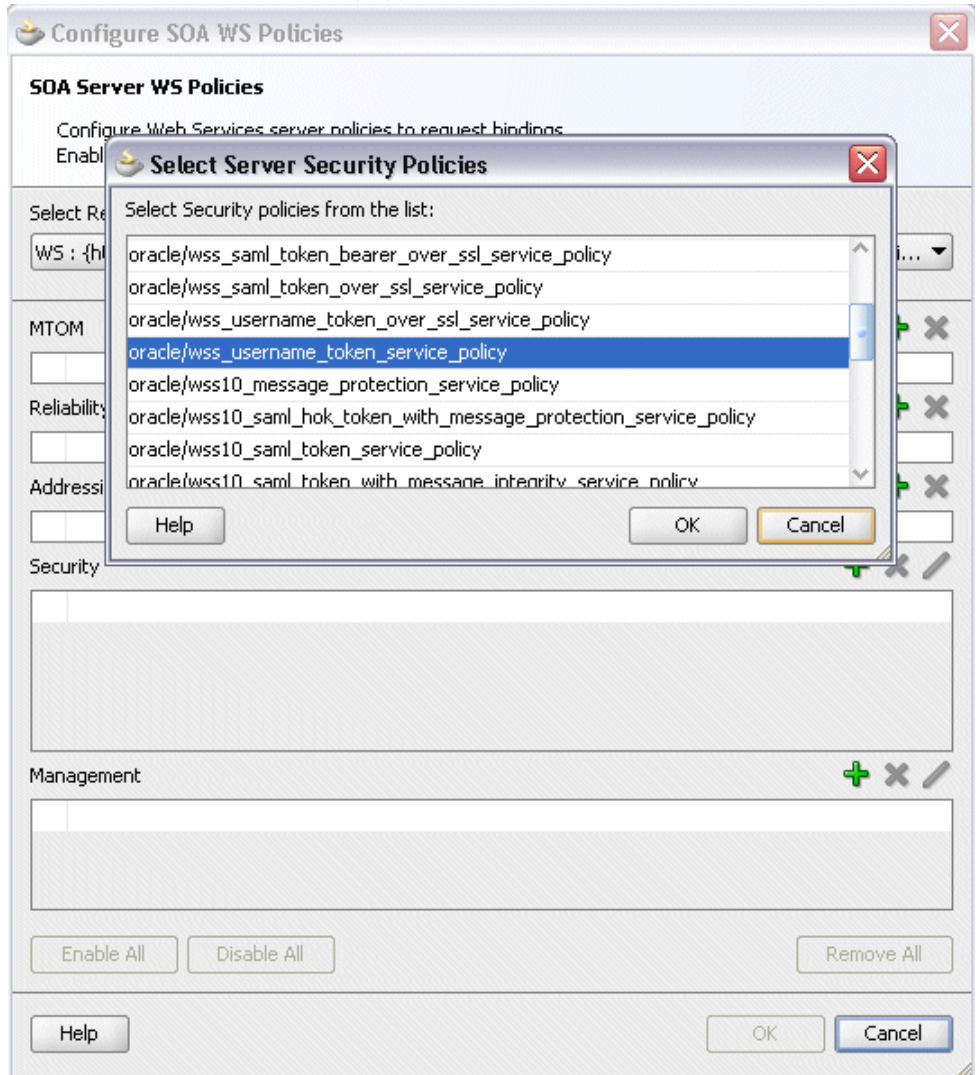
4. 「適用」→「OK」をクリックして、**Assign** アクティビティの構成を完了します。

Web サービス・ポリシーの構成

次のステップを使用して、設計時のセキュリティ・ポリシーを追加します。

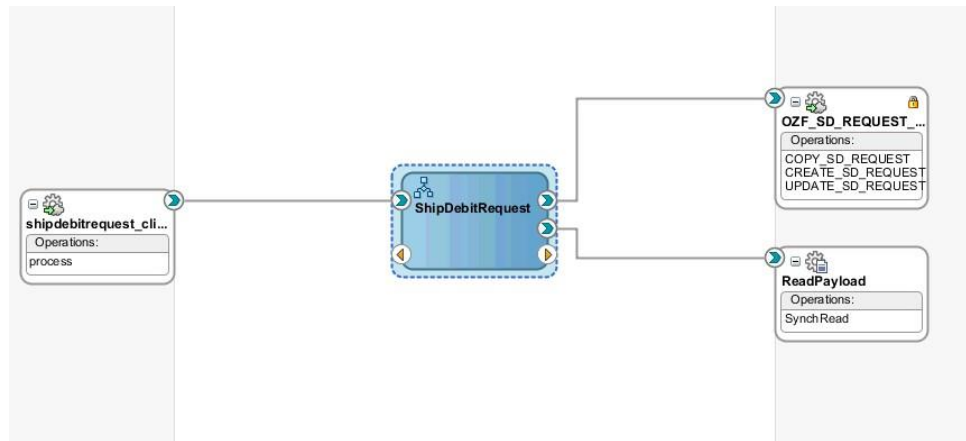
1. 「SOA コンテンツ」→「ビジネス・ルール」→**composite.xml** に移動します。
OZF_SD_REQUEST_PUB サービスを右クリックし、ドロップダウン・リストから「WS ポリシーの構成」を選択します。
2. 「SOA WS ポリシーの構成」ダイアログが表示されます。
「セキュリティ」セクションで「セキュリティ・ポリシーの追加」アイコン(+)をクリックします。
サーバー・セキュリティ・ポリシーの選択ダイアログが表示されます。

サーバー・セキュリティ・ポリシーの選択



「oracle/wss_username_token_service_policy」を選択して、「OK」をクリックします。添付されたセキュリティ・ポリシーが「セキュリティ」セクションに表示されます。

composite.xml の OZF_SD_REQUEST_PUB サービスにロック・アイコンが表示されます(これは、セキュリティ・ポリシーが正常に添付されたことを示します)。



3. ナビゲーション・メニューから、「表示」→「プロパティ・インスペクタ」を選択して、OZF_SD_REQUEST_PUB サービス・コンポーネントの「プロパティ・インスペクタ」ウィンドウを表示します。

「プロパティ」セクションで、バインディング・プロパティの「セキュリティ・ポリシーの追加」アイコン(+)をクリックします。「プロパティの作成」ダイアログが表示されます。

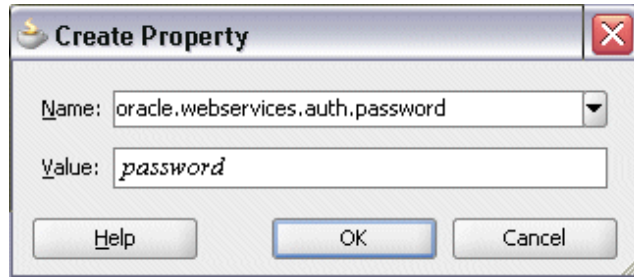
「名前」フィールドに `oracle.webservices.auth.username` と入力して、値として `operations` と入力します。

プロパティ値の入力

「OK」をクリックします。

4. 「プロパティ」セクションのバインディング・プロパティで「セキュリティ・ポリシーの追加」アイコン(+)を再度クリックする、同じアプローチを使用します。「名前」フィールドに `oracle.webservices.auth.password` と入力します。「値」フィールドに、operations ユーザーに紐づくパスワードを入力します。

プロパティ値の入力

A screenshot of the 'Create Property' dialog box in Oracle JDeveloper. The dialog has a title bar with a close button. It contains two text input fields: 'Name' with the value 'oracle.webservices.auth.password' and 'Value' with the value 'password'. At the bottom, there are three buttons: 'Help', 'OK', and 'Cancel'. The 'OK' button is highlighted with a blue border.

「OK」をクリックします。

「プロパティ」セクションに、選択したプロパティの名前と値の両方が表示されます。

composite.xml の「ソース」タブをクリックして、

oracle.webservices.auth.username と

oracle.webservices.auth.password のプロパティ名および関連する値が

OZF_SD_REQUEST_PUB 参照に追加されることに注目してください。

プロパティ情報の確認

```
27 <reference name="OZF_SD_REQUEST_PUB"
28   ui:wsdlLocation="OZF_SD_REQUEST_PUB.wsdl">
29   <interface.wsdl interface="http://xmlns.oracle.com/apps/ozf/soapprovider/plsql/ozf_sd_request_pub/#wsdl.interfa
30   <binding.ws port="http://xmlns.oracle.com/apps/ozf/soapprovider/plsql/ozf_sd_request_pub/#wsdl.endpoint(OZF_SD_
31     location="OZF_SD_REQUEST_PUB.wsdl" soapVersion="1.1">
32     <wsp:PolicyReference URI="oracle/wss_username_token_client_policy"
33       orawsp:category="security" orawsp:status="enabled"/>
34     <property name="oracle.webservices.auth.username" type="xs:string"
35       many="false" override="may">operations</property>
36     <property name="oracle.webservices.auth.password" type="xs:string"
37       many="false" override="may">password</property>
38   </binding.ws>
39 </reference>
```

同期 BPEL プロセスによる SOA コンポジットのデプロイおよびテスト

SOA コンポジットに含まれる BPEL クライアントから同期仕入出荷およびデビット要求サービス(OZF_SD_REQUEST_PUB)を起動するには、SOA コンポジットが Oracle WebLogic 管理対象サーバーにデプロイされている必要があります。これは、Oracle JDeveloper を使用して達成できます。コンポジットがデプロイされたら、Oracle Enterprise Manager Fusion Middleware Control コンソールからテストできます。

前提条件

Oracle JDeveloper を使用して BPEL プロセスによる SOA コンポジットをデプロイする前に、設計時環境とランタイム・サーバーの間の接続を確立する必要があります。必要なサーバー接続を構成する方法の詳細は、「サーバー接続の構成」(ページ B-1)を参照してください。

注意: WebLogic サーバーのローカル・インスタンスが使用される場合、Oracle JDeveloper から「実行」→「サーバー・インスタンスの起動」を選択して、WebLogic サーバーを起動します。WebLogic 管理サーバーの DefaultServer インスタンスが

正常に起動されたら、「実行中:DefaultServer」と「メッセージ」のログに<Server started in Running mode>と DefaultServer started のメッセージが表示されるはずです。

仕入先出荷およびデビット要求の作成に関するペイロード情報については、「仕入先出荷およびデビット要求の作成のためのサンプル・ペイロード」(ページ C-1)を参照してください。
次の実行時タスクを実行します。

1. BPEL プロセスによる SOA コンポジット・アプリケーションのデプロイ、ページ 3-38
2. BPEL プロセスによる SOA コンポジット・アプリケーションのテスト、ページ 3-43

BPEL プロセスによる SOA コンポジットのデプロイ

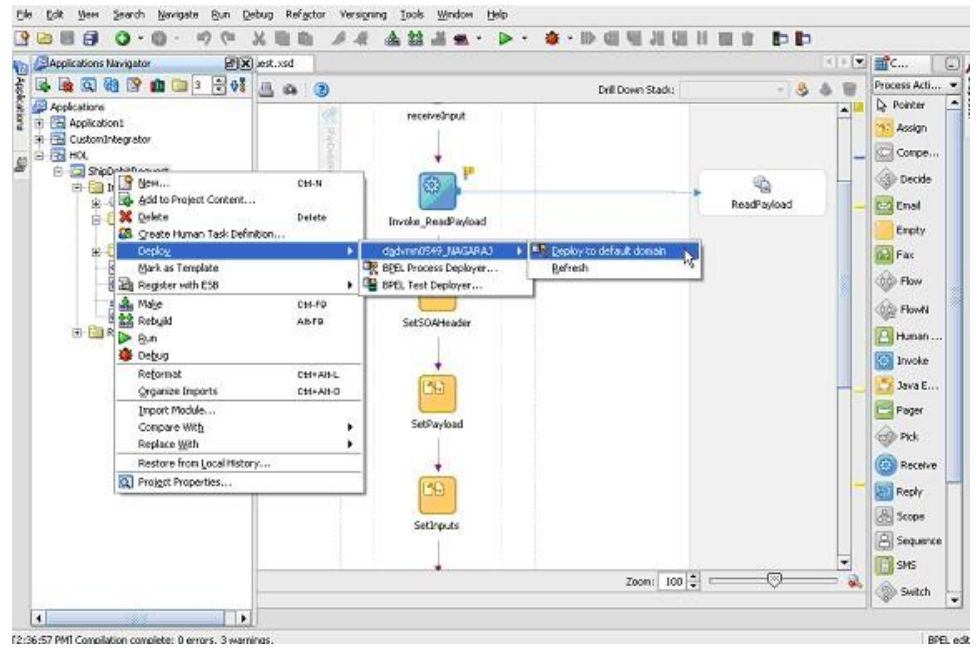
先ほど作成した BPEL プロセス(ShipDebitRequest.bpel)による SOA コンポジットは、実行する前にデプロイする必要があります。

SOA コンポジット・アプリケーションをデプロイするには:

1. JDeveloper のアプリケーション・ナビゲータで、**ShipDebitRequest** プロジェクトを選択します。
2. プロジェクトを右クリックして、メニューから「配置」→[project name]→**[serverConnection]**を選択します。

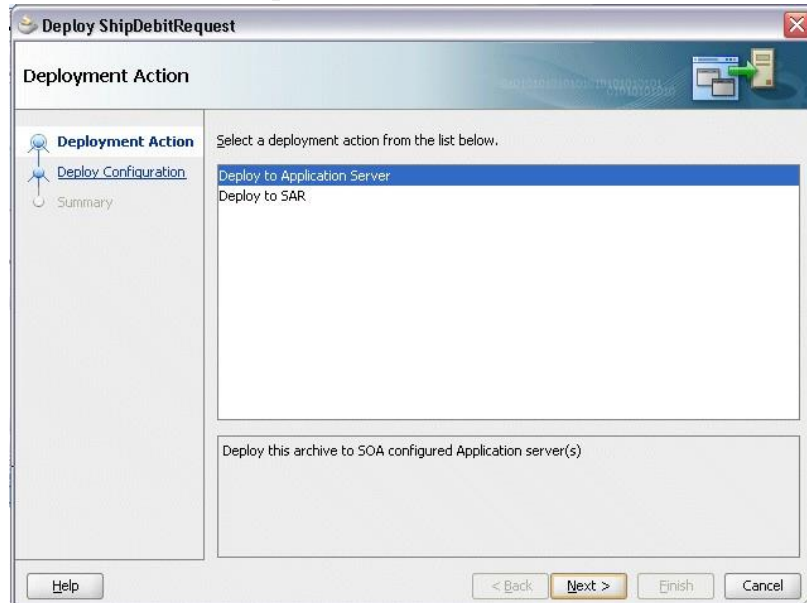
たとえば、接続を適切に設定している場合は、「デプロイ」→「ShipDebitRequest」→「SOAServer」を選択してプロセスをデプロイできます。

BPEL プロセスによる SOA コンポジットのデプロイ



注意: サーバー接続を初めて設定する場合は、「デプロイメント・アクション」ダイアログが表示されます。「アプリケーション・サーバーにデプロイ」を選択して、「次へ」をクリックします。

「デプロイメント・アクション」ダイアログ



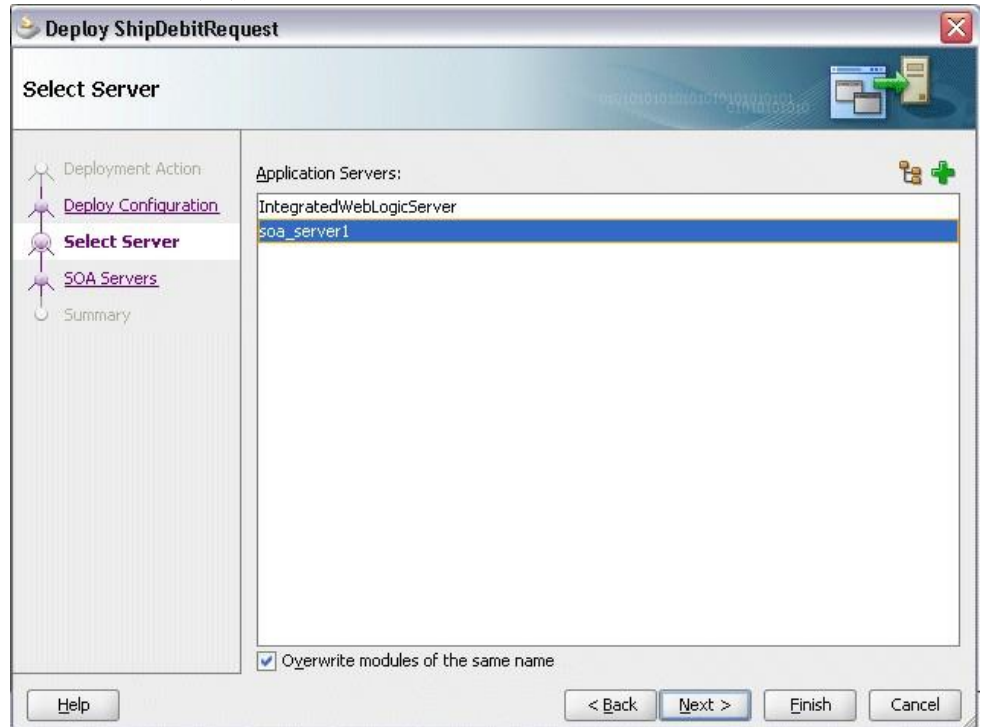
「構成のデプロイ」ダイアログで、「次へ」をクリックして新しいアプリケーション・サーバーを追加する前に、次の情報が選択されるようにします。

- 新規リビジョン ID: 1.0
- コンポジット・リビジョンをデフォルトとしてマークします。: このチェック・ボックスを選択します。
- 同じリビジョン ID で既存のコンポジットを上書きします。: このチェック・ボックスを選択します。

JDeveloper から新しい Oracle WebLogic Server 接続を作成するステップは、「サーバー接続の構成」(ページ B- 1)を参照してください。

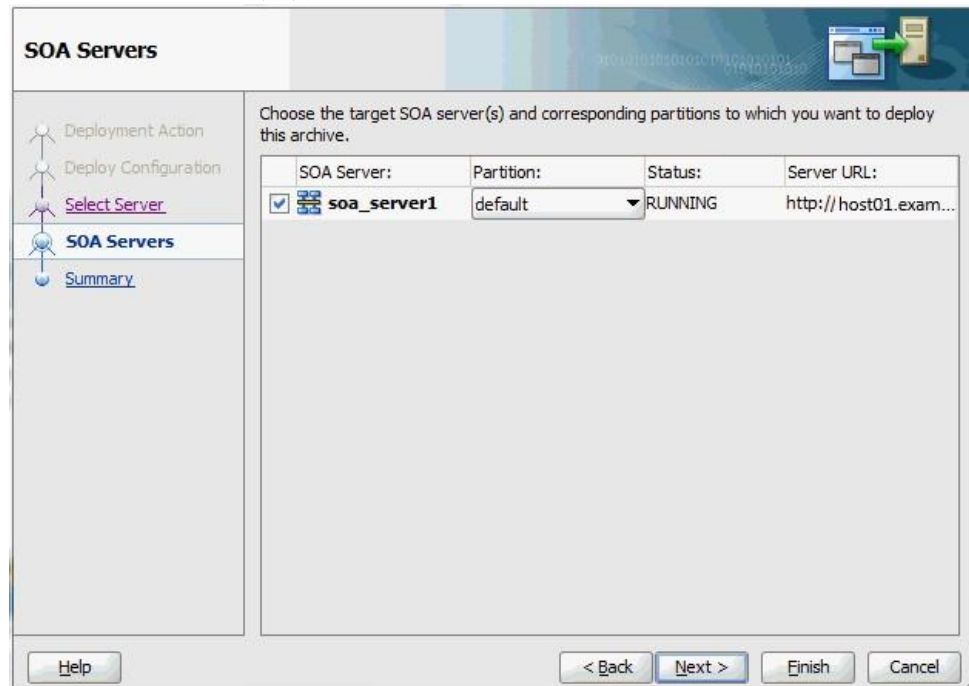
3. 「サーバーの選択」ダイアログで、以前にサーバー接続を確立した soa-server1 を選択します。「次へ」をクリックします。

必要なサーバーの選択



4. 「SOA サーバー」ダイアログで、デフォルトのターゲット SOA サーバー(soa-server1)の選択を受け入れます。

必要な SOA サーバーの選択

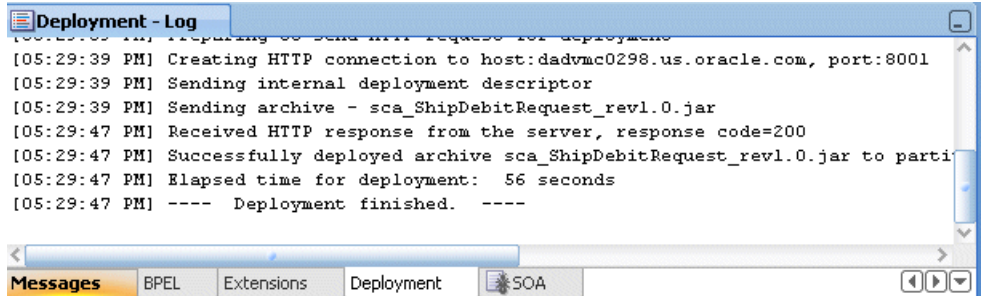


「次へ」→「終了」をクリックします。

5. Oracle JDeveloper セッションから初めてコンポジットをデプロイする場合は、「認証リクエスト」ウィンドウが表示されます。Oracle SOA Suite のインストール時に指定したユーザー名とパスワード情報を入力します。「OK」をクリックします。
6. デプロイ処理が始まります。SOA - ログ・ウィンドウで、デプロイ・プロセスをモニターして、コンパイルが成功したことを確認します。

デプロイメント - ログ・ウィンドウで、デプロイが成功したことを確認します。

デプロイ情報の表示



SOA コンポジット・アプリケーションのテスト

SOA コンポジット・アプリケーションに含まれる BPEL プロセスが正常にデプロイされたら、Oracle Enterprise Manager Fusion Middleware Control コンソールからプロセスを管理およびモニターできます。プロセスを手動で開始して、プロセスと統合インタフェースをテストすることもできます。

Oracle E-Business Suite にログオンして発注承認および発注確認プロセスを手動で開始し、関連するイベントが呼び出され、さらに、更新された発注詳細が XML ファイルに書き込まれることを確認できます。

SOA コンポジット・アプリケーションをテストするには:

1. Oracle Enterprise Manager Fusion Middleware Control コンソール

(<http://<servername>:<port>/em>)にナビゲートします。ログイン・ページが表示されます。

Oracle SOA Suite の詳細は、*Oracle Fusion Middleware Oracle SOA Suite 管理者ガイド*を参照してください。

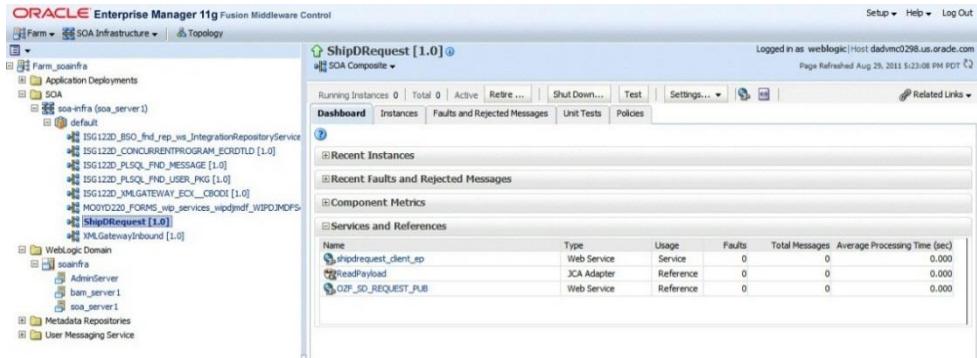
2. インストール時に指定したユーザー名とパスワード情報を入力します。「**ログイン**」をクリックして、ファームにログインします。デプロイしたコンポジット(ShipDebitRequest)がアプリケーション・ナビゲーション・ツリーに表示されます。

複数のターゲット Oracle Enterprise Manager Fusion Middleware Control コンソール・ファームがある場合、適切なターゲット・インスタンス・ファームを選択する必要がある場合があります。

3. 「ファーム」ナビゲーション・ペインのツリーで SOA ノード、次に soa-infra ノードを展開し、SOA インフラストラクチャのホーム・ページおよびメニューを介して、soa-infra 管理対象サーバーで実行されているデプロイ済 SOA コンポジット・アプリケーションにアクセスします。

ShipDebitRequest [1.0]リンクをクリックします。

デプロイ済サービスの特徴



- 「ポリシー」タブをクリックし、設計時に OZF_SD_REQUEST_PUB サービス・バインディングにアタッチした oracle/wss_username_token_service_policy ポリシーがここに表示されていることに注目してください。

選択したセキュリティ・ポリシーの確認



- ShipDebitRequest [1.0]のホーム・ページで、「テスト」をクリックします。
- インスタンスを開始するための「Web サービスのテスト」ページが表示されます。

注意: 設計時に WS-Security 資格証明が入力されない場合、「リクエスト」タブの上部にある「セキュリティ」セクションの「WSS ユーザー名トークン」オプションを選択して、実行時に資格証明を入力できます。「ユーザー名」フィールドに operations、「パスワード」フィールドに、operations ユーザーに紐づくパスワードを入力します。

WS-Security 資格証明の指定

The dialog shows the 'Request' tab. Under the 'Security' section, 'WSS Username Token' is selected. The 'Username' field contains 'operations' and the 'Password' field contains '*****'.

次のフィールドを、「引数の入力」セクションで使用する XML ペイロード・データとして指定できます。

ペイロード・データの入力

The 'Input Arguments' dialog shows a table with the following data:

Name	Type	Value
* payload	payload	
* request_number	string	BPEL-1
* description	string	ShipDebitRequest
* req_max_qty	decimal	100

At the bottom right, there is a 'Test Web Service' button.

- request_number: 一意の番号をこのフィールドに入力します(BPEL-1 など)。

注意: ここで入力する要求番号は、開始するたびに一意である必要があります。仕入先出荷およびデビット要求番号は、Oracle Trade Management の仕入先出荷およびデビットのユーザー間で一意である必要があります。

- description: 適切な説明情報を入力します。
- req_max_qty: 値として 100 を入力します。

「Web サービスのテスト」をクリックして、プロセスを開始します。

完了時に、「レスポンス」タブにテスト結果が表示されます。

7. コンソールでの SOAP レスポンスの確認

「レスポンス」タブで、「メッセージ・フロー・トレースの起動」リンクをクリックして、同期コンポジット・アプリケーションの結果を表示します。「フローのトレース」ページが表示されます。

「トレース」セクションで、ShipDebitRequest および OZF_SD_REQUEST_PUB コンポーネントが、アプリケーションが正常に実行されたことを示す「完了」状態であることを確認します。

「フォルト」セクションで、テスト中にエラーが発生したかどうかを確認できます。

8. BPEL サービス・コンポーネント・インスタンスのリンク(ShipDebitRequest など)をクリックして、「インスタンス」ページを表示します。このページでは、「監査証跡」タブで BPEL アクティビティの実行詳細を表示できます。

「フロー」タブをクリックして、BPEL プロセスのフロー・ダイアグラムを確認します。プロセス・ダイアグラムのアクティビティをクリックして、アクティビティの詳細と、プロセスを介したペイロードのフローを表示します。

9. 作成された仕入先出荷およびデビット要求の Oracle Trade Management での確認

「Oracle Trade Management ユーザー」職責を持つユーザーとして Oracle E-Business Suite にログインします。ナビゲーション・メニューから「仕入先出荷およびデビット」リンクを選択して、「出荷およびデビットの概要」ウィンドウを開きます。

10. 先ほど入力した要求番号 BPEL-1 がリストに表示されています。要求番号 BPEL-1 リンクをクリックして、作成された要求の「出荷およびデビット要求詳細」ページを開きます。詳細を確認します。

BPEL プロセスによる SOA コンポジット・アプリケーションからの非同期 Web サービスの起動

BPEL プロセスによる SOA コンポジット・アプリケーションのシナリオ

この例では、Oracle SOA Suite に配置された非同期サービス FND_USER_PKG のメソッド TESTUSERNAME を起動する BPEL プロセスの作成について説明します。

サービスが配置後に正常に実行されていると、入力引数として渡されたユーザー名が Oracle E-Business Suite に存在する場合、TESTUSERNAME メソッドによってプラスの数値が非同期的に返されます。ユーザー名が存在しない場合は、かわりに 0 の数値が戻されます。

WSDL URL の検索および記録

前述の必須タスク以外に、統合開発者は、サービスが非同期操作パターンのサポートを使用して生成されるようにする必要があります。これを行うには、統合リポジトリの FND_USER_PKG サービスの TESTUSERNAME 操作について相互作用パターン表で「非同期」相互作用パターン・チェック・ボックスを選択してから、「生成」ボタンをクリックして非同期操作を使用するサービスを生成します。

サービスが正常に生成されたら、サービスを配置してから、設計時に配置済 WSDL URL をコピーして BPEL クライアントを作成する必要があります。

配置された非同期 WSDL URL がある「SOAP Web サービス」タブ

Integration Repository > PLSQL Interface : User

Internal Name: FND_USER_PKG
Type: PL/SQL
Product: Application Object Library
Status: Active
Business Entity: User

Scope: Public
Interface Source: Oracle

Log Configuration: Disabled

Online Help: [See the related online help](#)

Overview SOAP Web Service REST Web Service

SOAP Service Status: Deployed | Active

Service Operations

Retire Undeploy Reset

Display Name	Internal Name	Synchronous	Asynchronous	Grant
User	FND_USER_PKG	☐	☐	
Set Old Person Party Id	SET_OLD_PERSON_PARTY_ID	☐	☐	
Set Old User GUID	SET_OLD_USER_GUID	☐	☐	
Test User Name	TESTUSERNAME	☐	☒	
User Form LDAP Wrapper Update User	FORM_LDAP_WRAPPER_UPDATE_USER	☐	☐	
Validate User Name	VALIDATE_USER_NAME	☐	☐	

TIP To apply any changes in Interaction Pattern, Generate or Regenerate the service.

Web Service Security

* Authentication Type: Username Token (selected), SAML Token (Sender Vouches)

クライアント側の設定タスク

Oracle E-Business Suite Integrated SOA Gateway 非同期サービスには、2 つのポート・タイプがあります。各ポート・タイプは一方方向操作を実行します。

- 1 つのポート・タイプは非同期プロセスを開始します。

このようなポート・タイプの例として、TESTUSERNAME_ASYNCH があります。

- もう一方は非同期応答でクライアントをコールバックします。

このようなポート・タイプの例として、TESTUSERNAME_ASYNCH_RESPONSE があります。

Oracle E-Business Suite Integrated SOA Gateway では、Web サービスとメッセージをアドレスおよび相互関連付けするトランスポート・ニュートラルなメカニズムを提供するために、Web Services Addressing (WS-Addressing) が使用されます。WS-Addressing は公開仕様であり、Oracle E-Business Suite Integrated SOA Gateway でサポートされているデフォルトの関連メ

ソッドです。非同期メッセージの相互関連付けに Simple Object Access Protocol (SOAP) ヘッダーが使用されます。メッセージは、使用されるトランスポートまたはアプリケーションとは無関係です。WS-Addressing の詳細は、<http://www.w3.org/Submission/ws-addressing/> を参照してください。

非同期操作が正しく動作するように、Web サービス・クライアントで次のものを指定する必要があります。

- エンドポイント・ロケーション(`wsa:ReplyTo` ヘッダー内): 返信先アドレスによって、同じコールバック・ポート・タイプを実装するクライアントがコールバック・メッセージをリスニングするロケーションが指定されます。
- 対話 ID (`wsa:MessageID` ヘッダー内): `MessageID` によって、Web サービス・クライアントのメッセージが一意に識別されます。同じ `MessageID` が、Oracle E- Business Suite Integrated SOA Gateway によってコールバック・メッセージ内の `wsa:RelatesTo` ヘッダーで返されます。

非同期サービスのプロセス・フローの確立

非同期サービスがどのように動作するかを十分に理解するために、この章では次の設計時タスクについて説明します。

1. SOA コンポジット・アプリケーションの作成、ページ 3-49

このステップを使用して、非同期 BPEL テンプレートを使用して新規 SOA コンポジット・アプリケーションを作成します。これにより、サード・パーティ・アプリケーションからの入力を受け入れる受入と、要求アプリケーションに BPEL プロセスの出力を返信する返信の 2 つのダミー・アクティビティが自動的に作成されます。

2. パートナ・リンクの追加、ページ 3-52

このステップを使用して、Web サービスとして公開される PL/SQL API `FND_USER_PKG` を使用するパートナ・リンクを作成します。

3. Invoke アクティビティの追加、ページ 3-53

このステップを使用して、サービスを起動するように Invoke アクティビティを構成します。

4. 受入アクティビティの追加、ページ 3-55

このステップを使用して、非同期操作から応答メッセージを受け入れて 2 つ目の Assign アクティビティに渡す、受入アクティビティを構成します。

5. Assign アクティビティの追加、ページ 3-57

このステップを使用して、ユーザー名をクライアントから Invoke アクティビティに入力値として渡すため、および応答メッセージを受入アクティビティから渡すために、2 つの Assign アクティビティを構成します。

応答の詳細は、BPEL プロセスが配置されたサーバーの出力ファイルとして書き込まれます。

BPEL プロセスのサービス・コンポーネントを使用した SOA コンポジット・アプリケーションの一般情報と作成方法の詳細は、*Oracle Fusion Middleware Oracle SOA Suite 開発者ガイド* を参照してください。

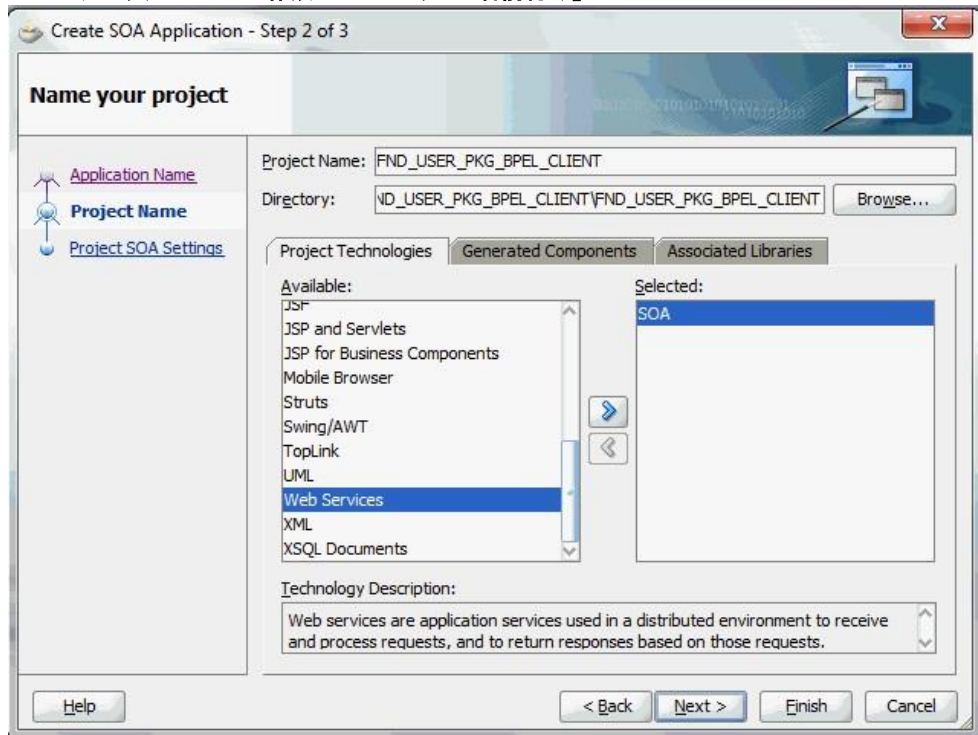
BPEL プロセスによる SOA コンポジット・アプリケーションの作成

このステップを使用して、様々な BPEL プロセス・アクティビティによる新規 SOA コンポジット・アプリケーションを作成します。

BPEL プロジェクトを使用して新規 SOA コンポジット・アプリケーションを作成するには:

1. Oracle JDeveloper を開きます。
2. アプリケーション・ナビゲータで「**新規アプリケーション**」をクリックします。
「SOA アプリケーションの作成 - アプリケーションの名前付け」ページが表示されます。
3. 「アプリケーション名」フィールドにアプリケーションの適切な名前を入力して、「アプリケーション・テンプレート」リストから「**SOA アプリケーション**」を選択します。
「**次へ**」をクリックします。「SOA アプリケーションの作成 - プロジェクトの名前付け」ページが表示されます。
4. 「プロジェクト名」フィールドにプロジェクトの適切な名前を入力します。たとえば、FND_USER_PKG_BPEL_CLIENT です。

「SOA アプリケーションの作成 - プロジェクトの名前付け」ページ



5. 「プロジェクト・テクノロジー」タブで、「Web サービス」を選択して、「使用可能」のテクノロジー・リストから「選択済」のテクノロジー・リストに「SOA」が選択されるようにします。
「次へ」をクリックします。「SOA アプリケーションの作成 - SOA 設定の構成」ページが表示されます。

6. 「コンポジット・テンプレート」リストからコンポジット **BPEL プロセス**を選択して、「終了」をクリックします。

新規アプリケーションと SOA プロジェクトを作成しました。これにより、SOA コンポジットが自動的に作成されます。

「BPEL プロセスの作成」ページが表示されます。

「BPEL プロセスの作成」ページ

7. デフォルトの「**BPEL 1.1 Specification**」の選択を変更せずに維持します。これにより、BPEL 1.1 Specificationをサポートする BPEL プロジェクトが作成されます。

「名前」フィールドに BPEL プロセスの適切な名前を入力します。たとえば、TESTUSERNAME_ASYNC_BPELProcess です。

「テンプレート」フィールドで「同期 BPEL プロセス」を選択します。

「トランザクション」ドロップダウン・リストから「requiresNew」を選択します。「OK」をクリックします。

受入アクティビティと返信アクティビティのある同期 BPEL プロセスが作成されます。指定した名前を使用した、bpel および wsdl を含む必須のソース・ファイル(たとえば、TESTUSERNAME_ASYNC_BPELProcess.bpel、TESTUSERNAME_ASYNC_BPELProcess.wsdl)と composite.xml も生成されます。

8. 「SOA コンテンツ」→「ビジネス・ルール」に移動して、composite.xml をクリックして、コンポジット・ダイアグラムを表示します。

TESTUSERNAME_ASYNC_BPELProcess コンポーネントをダブルクリックして同期 BPEL プロセスを開きます。

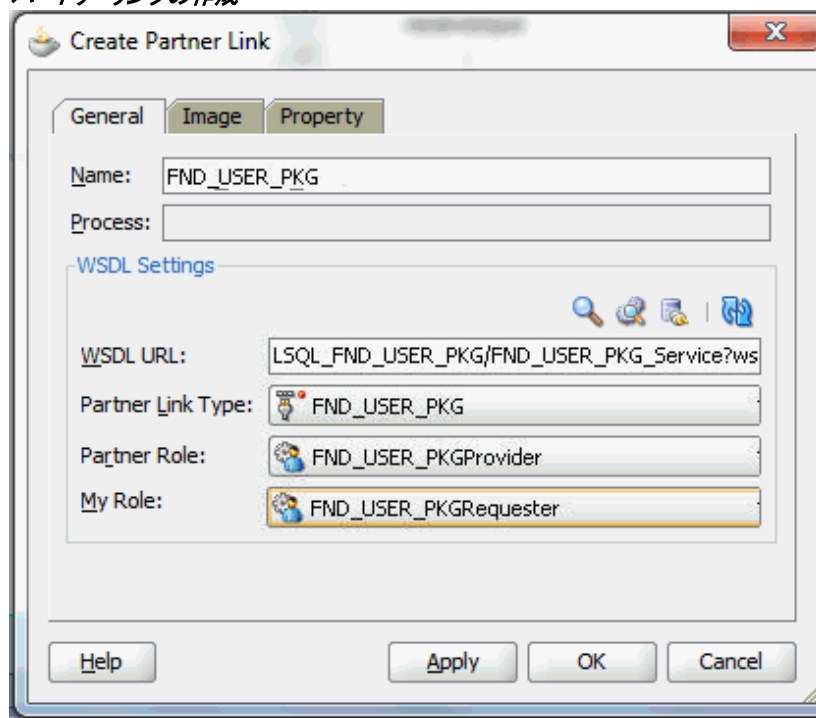
パートナー・リンクの追加

このステップを使用して、非同期操作から受信した応答をファイルに書き込むことによって BPEL プロセスを構成します。

パートナー・リンクを追加するには:

1. Oracle JDeveloper で、「パートナー・リンク」領域にマウスを置いて右クリックして、プルダウン・メニューから「パートナー・リンクの作成」を選択します。または、「BPEL コンストラクト」リストから「パートナー・リンク」を、プロセス・ダイアグラムの右側の「パートナー・リンク」スイムレーンにドラッグ・アンド・ドロップできます。
「パートナー・リンクの作成」ウィンドウが表示されます。
2. 以前に記録した FND_USER_PKG サービスに対応する WSDL URL を統合リポジトリからコピーし、「WSDL ファイル」フィールドに貼り付けます。**[Tab]**キーを押します。
3. パートナリンク名を FND_USER_PKG として入力します。
「WSDL 設定」リージョンで、次の値を選択します。
 - パートナ・リンク・タイプ: FND_USER_PKG
 - パートナ・ロール: FND_USER_PKGProvider
 - マイ・ロール: FND_USER_PKGRequester

パートナ・リンクの作成



「適用」をクリックします。

パートナ・リンクは、必要な WSDL 設定で作成され、BPEL プロジェクトでは、プロセス・ダイアグラムの枠線の領域の新しいアイコンによって表現されます。

4. 「OK」をクリックして、パートナ・リンクの構成を完了します。

パートナ・リンク FND_USER_PKG が BPEL プロセス・ダイアグラムの「パートナ・リンク」セクションに追加されます。

「ファイル」→「すべて保存」を選択して、変更を保存します。

Invoke アクティビティの追加

このステップを使用して、ファイル・アダプタが応答を出力ファイルに書き込むように **Invoke** アクティビティを構成します。

WriteFile パートナ・リンクの **Invoke** アクティビティを追加するには:

1. Oracle JDeveloper で、コンポーネント・パレットから「**BPEL コンストラクト**」を展開します。**Invoke** アクティビティを、プロセス・ダイアグラムの中央のスイムレーンの、**receiveInput** アクティビティの後にドラッグ・アンド・ドロップします。
2. **Invoke** アクティビティを FND_USER_PKG サービスにリンクします。「編集 Invoke」ダイアログ・ボックスが表示されます。

3. **Invoke** アクティビティの名前を入力し、「**入力変数**」フィールドの横にある「**作成**」アイコンをクリックして、新しい変数を作成します。「変数の作成」ダイアログ・ボックスが表示されます。

入力変数名として

TESTUSERNAME_ASYNCH_Invoke_TESTUSERNAME_ASYNCH_InputVariable と入力します。デフォルト名を受け入れることもできます。

「**グローバル変数**」を選択して、「**OK**」をクリックします。

非同期機能を選択したことに注意してください。

Invoke アクティビティの編集

The screenshot shows the 'Edit Invoke' dialog box with the following details:

- General Tab:**
 - Name:** TESTUSERNAME_ASYNCH_Invoke
 - Conversation ID:** (empty)
 - Detail Label:** (empty)
 - Invoke as Detail:** ☐
 - Interaction Type:** Partner Link
 - Partner Role Web Service Interface:**
 - Partner Link:** FND_USER_PKG
 - Operation:** TESTUSERNAME_ASYNCH
 - Variables:**
 - Input:** Invoke1_TESTUSERNAME_ASYNCH_InputVariable
 - Output:** (empty)

4. 「編集 Invoke」ダイアログ・ボックスで「**適用**」→「**OK**」をクリックして、**Invoke** アクティビティの構成を完了します。

プロセス・ダイアグラムに **Invoke** アクティビティが表示されます。

受入アクティビティの追加

このステップを使用して、非同期操作から応答メッセージを受け入れて 2 つ目の Assign アクティビティに渡す、受入アクティビティを構成します。

1. Oracle JDeveloper で、コンポーネント・パレットから「**BPEL コンストラクト**」を展開します。
受入アクティビティを、プロセス・ダイアグラムの中央のスイムレーンの、**Invoke** アクティビティの後にドラッグ・アンド・ドロップします。
2. 受入アクティビティを FND_USER_PKG パートナ・リンクにリンクします。受入アクティビティでは、パートナ・リンクから応答メッセージを受信します。「**Receive の編集**」ダイアログ・ボックスが表示されます。
3. 受入アクティビティの名前を入力します。「操作」ドロップダウン・リストから TESTUSERNAME_ASYNCH_RESPONSE を選択します。
4. 「**変数**」フィールドの横にある「**入力変数の作成**」アイコンをクリックして、新しい変数を作成します。「**変数の作成**」ダイアログ・ボックスが表示されます。
「**グローバル変数**」を選択して、変数の名前を入力します。デフォルト名を受け入れることができます
(TESTUSERNAME_ASYNCH_Receive_TESTUSERNAME_ASYNCH_RESPONSE_InputVariable など)。「**OK**」をクリックして、「**Receive の編集**」ダイアログ・ボックスに戻ります。

受入アクティビティの作成

Edit Receive

Annotations Assertions Skip Condition Timeout Headers

General Correlations Properties

Name: TESTUSERNAME_ASYNCH_Receive

Conversation ID:

☐ Create Instance

Interaction Type: Partner Link

My Role Web Service Interface

Partner Link: FND_USER_PKG

Operation: TESTUSERNAME_ASYNCH_RESPONSE

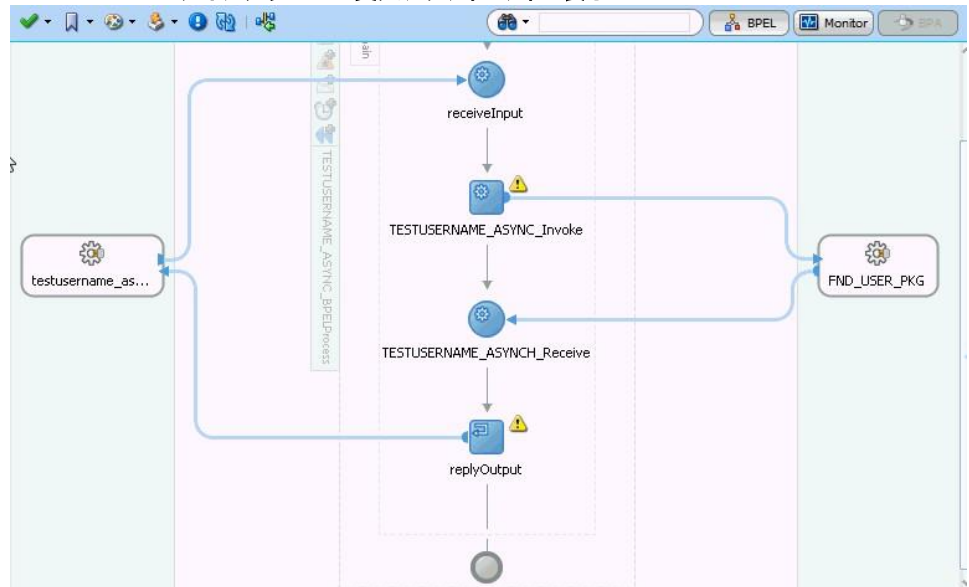
Variable

Variable: 1_TESTUSERNAME_ASYNCH_RESPONSE_InputVariable

Help Apply OK Cancel

5. 「適用」→「OK」をクリックして、受入アクティビティの構成を終了します。BPEL プロセス・ダイアグラムに受入アクティビティが表示されます。

BPEL プロセス・ダイアグラムでの受入アクティビティの表示



Assign アクティビティの追加

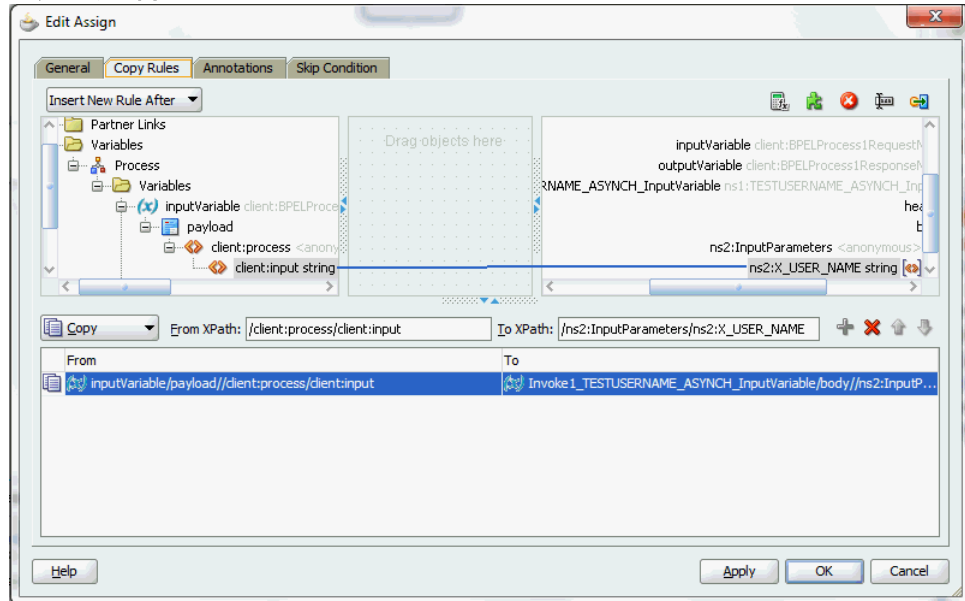
このステップでは、ユーザー名をクライアントから Invoke アクティビティに入力値として渡すため、および応答メッセージを受入アクティビティから渡すために、2つの **Assign** アクティビティを構成します。応答の詳細は、BPEL プロセスが配置されたサーバーの出力ファイルとして書き込まれます。

ユーザー名をクライアントから Invoke アクティビティに入力値として渡す 1 つ目の **Assign** アクティビティを入力するには:

1. **Assign** アクティビティをコンポーネント・パレットの「**BPEL コンストラクト**」から **receiveInput** アクティビティと **Invoke** アクティビティの間にドラッグ・アンド・ドロップすることによって、**Assign** アクティビティを追加します。
2. **Assign** アクティビティをダブルクリックして、「編集 Assign」ダイアログ・ボックスにアクセスします。1 つ目の **Assign** アクティビティの名前として Input_to_Service と入力します。
3. 「コピー・ルール」タブを選択して、ソース・ツリーとターゲット・ツリーを展開します。
 - 「元」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「InputVariable」→「payload」→「client:process」→「client:input」にナビゲートします。「送信元 XPath」フィールドも表示されます。
 - 「先」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「TESTUSERNAME_ASYNC_Invoke_TESTUSERNAME_ASYNC_InputVariable」→「body」→「ns2:InputParameter」にナビゲートし、「ns2:X_USER_NAME」を選択します。「送信先 XPath」フィールドも表示されます。

4. ソース・ノード(client:input)をドラッグして、前のステップで識別したばかりのターゲット・ノード(ns2: X_USER_NAME)を指します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

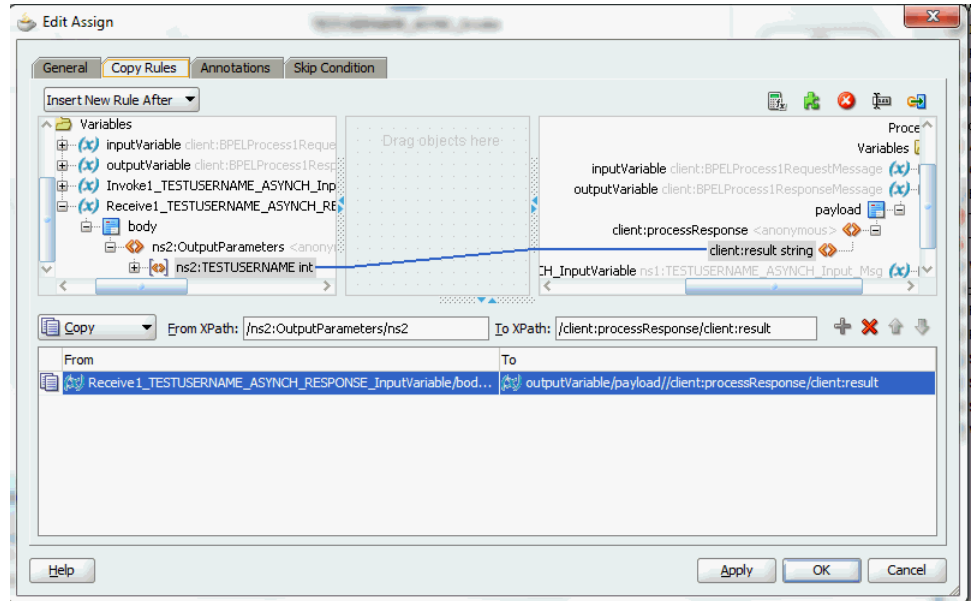
パラメータの割当て



5. 「適用」→「OK」をクリックして、1 つ目の **Assign** アクティビティの構成を完了します。
受入アクティビティからの応答メッセージを渡す 2 つ目の **Assign** アクティビティを入力するには:
 1. **Assign** アクティビティをコンポーネント・パレットの「**BPEL コンストラクト**」から受入アクティビティと **replyOut** アクティビティの間にドラッグ・アンド・ドロップすることによって、2 つ目の **Assign** アクティビティを追加します。
 2. **Assign** アクティビティをダブルクリックして、「編集 Assign」ダイアログ・ボックスにアクセスします。2 つ目の **Assign** アクティビティの名前として **Service_Response** と入力します。
 3. 「コピー・ルール」タブを選択して、ソース・ツリーとターゲット・ツリーを展開します。
 - 「元」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「**TESTUSERNAME_ASYNC_Receive_TESTUSERNAME_ASYNC_Input Variable**」→「body」→「**ns2:OutputParameters**」にナビゲートし、「**ns2:TESTUSERNAME**」を選択します。「送信元 XPath」フィールドも表示されます。

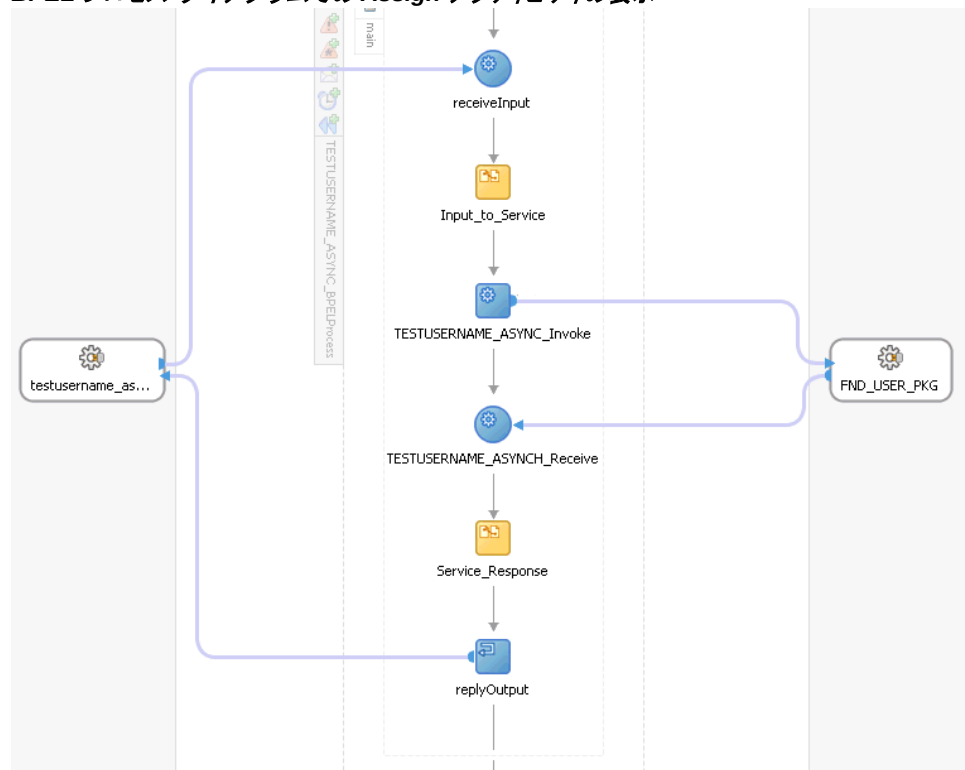
- 「先」のナビゲーション・ツリーで、タイプ「変数」を選択し、「変数」→「プロセス」→「変数」→「OutputVariable」→「payload」→「client: processResponse」にナビゲートし、「client:result」を選択します。「送信先 XPath」フィールドも表示されます。
4. ソース・ノード(ns2:TESTUSERNAME)をドラッグして、前のステップで識別したばかりのターゲット・ノード(client: result)を指します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

パラメータの割当て



5. 「適用」→「OK」をクリックして、2つ目の **Assign** アクティビティの構成を完了します。
- BPEL プロセス・ダイアグラムに **Assign** アクティビティが表示されます。

BPEL プロセス・ダイアグラムでの Assign アクティビティの表示

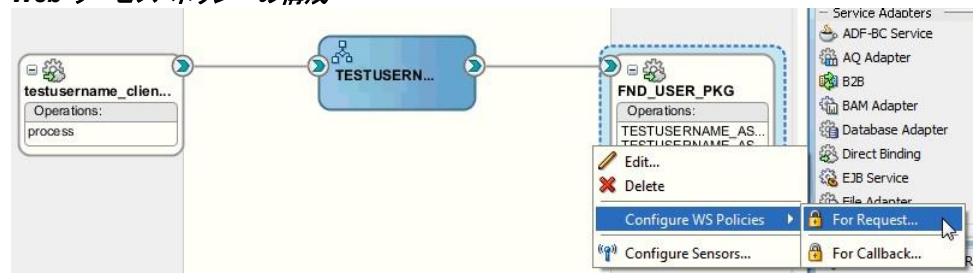


Web サービス・ポリシーの構成

次のステップを使用して、設計時のセキュリティ・ポリシーを追加します。

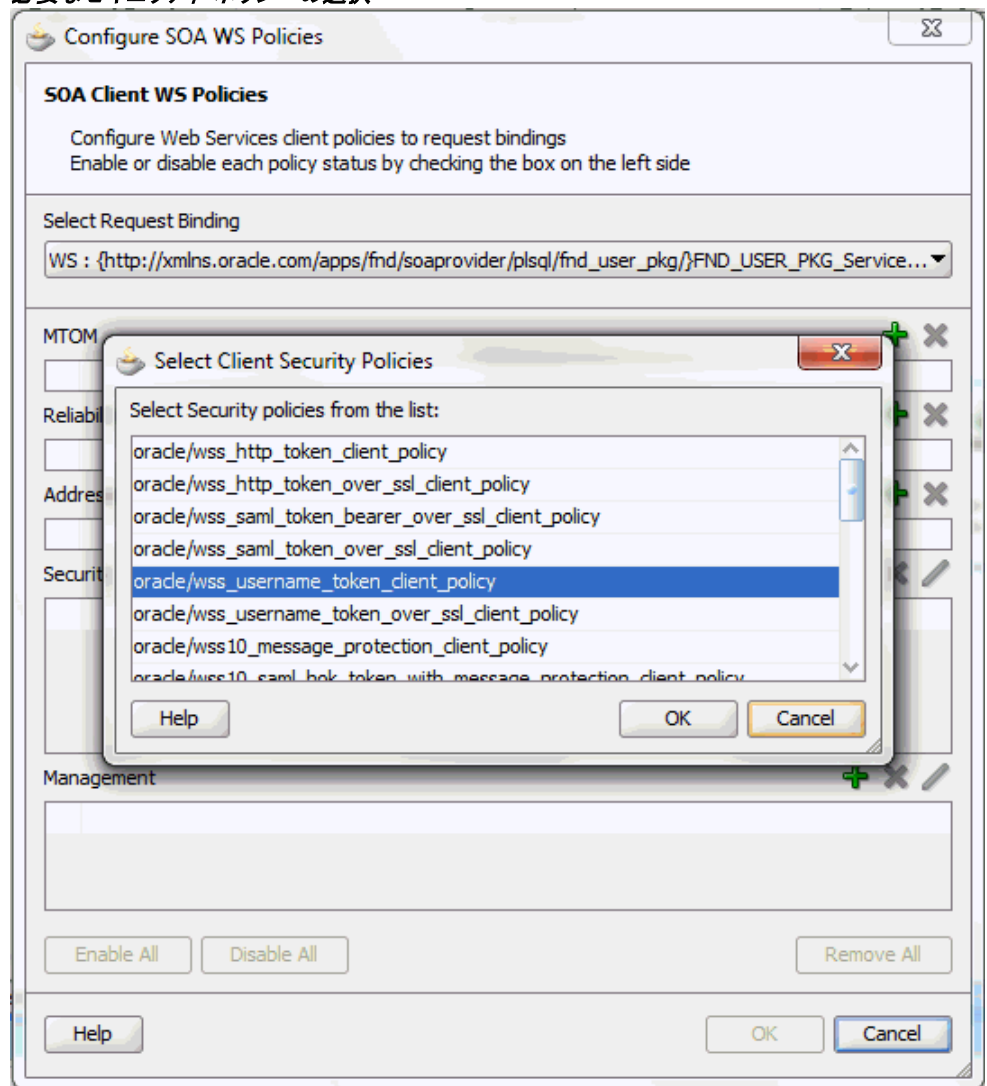
1. 「SOA コンテンツ」→composite.xml に移動します。FND_USER_PKG サービスを右クリックし、ドロップダウン・リストから「WS ポリシーの構成」→「リクエスト対象」を選択します。

Web サービス・ポリシーの構成



2. 「SOA WS ポリシーの構成」ダイアログが表示されます。
「セキュリティ」セクションで「セキュリティ・ポリシーの追加」アイコン(+)をクリックします。
サーバー・セキュリティ・ポリシーの選択ダイアログが表示されます。

必要なセキュリティ・ポリシーの選択

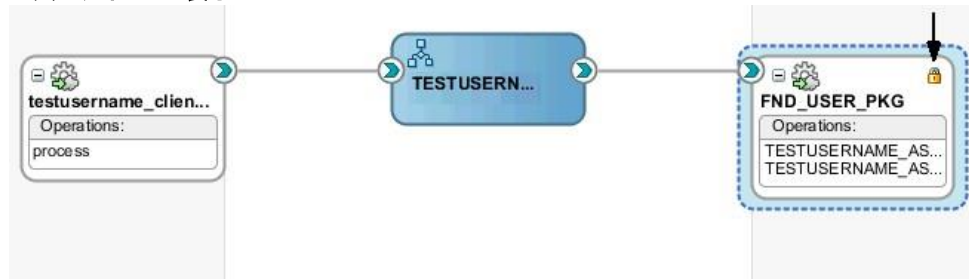


「oracle/wss_username_token_service_policy」を選択して、「OK」をクリックします。

添付されたセキュリティ・ポリシーが「セキュリティ」セクションに表示されます。

composite.xml の FND_USER_PKG サービスにロック・アイコンが表示されます(これは、セキュリティ・ポリシーが正常に添付されたことを示します)。

ロック・アイコンの表示



3. FND_USER_PKG サービスを選択し、ナビゲーション・メニューから「表示」→「プロパティ・インスペクタ」を選択することによって参照バインディングを追加します。これにより、FND_USER_PKG サービス・コンポーネントの「プロパティ・インスペクタ」ウィンドウが表示されます。

注意: これは、FND_USER_PKG サービスの composite.xml ファイルの「ソース」タブで、次の参照バインディングを直接追加することで設定できます。

```
<property name="oracle.webservices.auth.username"
type="xs:string"
many="false" override="may"
>operations</property>
<property name="oracle.webservices.auth.password"
type="xs:string"
many="false" override="may">password
</property>
```

password は、ユーザー名 *operations* の実際のパスワードに置き換えます。

「プロパティ」セクションで、バインディング・プロパティの「セキュリティ・ポリシーの追加」アイコン(+)をクリックします。「プロパティの作成」ダイアログが表示されます。

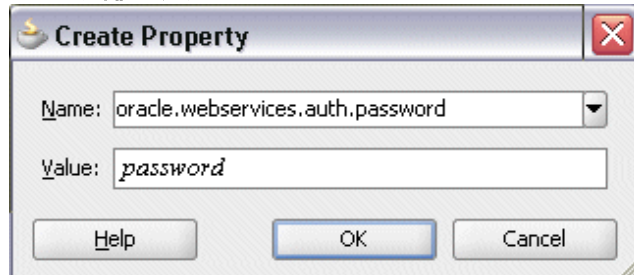
「名前」フィールドに `oracle.webservices.auth.username` と入力して、値として `operations` と入力します。

プロパティ値の入力

「OK」をクリックします。

4. 「プロパティ」セクションのバインディング・プロパティで「セキュリティ・ポリシーの追加」アイコン(+)を再度クリックする、同じアプローチを使用します。「名前」フィールドに `oracle.webservices.auth.password` と入力します。「値」フィールドに、operations ユーザーに紐づくパスワードを入力します。

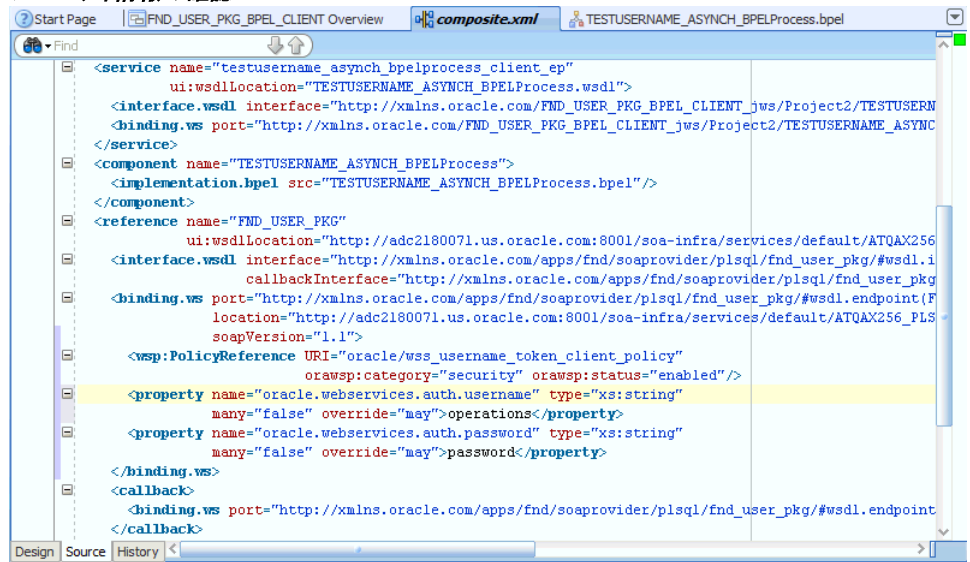
プロパティ値の入力

A screenshot of the 'Create Property' dialog box. It has a title bar with a close button. Inside, there are two text input fields: 'Name' and 'Value'. The 'Name' field contains 'oracle.webservices.auth.password' and the 'Value' field contains 'password'. At the bottom, there are three buttons: 'Help', 'OK', and 'Cancel'.

「OK」をクリックします。

「プロパティ」セクションに、選択したプロパティの名前と値の両方が表示されます。composite.xml の「ソース」タブをクリックして、`oracle.webservices.auth.username` および `oracle.webservices.auth.password` プロパティ名および関連する値が FND_USER_PKG 参照に追加されることに注目してください。

プロパティ情報の確認



非同期 BPEL プロセスによる SOA コンポジットのデプロイおよびテスト

SOA コンポジットに含まれる BPEL クライアントから非同期 TESTUSERNAME メソッドを起動するには、SOA コンポジットが Oracle WebLogic 管理対象サーバーにデプロイされている

必要があります。これは、Oracle JDeveloper を使用して達成できます。コンポジットがデプロイされたら、Oracle Enterprise Manager Fusion Middleware Control コンソールからテストできます。

前提条件

Oracle JDeveloper を使用して BPEL プロセスによる SOA コンポジットをデプロイする前に、設計時環境とランタイム・サーバーの間の接続を確立する必要があります。必要なサーバー接続を構成する方法の詳細は、「サーバー接続の構成」(ページ B-1)を参照してください。

注意: WebLogic サーバーのローカル・インスタンスが使用される場合、Oracle JDeveloper から「実行」→「サーバー・インスタンスの起動」を選択して、WebLogic サーバーを起動します。WebLogic 管理サーバーの DefaultServer インスタンスが正常に起動されたら、「実行中:DefaultServer」と「メッセージ」のログに<Server started in Running mode>と DefaultServer started のメッセージが表示されるはずです。

次の実行時タスクを実行します。

1. BPEL プロセスによる SOA コンポジット・アプリケーションのデプロイ、ページ 3-64
2. BPEL プロセスによる SOA コンポジット・アプリケーションのテスト、ページ 3-69

BPEL プロセスによる SOA コンポジットのデプロイ

先ほど作成した BPEL プロセスによる SOA コンポジットは、実行する前にデプロイする必要があります。

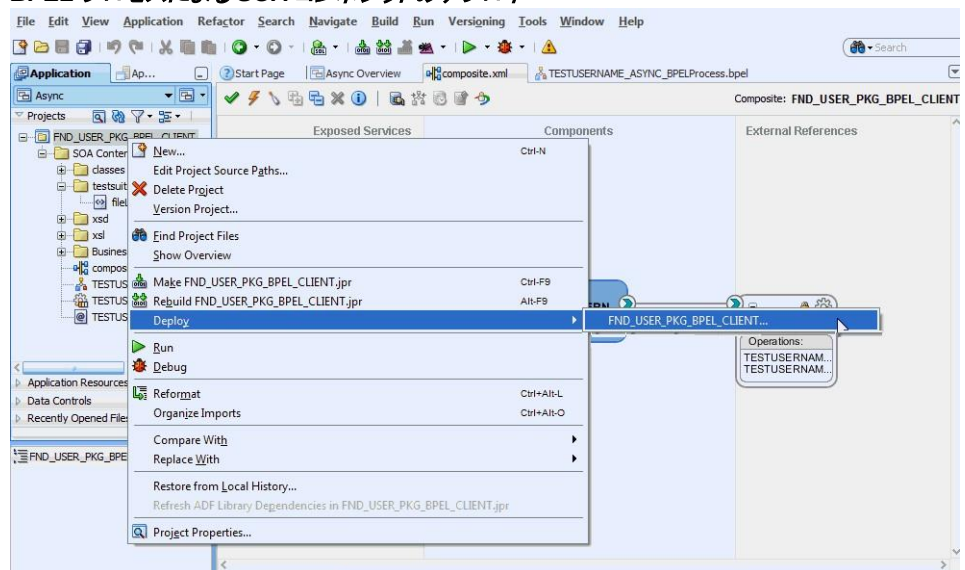
SOA コンポジット・アプリケーションをデプロイするには:

1. JDeveloper のアプリケーション・ナビゲータで、**FND_USER_PKG_BPEL_CLIENT** プロジェクトを選択します。
2. プロジェクトを右クリックして、メニューから「デプロイ」→「[project name]」→「[serverConnection]」を選択します。

たとえば、接続を適切に設定している場合は、「デプロイ」→

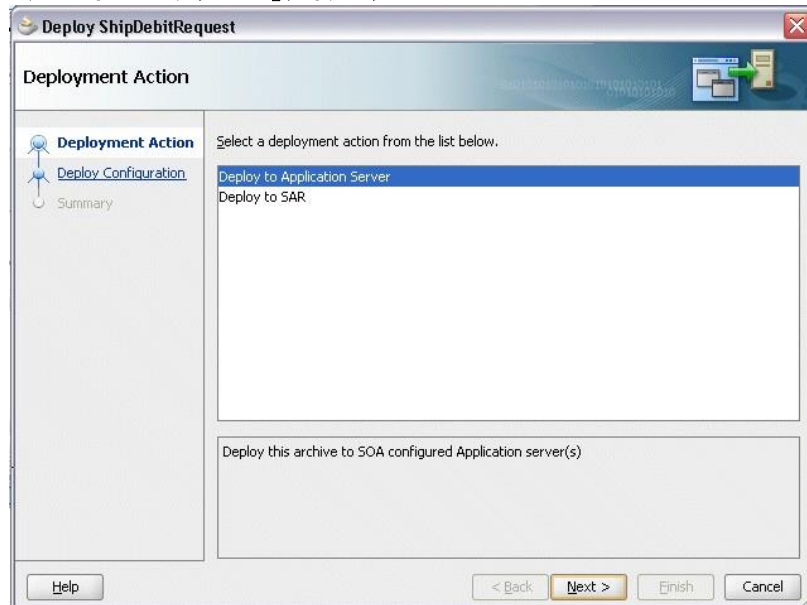
「FND_USER_PKG_BPEL_CLIENT」→「SOAServer」を選択してプロセスをデプロイできます。

BPEL プロセスによる SOA コンポジットのデプロイ



注意: サーバー接続を初めて設定する場合は、「デプロイメント・アクション」ダイアログが表示されます。「アプリケーション・サーバーにデプロイ」を選択して、「次へ」をクリックします。

「デプロイメント・アクション」ダイアログ



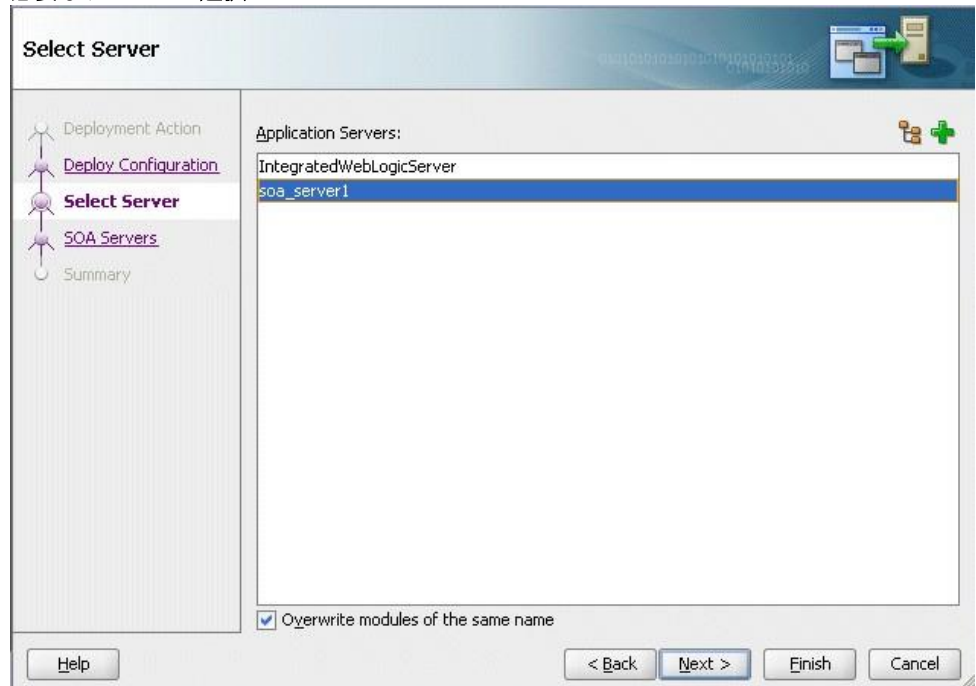
「構成のデプロイ」ダイアログで、「次へ」をクリックして新しいアプリケーション・サーバーを追加する前に、次の情報が選択されるようにします。

- 新規リビジョン ID: 1.0
- コンポジット・リビジョンをデフォルトとしてマークします。: このチェック・ボックスを選択します。
- 同じリビジョン ID で既存のコンポジットを上書きします。: このチェック・ボックスを選択します。

JDeveloper から新しい Oracle WebLogic Server 接続を作成するステップは、「サーバー接続の構成」(ページ B- 1)を参照してください。

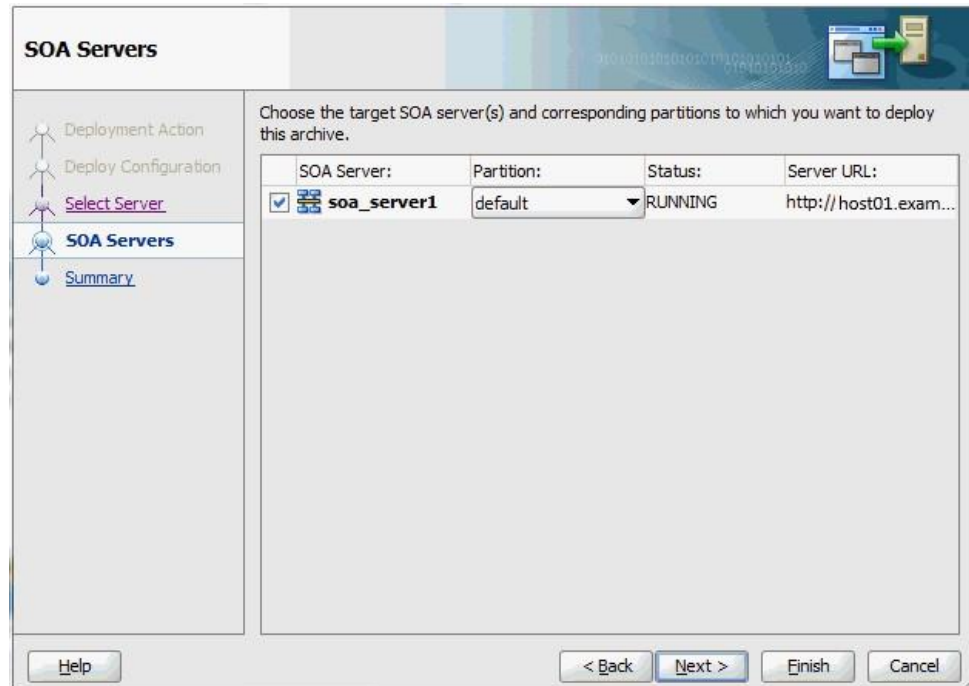
3. 「サーバーの選択」ダイアログで、以前にサーバー接続を確立した soa-server1 を選択します。「次へ」をクリックします。

必要なサーバーの選択



4. 「SOA サーバー」ダイアログで、デフォルトのターゲット SOA サーバー(soa-server1)の選択を受け入れます。

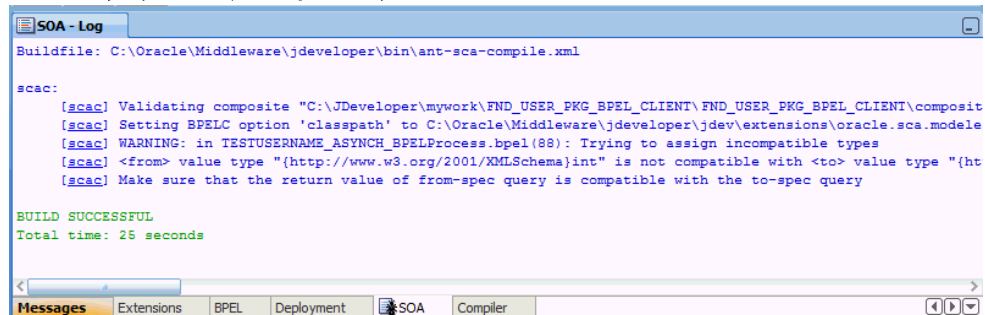
必要な SOA サーバーの選択



「次へ」→「終了」をクリックします。

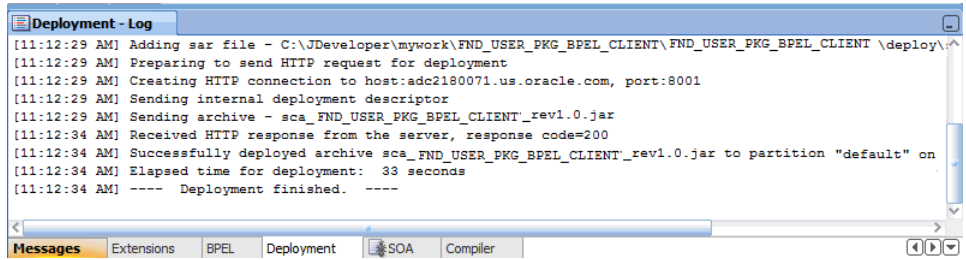
5. Oracle JDeveloper セッションから初めてコンポジットをデプロイする場合は、「認証リクエスト」ウィンドウが表示されます。Oracle SOA Suite のインストール時に指定したユーザー名とパスワード情報を入力します。「OK」をクリックします。
6. デプロイ処理が開始します。SOA - ログ・ウィンドウで、デプロイ・プロセスをモニターして、コンパイルが成功したことを確認します。

SOA - ログ・タブでのデプロイのモニター



デプロイメント - ログ・ウィンドウで、デプロイが成功したことを確認します。

デプロイ情報の表示



```
[11:12:29 AM] Adding sar file - C:\JDeveloper\mywork\FND_USER_PKG_BPTEL_CLIENT\FND_USER_PKG_BPTEL_CLIENT \deploy\
[11:12:29 AM] Preparing to send HTTP request for deployment
[11:12:29 AM] Creating HTTP connection to host:adc2180071.us.oracle.com, port:8001
[11:12:29 AM] Sending internal deployment descriptor
[11:12:29 AM] Sending archive - sca_FND_USER_PKG_BPTEL_CLIENT_rev1.0.jar
[11:12:34 AM] Received HTTP response from the server, response code=200
[11:12:34 AM] Successfully deployed archive sca_FND_USER_PKG_BPTEL_CLIENT_rev1.0.jar to partition "default" on
[11:12:34 AM] Elapsed time for deployment: 33 seconds
[11:12:34 AM] ---- Deployment finished. ----
```

SOA コンポジット・アプリケーションのテスト

SOA コンポジット・アプリケーションに含まれる BPEL プロセスが正常にデプロイされたら、Oracle Enterprise Manager Fusion Middleware Control コンソールからプロセスを管理およびモニターできます。

Oracle SOA Suite の詳細は、*Oracle Fusion Middleware Oracle SOA Suite 管理者ガイド*を参照してください。

コンソールで非同期操作をテストするには:

1. インストール時に指定したユーザー名およびパスワード情報を使用して、Oracle Enterprise Manager Fusion Middleware Control コンソール (<http://<hostname>:<port>/em>)にログインします。
2. 「ファーム」のナビゲーション・ペインから、ツリーの「SOA」→「soa-infra」ノードを展開して、「SOA インフラストラクチャ」のホーム・ページおよびメニューを使用して、soa-infra 管理対象サーバーで実行されているデプロイ済の SOA コンポジット・アプリケーションにアクセスします。

FND_USER_PKG_BPTEL_CLIENT [1.0]リンクをクリックします。

3. FND_USER_PKG_BPTEL_CLIENT [1.0]のホーム・ページで、「テスト」をクリックします。
4. インスタンスを開始するための「Web サービスのテスト」ページが表示されます。「引数の入力」セクションで、ユーザー名 operations を XML ペイロード入力データとして入力します。

「Web サービスのテスト」をクリックして、プロセスを開始します。

Web サービスのテスト

↑ FND_USER_PKG_BPEL_CLIENT [1.0] ⓘ Logged in as weblogic | Host adc2180071.us.oracle.com
SOA Composite ▼ Page Refreshed May 3, 2012 4:00:29 PM PDT

Test Web Service ⓘ Test Web Service

Use this page to test any WSDL, including WSDLs that are not in the farm. To test a Web service, enter the WSDL and click Parse WSDL. When the page refreshes with the WSDL details, first select the Service, then select the Port, and then select the Operation that you want to test. Specify any input parameters, and click Test Web Service.

WSDL Parse WSDL

HTTP Basic Auth Option for WSDL Access

Service testusername_async_bpelprocess_client_ep
Port TESTUSERNAME_ASYNC_BPELProcess_pt
Operation process ▼

Endpoint URL Edit Endpoint URL

Request **Response**

[-] Security
[-] Quality of Service
[-] HTTP Transport Options
[-] Additional Test Options
[-] Input Arguments

Tree View ▼

Name	Type	Value
[-] * payload	payload	
* input	string	operations

5. サービス起動が成功し、「レスポンス」タブに応答が表示されます。

応答の確認

Request **Response**

Test Status Request successfully received.
Response Time (ms) 3428
Tree View ▼

A new composite instance was generated. Launch Flow Trace

Name	Type	Value
[-] payload	payload	
result	string	2

引数として渡された入力ユーザー名が Oracle E-Business Suite に存在する場合、TESTUSERNAME 操作によってプラスの数値が非同期的に返されることに注意してください。ユーザー名が存在しない場合は、かわりに 0 の数値が戻されます。

- 「レスポンス」タブで「フロー・トレースの起動」をクリックして、「フローのトレース」ページを開きます。

「フローのトレース」ページでの起動詳細の確認

Flow Trace
This page shows the flow of the message through various composite and component instances.

ECID: 9a4b704bb5498b61-7fb6e65c1370d9d698e-8000-000000000025837
Started: May 3, 2012 3:02:29 PM

Faults (0)
Select a fault to locate it in the trace view.
Error Message Recovery Fault Time Fault Location Composite Instance
No faults found

Sensors (0)

Trace
Click a component instance to see its detailed audit trail.
Show Instance IDs

Instance	Type	Usage	State	Time	Composite Instance
testusername_async_bpelprocess_client_ep	Web Service	Service	Completed	May 3, 2012 3:02:29 PM	FND_USER_PKG_BPEL_CLIENT of 20128
TESTUSERNAME_ASYNC_BPELProcess	BPEL Component		Completed	May 3, 2012 3:02:32 PM	FND_USER_PKG_BPEL_CLIENT of 20128
FND_USER_PKG	Web Service	Reference	Completed	May 3, 2012 3:02:29 PM	FND_USER_PKG_BPEL_CLIENT of 20128
FND_USER_PKG_Service	Web Service	Service	Completed	May 3, 2012 3:02:29 PM	ATQAX256_PLSQL_FND_USER_PKG of 20129
Mediator	Mediator Component		Completed	May 3, 2012 3:02:32 PM	ATQAX256_PLSQL_FND_USER_PKG of 20129
TESTUSERNAME_AYSNCH	BPEL Component		Completed	May 3, 2012 3:02:32 PM	ATQAX256_PLSQL_FND_USER_PKG of 20129
TESTUSERNAME_AYSNCH	JCA Adapter	Reference	Completed	May 3, 2012 3:02:32 PM	ATQAX256_PLSQL_FND_USER_PKG of 20129

- 「トレース」セクションで、FND_USER_PKG および FND_USER_PKG_Service サービスとともに testusername_async_bpelprocess_client_ep サービスに含まれるデプロイ済 TESTUSERNAME_ASYNC_BPELProcess BPEL コンポーネントが、起動が正常に実行されたことを示す「完了」状態であることを確認します。
「フォルト」セクションで、テスト中にエラーが発生したかどうかを確認することもできます。
- 「フローのトレース」ページで TESTUSERNAME_AYSNCH リンクをクリックして、「監査証跡」タブでインスタンス詳細を表示できます。

コンポーネント・インスタンスの監査証跡の表示

Flow Trace > Instance of TESTUSERNAME_ASYNCH

Instance of TESTUSERNAME_ASYNCH ⓘ
This page shows BPEL process instance details. ⓘ

Audit Trail | Flow | Sensor Values | Faults

Expand a payload node to view the details.

May 3, 2012 4:02:32 PM	Received "process" call from partner "TESTUSERNAME_ASYNCH"
View XML Document	
Transform_1	
May 3, 2012 4:02:32 PM	Updated variable "Invoke_1_TESTUSERNAMEASYNCH_InputVariable"
May 3, 2012 4:02:32 PM	Completed assign
Invoke_1	
May 3, 2012 4:02:32 PM	Started invocation of operation "TESTUSERNAMEASYNCH" on partner "TESTUSERNAMEASYNCH".
May 3, 2012 4:02:32 PM	Sending property "jca.apps.Username", value is "operations".
May 3, 2012 4:02:32 PM	Sending property "jca.apps.NLSLanguage", value is "".
May 3, 2012 4:02:32 PM	Sending property "jca.apps.SecurityGroup", value is "".
May 3, 2012 4:02:32 PM	Sending property "jca.apps.Responsibility", value is "".
May 3, 2012 4:02:32 PM	Sending property "jca.apps.RespApplication", value is "".
May 3, 2012 4:02:32 PM	Sending property "jca.apps.ORG_ID", value is "".
May 3, 2012 4:02:34 PM	Invoked 2-way operation "TESTUSERNAMEASYNCH" on partner "TESTUSERNAMEASYNCH".
<payload>	
<pre> <messages> <Invoke_1_TESTUSERNAMEASYNCH_InputVariable> <part name="InputParameters"> <InputParameters> <X_USER_NAME>operations</X_USER_NAME> </InputParameters> </part> </Invoke_1_TESTUSERNAMEASYNCH_InputVariable> <Invoke_1_TESTUSERNAMEASYNCH_OutputVariable> <part name="OutputParameters"> <OutputParameters> <TESTUSERNAME>2</TESTUSERNAME> </OutputParameters> </part> </Invoke_1_TESTUSERNAMEASYNCH_OutputVariable> </messages> </pre>	
Transform_2	
May 3, 2012 4:02:34 PM	Updated variable "outputVariable"
May 3, 2012 4:02:34 PM	Completed assign
callbackClient	
May 3, 2012 4:02:34 PM	Started invocation of operation "processResponse" on partner "TESTUSERNAME_ASYNCH".
May 3, 2012 4:02:34 PM	Invoked 1-way operation "processResponse" on partner "TESTUSERNAME_ASYNCH".
<payload>	
<pre> <outputVariable> <part name="payload"> <OutputParameters> <TESTUSERNAME>2</TESTUSERNAME> </OutputParameters> </part> </outputVariable> </pre>	
May 3, 2012 4:02:34 PM	BPEL process instance "20131" completed

PL/SQL REST サービスの使用

Oracle E-Business Suite Integrated SOA Gateway によって提供される REST サービスは、Oracle E-Business Suite でリソースを作成または更新するために使用できます。

すべての REST サービスは HTTP Basic 認証またはトークン・ベース認証によって保護されているため、これらのセキュリティ・メソッドが REST サービスの起動とどのように連携するかを十分に理解するために、この項では次の REST サービスの起動例について説明します。

- HTTP Basic 認証および REST ヘッダー付き XML ペイロードを使用した REST サービスの起動、ページ 3-73
- トークン・ベース認証および JSON ペイロードを使用した REST サービスの起動、ページ 3-83

HTTP Basic 認証および REST ヘッダー付き XML ペイロードを使用した REST サービスの起動

REST サービスの起動シナリオ

REST サービスの起動について説明する例として、PL/SQL API の Profile Management APIs (FND_PROFILE)を検討します。

特定のユーザー・プロファイル・オプションの現在の値を取得するための要求が受信されると、この API に含まれる Get Profile REST サービス操作を起動するために Java クライアントが使用されます。この例では、要求によってユーザー名およびパスワード情報が HTTP ヘッダーで提供され、ユーザー資格証明が認証および承認されます。検証後、認証されたユーザーに対して Get Profile サービス操作を起動できます。

サービス起動の成功後、クライアントは要求のプロファイル値が含まれた REST 応答メッセージを受信します。プロファイルが存在しない場合は、null が返されます。

PL/SQL REST サービスを使用するための前提条件

PL/SQL REST サービスを起動する前に、次のタスクが実行されるようにします。

HTTP 要求の RESTHeader の変数の設定

アプリケーション・コンテキストの値を必要とする REST サービスの起動前に、これらの値を RESTHeader 要素で渡すことができます。

PL/SQL インタフェース・タイプでのこれらのコンテキスト要素は、*Responsibility*、*RespApplication*、*SecurityGroup*、*NLSLanguage* および *Org_Id* です。

これらの要素の予期される値を次の表に示します。

PL/SQL インタフェース・タイプのヘッダー変数と予期される値

要素名	予期される値
Responsibility	responsibility_key (SYSTEM_ADMINISTRATOR など)

要素名	予期される値
RespApplication	アプリケーション短縮名(SYSADMIN など)
SecurityGroup	セキュリティ・グループ・キー(STANDARD など)
NLSLanguage	NLS 言語(AMERICAN など)
Org_Id	組織 ID (202 など)

注意: NLS 言語と組織 ID は、オプションで渡される値です。

- NLS 言語の要素が指定された場合、渡された言語で REST 要求を使用できます。すべての対応する REST 応答とエラー・メッセージも、同じ言語で戻すことができます。言語が指定されていない場合、ユーザーのデフォルトの言語が使用されます。
- サービスの実行が特定組織に依存する場合は、その REST 要求の Org_Id 要素を渡す必要があります。

Java を使用した REST サービスの起動

この項には、REST サービスの起動シナリオに基づく次のタスクが含まれています。

1. PL/SQL REST Web サービスのデプロイ、ページ 3-74
2. 配置済 WADL からのリソース情報の記録、ページ 3-76
3. Java クラスでのプロジェクトの作成、ページ 3-77
4. Java クラスを使用した REST サービスの起動、ページ 3-83

PL/SQL REST Web サービスの配置

次のステップを使用して、Profile Management APIs (FND_PROFILE)を配置します。

1. 「統合管理者」ロールを持つユーザーとして、Oracle E-Business Suite にログインします。ナビゲーション・メニューから、「統合 SOA ゲートウェイ」の職責を選択して、「統合リポジトリ」リンクを選択します。
2. 「統合リポジトリ」タブで、「検索」をクリックしてメインの「検索」ページにアクセスします。

3. 「内部名」フィールドに FND_PROFILE と入力します。「進む」をクリックして、検索を実行します。

Profile Management APIs のインタフェース名リンクをクリックして、インタフェースの詳細ページを開きます。

4. 「REST Web サービス」タブで、次の情報を入力します。

REST サービスとしての PL/SQL API の配置

The screenshot shows the 'Administration' tab in the 'Integration Repository' for 'Profile Management APIs'. The 'REST Web Service' tab is selected. The 'Service Alias' is 'FndProfileSvc' and the 'REST Service Status' is 'Not Deployed'. Under 'Service Operations', a table lists three operations: 'Get Profile' (selected), 'Get Profile Value', and 'Put Profile'. The 'Get Profile' operation is associated with the 'GET' method and the 'Internal Name' 'GET'. Below the table, there is a 'REST Service Security' section with a 'Deploy' button.

Select	Name	Internal Name	Grant
☒	Get Profile	GET	
☐	Get Profile Value	VALUE	
☐	Put Profile	PUT	

- サービス別名: FndProfileSvc
選択したメソッドまたは操作(この例では、Get Profile)の WADL およびスキーマのサービス・エンドポイントとして別名が表示されます。
 - 「サービス操作」リージョンで、「Get Profile」サービス操作を選択します。選択されたメソッドは、REST サービス操作として公開されます。
5. 「配置」をクリックして、Oracle E-Business Suite WebLogic 環境にサービスを配置します。

REST サービスが正常に配置されたら、「WADL の表示」リンクとともに、「REST サービス・ステータス」フィールドに「配置済」が表示されます。「WADL の表示」リンクをクリックして、配置されたサービス WADL の記述を表示します。

REST サービスの配置の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプレメンテーション・ガイド*の「REST Web サービスの配置」を参照してください。

配置済 WADL からのリソース情報の記録

FND_PROFILE サービスの配置済 WADL からサービス・リソース情報を取得するために、統合開発者は「REST Web サービス」タブの「**WADL の表示**」リンクをクリックします。

WADL の記述の表示

Overview SOAP Web Service **REST Web Service**

Service Alias: FndProfileSvc
REST Service Status: Deployed [View WADL](#)
Verb: POST

Service Operations

Name ▲	Internal Name ▲	Included Operations	Grant
Get Profile	GET	✓	
Get Profile Value	VALUE		
Put Profile	PUT		

TIP To apply any changes in Operation, Undeploy the service.

REST Service Security

REST Web Service is secured by HTTP Basic Authentication at HTTP Transport level. Send either of the following in "Authorization" header as per HTTP Basic scheme:

- Username:Password
- Security Token.

Tip: Use [Login Service](#) to obtain Security Token for given user credentials.

次の WADL 記述が表示されます。

```

<xml version="1.0" encoding="UTF-8" standalone="no" ?>
<application xmlns:tns="http://xmlns.oracle.
com/apps/fnd/soapprovider/plsql/rest/fnd_profile/" xmlns="http://wadl.
dev.java.net/2009/02" xmlns:tns1="http://xmlns.oracle.
com/apps/fnd/rest/fndprofilesvc/get/" name="FND_PROFILE"
targetNamespace="http://xmlns.oracle.
com/apps/fnd/soapprovider/plsql/rest/fnd_profile/">
<grammars>
  <include xmlns="http://www.w3.org/2001/XMLSchema" href="https:
//<hostname>:<port>/webservicess/rest/FndProfileSvc/?
XSD=GET_SYNCH_TYPEDEF.xsd" />
</grammars><resources base="http://<hostname>:
<port>/webservicess/rest/FndProfileSvc/"><resource path="/get/">
  <method id="GET" name="POST">
    <request>
      <representation mediaType="application/xml" type="tns1:
InputParameters" />
      <representation mediaType="application/json" type="tns1:
InputParameters" />
    </request>
    <response>
      <representation mediaType="application/xml" type="tns1:
OutputParameters" />
      <representation mediaType="application/json" type="tns1:
OutputParameters" />
    </response>
  </method>
</resource>
</resources>
</application>

```

後で Java クライアントを定義するときに使用される次の情報をコピーまたは記録します。

- `<resources`
`base>="http://<hostname>:<port>/webservicess/rest/FndProfile`
`Svc/">`
この情報
`http://<hostname>:<port>/webservicess/rest/FndProfileSvc` は、後
で Java クライアント・プログラムでベース URL として使用されます。
- `<resource path>="/get/">`
この情報 `/get/` は、後でサービス URL の後ろの部分を作成するために使用されます。

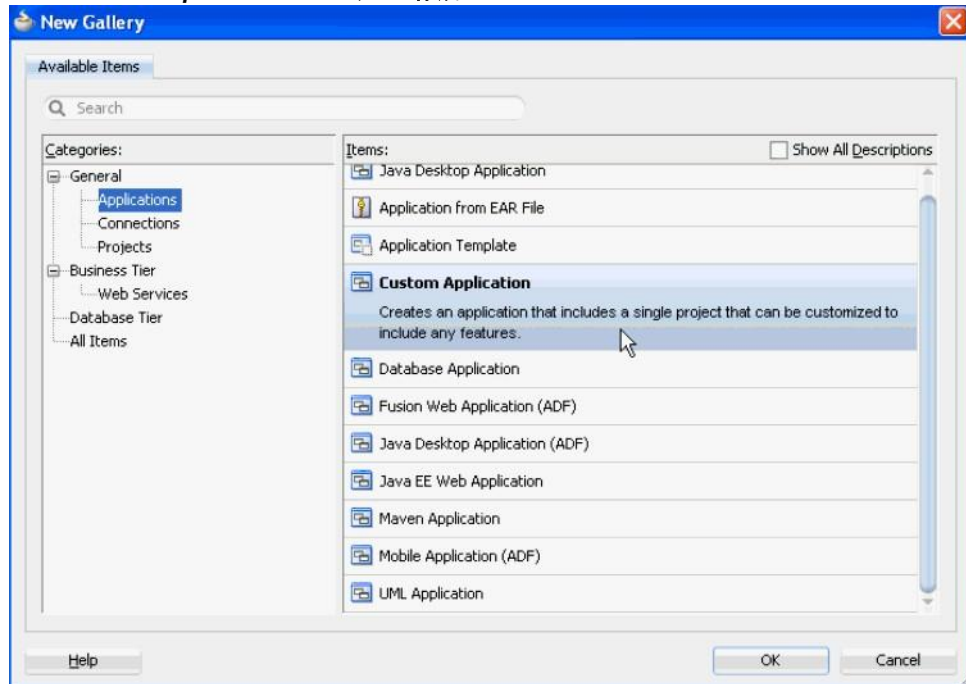
Java クラスを含むプロジェクトの作成

この項では、FND_PROFILE REST サービスの起動に使用される Java クラスを含むプロジェクトを作成する方法について説明します。

プロジェクトと Java クラスを作成するには:

1. Oracle JDeveloper を開きます。
2. メイン・メニューから「ファイル」→「新規」を選択します。
「新規ギャラリー」ウィンドウで、「一般」カテゴリを展開して「アプリケーション」を選択しま
す。「アイテム」リストで「カスタム・アプリケーション」を選択します。

Oracle JDeveloper でのプロジェクトの作成



「OK」をクリックします。「カスタム・アプリケーションの作成 - アプリケーションの名前付け」ページが表示されます。

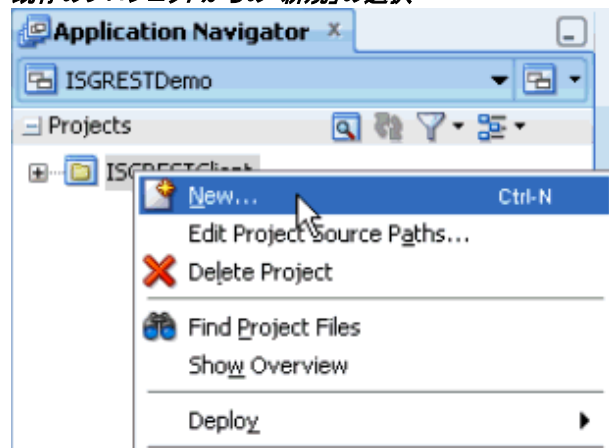
3. 「アプリケーション名」フィールドにアプリケーションの適切な名前を入力します。「次へ」をクリックします。
4. 「カスタム・アプリケーションの作成 - プロジェクトの名前付け」ページが表示されます。「プロジェクト名」フィールドにプロジェクトの適切な名前を入力します。たとえば、ISGRESTClient です。

「プロジェクトの機能」タブで、「使用可能」リストから「Java」を選択します。右矢印ボタンを使用して、「使用可能」ウィンドウから「選択済」ウィンドウに、選択した機能を移動します。

「次へ」をクリックします。

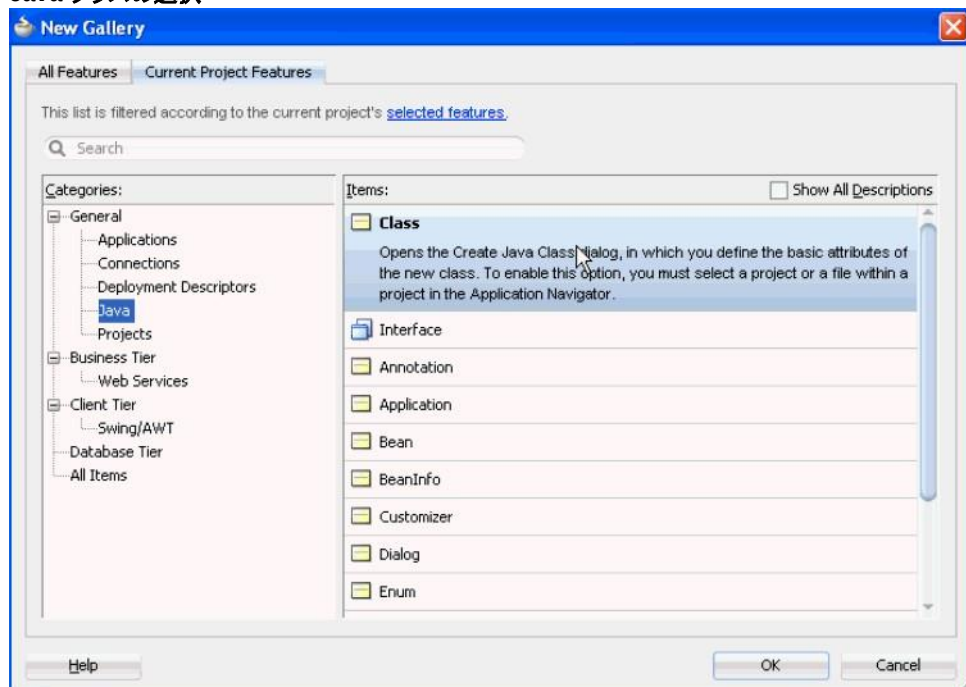
5. 「Java 設定の構成」ダイアログ・ボックスで「終了」をクリックします。
新しく作成したプロジェクトが「プロジェクト」ワークスペースに表示されるはずです。
6. アプリケーション・ナビゲータで、作成したばかりのプロジェクトの名前を選択して右クリックし、ドロップダウン選択メニューから「新規」を選択します。

既存のプロジェクトからの「新規」の選択



7. 「新規ギャラリー」ウィンドウで、「一般」カテゴリを展開して「Java」を選択します。「アイテム」リストで「Java クラス」を選択します。「OK」をクリックします。

Java クラスの選択



8. 「Java クラスの作成」ダイアログで、デフォルトのクラス名を `RestInvocationBasicAuthWithHeader` に変更します。その他の項目はデフォルトのまま、**「OK」**をクリックします。
9. ソース・エディタで新しいクラスが自動的に開き、スケルトン・クラス定義が表示されます。次の Java コードでスケルトン・クラス定義を置き換えます。

```

package sample;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import com.sun.jersey.core.util.Base64;

public class RestInvocationBasicAuthWithHeader {
    // xml payload with REST header for invoking the service
    private static final String xmlRequest4 = "<ns:GET_Input xmlns:
ns=\"http://xmlns.oracle.
com/apps/fnd/soapprovider/plsql/rest/fnd_profile/get/\">
    + \"      xmlns:ns1=\"http://xmlns.oracle.
com/apps/fnd/soapprovider/plsql/rest/fnd_profile/header/\">\"
    + \"      <ns1:RESTHeader>\"
    + \"      <ns1:Responsibility>SYSTEM_ADMINISTRATOR</ns1:
Responsibility>\"
    + \"      <ns1:RespApplication>SYSADMIN</ns1:RespApplication>\"
    + \"      <ns1:SecurityGroup>STANDARD</ns1:SecurityGroup>\"
    + \"      <ns1:NLSLanguage>AMERICAN</ns1:NLSLanguage>\"
    + \"      <ns1:Org_Id>202</ns1:Org_Id>\"
    + \"      </ns1:RESTHeader>\"
    + \"      <ns:InputParameters>\"
    + \"      <ns:NAME>APPS_SERVLET_AGENT</ns:NAME>\"
    + \"      </ns:InputParameters> + </ns:GET_Input>\";
/**
 * This method invokes a REST service using basic Authentication
and xml payload with REST headers.
 */
    public static void postXml_BasicAuth(String svcUrlStr, String
username,String passwd) throws IOException {

        URL url = new URL(svcUrlStr);
        // Obtaining connection to invoke the service
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        // Setting Http header values
        conn.setRequestMethod("POST");
        conn.setRequestProperty("Content-Type", "application/xml");
        String auth = username + ":" + passwd;
        byte[] bytes = Base64.encode(auth);
        String authStr = new String(bytes);
        conn.setRequestProperty("Authorization", "Basic " + authStr);
        conn.setRequestProperty("Accept", "application/xml");
        conn.setRequestProperty("Content-Language", "en-US");
        conn.setUseCaches(false);
        conn.setDoInput(true);
        conn.setDoOutput(true);
        // Send request
        OutputStreamWriter wr = new OutputStreamWriter(conn.
getOutputStream());
        wr.write(xmlRequest4.toCharArray());
        wr.flush();
        wr.close();
        conn.connect();
        System.out.println("Response code - " + conn.getResponseCode());
        // Get Response
        String response = null;
        try {
            response = readHttpResponse(conn);
        } finally {

```

```

if (conn != null)
    conn.disconnect();
}
// Show response
System.out.println("Response is : \n" + response);
}

/**
 * This method reads response from server and returns it in a
 * string representation.
 */
private static String readHttpResponse(HttpURLConnection conn) {

    InputStream is = null;
    BufferedReader rd = null;
    StringBuffer response = new StringBuffer();
    try {

        if (conn.getResponseCode() >= 400) {
            is = conn.getErrorStream();
        } else {
            is = conn.getInputStream();
        }
        rd = new BufferedReader(new InputStreamReader(is));
        String line;
        while ((line = rd.readLine()) != null) {
            response.append(line);
            response.append('\n');
        }
    } catch (IOException ioe) {
        response.append(ioe.getMessage());
    } finally {
        if (is != null) {
            try {
                is.close();
            } catch (Exception e) {
            }
        }
        if (rd != null) {
            try {
                rd.close();
            } catch (Exception e) {
            }
        }
    }
    return (response.toString());
}

public static void main(String a[]) {
    String baseUrl = "http://<server hostname>:<port>/webservices/rest";
    String svcUrlStr1 = baseUrl + "/FndProfileSvc/get/";
    // invoke Rest service using basic authentication method
    try {
        postXml_BasicAuth(svcUrlStr1, "<EBS username>", "<password>");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}

```

デプロイ済 WADL から以前に記録されたリソース情報が、baseUrl と svcUrlStr1 の要素に置かれたこと注意してください。

注意: TLS が有効化された環境で Oracle E- Business Suite インスタンスが実行されている場合、baseUrl で **https** (http ではなく)を使用します。また、クライアントの JVM のキーストアに TLS 証明書をインポートする必要があります。

10. コードで、<server hostname>:<port>、<EBS username>および<password>を実際の値に置き換えます。
11. 「ファイル」→「すべて保存」を選択して、作業内容を保存します。

Java クラスを使用した REST サービスの起動

Java クラス RestInvocationBasicAuthWithHeader. java を使用してプロジェクトを作成した後、プロセスをコンパイルおよび実行して REST サービスを起動する必要があります。

次のステップを使用して、Java クラスをコンパイルして実行します。

1. アプリケーション・ナビゲータで、設計時に作成したばかりの RestInvocationWithLogin. java Java クラスを右クリックします。メニューから「メイク」を選択します。
2. RestInvocationBasicAuthWithHeader. java Java クラスを右クリックします。メニューから「実行」を選択します。

「ログ」ウィンドウで、このプロセスをモニターして、コンパイルが成功したことを確認します。
「ログ」ウィンドウで、実行が成功したことを確認します。

出力メッセージの表示

REST サービスが正常に起動されると、「ログ」ウィンドウに次の出力が表示されます。

```
Response code - 200
Response is :
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<OutputParameters xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.oracle.com/apps/fnd/rest/FndProfileSvc/get/">
<VAL>http://<server hostname>:<port>/OA_HTML</VAL>
</OutputParameters>
```

プロファイル・オプション APPS_SERVLET_AGENT の値は、サービスによって取得され、<VAL></VAL>タグに表示されます。

サービスのデプロイ時に入力したサービス別名情報 **FndProfileSvc** が、サービス・エンドポイントの一部として表示されていることに注目してください。

トークン・ベース認証および JSON ペイロードを使用した REST サービスの起動

REST サービスの起動シナリオ

REST サービスの起動について説明するために、この例では PL/SQL API の「ユーザー」(FND_USER_PKG)が使用されます。

Oracle E-Business Suite でのユーザー名のテストを要求するために結果の HTTP 要求が同じユーザーから受信されると、ユーザーを FND_USER 表に対してテストするために、API に含まれる Test User Name (TESTUSERNAME) REST サービス操作が起動されます。

この要求にパスワードが指定されない場合、ユーザー資格証明を認証するためにトークン・ベースのセキュリティ・メソッドが使用されます。Oracle E-Business Suite ユーザー・セッションを作成するためにセキュリティ・ログイン・サービスが起動されて、ユーザー認証のパスワードのかわりに、Cookie としてのセッション ID が戻されます。検証後、Test User Name (TESTUSERNAME) REST サービス操作を起動できます。

この例では、テストされるユーザー名情報は、REST サービスの起動の JSON ベースのペイロードで渡されます。サービスが正常に実行されていると、ペイロードで渡されたユーザー名が Oracle E-Business Suite に存在する場合、TESTUSERNAME 操作によってプラスの数値が返されます。ユーザー名が存在しない場合は、かわりに 0 の数値が返されます。

PL/SQL REST サービスを使用するための前提条件

設計時タスクを実行する前に、次のタスクが実行されるようにします。

必要なライブラリの取得

JSON 形式のペイロードを使用して REST サービスを正常に起動するには、
<\$COMMON_TOP>/java/lib ディレクトリで入手できる次のライブラリを取得する必要があります。

- jersey-bundle_1.0.0.0_1-1-5-1.jar
- jackson-core-asl_1.0.0.0_1-1-1.jar
- jackson-mapper-asl_1.0.0.0_1-1-1.jar

これらのライブラリ・ファイルは、後で設計時にプロジェクト作成中にプロジェクトに追加されます。

HTTP 要求の RESTHeader の変数の設定

REST サービスの起動前に、REST サービスで、アプリケーション・コンテキストの値を RESTHeader 要素で渡すことができます。

PL/SQL インタフェース・タイプでのこれらの RESTHeader 要素は、Responsibility、RespApplication、SecurityGroup、NLSLanguage および Org_Id です。

PL/SQL インタフェース・タイプでの RESTHeader 要素の詳細は、「HTTP Basic 認証および REST ヘッダー付き XML ペイロードを使用した REST サービスの起動」(ページ 3-73)を参照してください。

Java を使用した REST サービスの起動

この項には、REST サービスの起動シナリオに基づく次のタスクが含まれています。

1. PL/SQL REST Web サービスの配置、ページ 3-85

2. 配置済 WADL URL の記録、ページ 3-87
3. Java クラスでのプロジェクトの作成、ページ 3-88
4. Java クラスを使用した REST サービスの起動、ページ 3-99

PL/SQL REST Web サービスの配置

次のステップを使用して、「ユーザー」API (FND_USER_PKG)を配置します。

1. 「統合管理者」ロールを持つユーザーとして、Oracle E-Business Suite にログインします。ナビゲーション・メニューから、「統合 SOA ゲートウェイ」の職責を選択して、「統合リポジトリ」リンクを選択します。
2. 「統合リポジトリ」タブで、「**検索**」をクリックしてメインの「検索」ページにアクセスします。
3. 「内部名」フィールドに FND_USER_PKG と入力します。「**進む**」をクリックして、検索を実行します。「ユーザー」インタフェース名リンクをクリックして、インタフェース詳細ページを開きます。
4. 「REST Web サービス」タブで、次の情報を入力します。

REST サービスとしての PL/SQL API の配置

Overview SOAP Web Service **REST Web Service** Grants

* Service Alias:
 REST Service Status: Not Deployed

Service Operations

Select All | Select None Rows 1 to 15

Select	Name ▲	Internal Name ▲	Grant
☐	LDAP Wrapper Update User	LDAP_WRP_UPDATE_USER_LOADER	
☐	LDAP Wrapper Update User	LDAP_WRAPPER_UPDATE_USER	
☐	Pre-validate EBS user	ISUSERACTIVE	
☐	Pre-validate EBS user (2)	ISUSERACTIVE	
☐	Remove PII user information	REMOVE_PII_USER	
☐	Set Old Person Party Id	SET_OLD_PERSON_PARTY_ID	
☐	Set Old User GUID	SET_OLD_USER_GUID	
☒	Test User Name	TESTUSERNAME	
☐	User Form LDAP Wrapper Update User	FORM_LDAP_WRAPPER_UPDATE_USER	
☐	Validate User Name	VALIDATE_USER_NAME	

☒ TIP To apply any changes in Operation, Undeploy the service.

REST Service Security

REST Web Service is secured by HTTP Basic Authentication at HTTP Transport level. Send either of the following in "Authorization" header as per HTTP Basic scheme:
 - Username:Password
 - Security Token.

Tip: Use [Login Service](#) to obtain Security Token for given user credentials.

- サービス別名: fndMessageSvc
 選択したメソッドまたはサービス操作(この例では、TESTUSERNAME)の WADL およびスキーマのサービス・エンドポイントとして別名が表示されます。
- 「サービス操作」リージョンで、「Test User Name」(TESTUSERNAME)を選択します。
 選択されたメソッドは、REST サービス操作として公開されます。

5. 「配置」をクリックして、Oracle E-Business Suite WebLogic 環境にサービスを配置します。

REST サービスが正常に配置されたら、WADL の記述を表示できる「**WADL の表示**」リンクとともに、「REST サービス・ステータス」フィールドに「配置済」が表示されます。

REST サービスの配置の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプレメンテーション・ガイド*の「REST Web サービスの配置」を参照してください。

配置済 WADL URL の記録

FND_USER_PKG サービスの配置済 WADL からサービス・リソース情報を取得するために、統合開発者は「REST Web サービス」タブの「**WADL の表示**」リンクをクリックします。

WADL の記述の表示

OverviewSOAP Web ServiceREST Web Service

Service Alias fndMessageSvc
REST Service Status Deployed
Verb POST

[View WADL](#)

Service Operations

Rows 1 to 15

Name ▲	Internal Name ▲	Included Operations	Grant
LDAP Wrapper Update User	LDAP_WRP_UPDATE_USER_LOADER		
LDAP Wrapper Update User	LDAP_WRAPPER_UPDATE_USER		
Pre-validate EBS user	ISUSERACTIVE		
Pre-validate EBS user (2)	ISUSERACTIVE		
Remove PII user information	REMOVE_PII_USER		
Set Old Person Party Id	SET_OLD_PERSON_PARTY_ID		
Set Old User GUID	SET_OLD_USER_GUID		
Test User Name	TESTUSERNAME	✓	
User Form LDAP Wrapper Update User	FORM_LDAP_WRAPPER_UPDATE_USER		
Validate User Name	VALIDATE_USER_NAME		

TIP To apply any changes in Operation,Undeploy the service.

REST Service Security

REST Web Service is secured by HTTP Basic Authentication at HTTP Transport level. Send either of the following in "Authorization" header as per HTTP Basic scheme:
- Username:Password
- Security Token.

Tip: Use [Login Service](#) to obtain Security Token for given user credentials.

次の WADL 記述が表示されます。

```

<xml version="1.0" encoding="UTF-8" standalone="no" ?>
<application xmlns:tns="http://xmlns.oracle.
com/apps/fnd/soapprovider/plsql/rest/fnd_user_pkg/" xmlns="http://wadl.
dev.java.net/2009/02" xmlns:tns1="http://xmlns.oracle.
com/apps/fnd/rest/fndMessageSvc/testusername/" name="FND_USER_PKG"
targetNamespace="http://xmlns.oracle.
com/apps/fnd/soapprovider/plsql/rest/fnd_user_pkg/">
<grammars>
  <include xmlns="http://www.w3.org/2001/XMLSchema" href="https:
//<hostname>:<port>/webservicess/rest/fndMessageSvc/?
XSD=TESTUSERNAME_SYNCH_TYPEDEF.xsd" />
</grammars><resources base="http://<hostname>:
<port>/webservicess/rest/fndMessageSvc/"><resource
path="/testusername/">
  <method id="GET" name="POST">
    <request>
      <representation mediaType="application/xml" type="tns1:
InputParameters" />
      <representation mediaType="application/json" type="tns1:
InputParameters" />
    </request>
    <response>
      <representation mediaType="application/xml" type="tns1:
OutputParameters" />
      <representation mediaType="application/json" type="tns1:
OutputParameters" />
    </response>
  </method>
</resource>
</resources>
</application>

```

後で Java クライアントを定義するときに使用される次の情報をコピーまたは記録します。

- `<resources base>="http://<hostname>:
<port>/webservicess/rest/fndMessageSvc/"`

この情報

`http://<hostname>:<port>/webservicess/rest/fndMessageSvc` は、後で Java クライアント・プログラムでベース URL として使用されます。

- `<resource path>="/testusername/"`

この情報は、後でサービス URL の後ろの部分形成するために使用されます。

Java クラスを含むプロジェクトの作成

この項では、FND_USER_PKG REST サービスの起動に使用される Java クラス (RestInvocationWithLogin.java) および JSON ペイロードを含むプロジェクトを作成する方法について説明します。

Java クラスを含むプロジェクトを作成するには:

1. Oracle JDeveloper で、メイン・メニューから「ファイル」→「新規」を選択します。

「新規ギャラリー」ウィンドウで、「一般」カテゴリを展開して「アプリケーション」を選択します。「アイテム」リストで「カスタム・アプリケーション」を選択します。

「OK」をクリックします。「カスタム・アプリケーションの作成 - アプリケーションの名前付け」ページが表示されます。

2. 「アプリケーション名」フィールドにアプリケーションの適切な名前を入力します。「次へ」をクリックします。
3. 「カスタム・アプリケーションの作成 - プロジェクトの名前付け」ページが表示されます。「プロジェクト名」フィールドにプロジェクトの適切な名前を入力します。たとえば、ISGRESTClient3 です。

「プロジェクトの機能」タブで、「使用可能」リストから「**Java**」を選択します。右矢印ボタンを使用して、「使用可能」ウィンドウから「選択済」ウィンドウに、選択した機能を移動します。

「次へ」をクリックします
4. 「Java 設定の構成」ダイアログ・ボックスで「**終了**」をクリックします。
5. アプリケーション・ナビゲータで、作成したばかりのプロジェクトを右クリックし、ドロップダウン・メニューから「**新規**」を選択します。
6. 「新規ギャラリー」ウィンドウで、「一般」カテゴリを展開して「**Java**」を選択します。「アイテム」リストで「**Java クラス**」を選択します。「**OK**」をクリックします。
7. 「Java クラスの作成」ダイアログで、デフォルトのクラス名を RestInvocationWithLogin に変更します。その他の項目はデフォルトのままで、「**OK**」をクリックします。
8. ソース・エディタで新しいクラスが自動的に開き、スケルトン・クラス定義が表示されます。次の Java コードでスケルトン・クラス定義を置き換えます。

```

package sample;

import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import org.codehaus.jackson.JsonNode;
import org.codehaus.jackson.JsonParseException;
import org.codehaus.jackson.JsonParser;
import org.codehaus.jackson.map.ObjectMapper;
import com.sun.jersey.core.util.Base64;

public class RestInvocationWithLogin {

    private static final String jsonRequest1 = "{ \"
TESTUSERNAME_Input\":{ \"
+ \"   \"@xmlns\": \"http://xmlns.oracle.com/apps/fnd/rest/
fndMessageSvc/testusername/\", \"
+ \"   \"RESTHeader\":{ \"
+ \"       \"@xmlns\": \"http://xmlns.oracle.
com/apps/fnd/rest/fndMessageSvc/header/\", \"
+ \"       \"Responsibility\": \"SYSTEM ADMINISTRATOR\", \"
+ \"       \"RespApplication\": \"SYSADMIN\", \"
+ \"       \"SecurityGroup\": \"STANDARD\", \"
+ \"       \"NLSLanguage\": \"AMERICAN\", \"
+ \"       \"Org_Id\": \"202\" \"
+ \"   }, \" + \"InputParameters\":{ \"
+ \"       \"X_USER_NAME\": \"operations\" \" + \"   }\" + \"}}\";

/**
 * This Method invokes the a rest service using the accessTokenName
and accessToken values returned by AOL login service
 */
    public static void postJSON_AolToken(String svcUrlStr, String
tokenName,String tokenValue) throws IOException {

        URL url = new URL(svcUrlStr);
        //Obtaining connection to invoke the service
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        //Setting Http header values
        conn.setRequestMethod("POST");
        conn.setRequestProperty("Content-Type", "application/json");
        //Adding the accessTokenName and accessToken as Cookies
        conn.addRequestProperty("Cookie", tokenName + "=" + tokenValue);
        conn.setRequestProperty("Accept", "application/json");
        conn.setRequestProperty("Content-Language", "en-US");
        conn.setUseCaches(false);
        conn.setDoInput(true);
        conn.setDoOutput(true);
        //Send request
        OutputStreamWriter wr = new OutputStreamWriter(conn.
getOutputStream());
        wr.write(jsonRequest1.toCharArray());
        wr.flush();
        wr.close();
        conn.connect();
        System.out.println("Response code - " + conn.getResponseCode());
        //Get Response
        String response = null;
        try {

```



```

response = readHttpResponse(conn);
    } finally {
        if (conn != null)
            conn.disconnect();
    }
    //Show Response
    System.out.println("Response is : \n" + response);
}

/**
 * This method invokes the AOL login service.It authenticates login
 * credentials and returns accessTokenName and accessToken values after
 * successful validation of login credentials.
 */
private static String[] getAolToken(String baseUrl, String
username,String passwd) throws Exception {

    String rfUrl = baseUrl + "/login";
    URL url = new URL(rfUrl);
    //Obtaining connection to invoke login service.
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    String auth = username + ":" + passwd;
    byte[] bytes = Base64.encode(auth);
    String authStr = new String(bytes);
    //Setting Http request method
    conn.setRequestMethod("GET");
    //Setting the Http header values
    conn.setRequestProperty("Authorization", "Basic " + authStr);
    conn.setRequestProperty("Content-type", "application/json");
    conn.setRequestProperty("Accept", "application/json");
    conn.setUseCaches(false);
    conn.setDoInput(true);
    conn.setDoOutput(true);
    conn.connect();
    String response = null;
    //Get Response
    try {
        response = readHttpResponse(conn);
    } finally {
        if (conn != null)
            conn.disconnect();
    }
    //Parsing Response to obtain
    JsonParser jp = null;
    JsonNode root = null;
    ObjectMapper mapper = new ObjectMapper();
    try {
        jp = mapper.getJsonFactory().createJsonParser(new
ByteArrayInputStream(response.getBytes()));
        jp.disableFeature(org.codehaus.jackson.JsonParser.Feature.
AUTO_CLOSE_SOURCE);
        root = jp.readValueAsTree();
    } catch (JsonParseException jpe) {
        jpe.printStackTrace();
    } catch (IOException ioe) {
        ioe.printStackTrace();
    }
    JsonNode dataNode = root.get("data");
    JsonNode accessTokenNode = dataNode.get("accessToken");
    String accessToken = accessTokenNode.getTextValue();
    JsonNode accessTokenNameNode = dataNode.get("accessTokenName");
    String accessTokenName = accessTokenNameNode.getTextValue();
    return (new String[] { accessTokenName, accessToken });
}

```

```

/**
 * This method reads response sent by the server and returns it
 * in a string representation.
 */
private static String readHttpResponse(HttpURLConnection conn) {
    InputStream is = null;
    BufferedReader rd = null;
    StringBuffer response = new StringBuffer();
    try {
        if (conn.getResponseCode() >= 400) {
            is = conn.getErrorStream();
        } else {
            is = conn.getInputStream();
        }
        rd = new BufferedReader(new InputStreamReader(is));
        String line;
        while ((line = rd.readLine()) != null) {
            response.append(line);
            response.append('\n');
        }
    } catch (IOException ioe) {
        response.append(ioe.getMessage());
    } finally {
        if (is != null) {
            try {
                is.close();
            } catch (Exception e) {
            }
        }
        if (rd != null) {
            try {
                rd.close();
            } catch (Exception e) {
            }
        }
    }
    return (response.toString());
}

public static void main(String[] args) throws Exception {
    String baseUrl = "http://<server hostname>:<port>/webservices/rest
";
    String svcUrlStr1 = baseUrl + "/fndMessageSvc/testusername/";
    //Get Access Token by invoking AOL Login Service
    String[] token = getAolToken(baseUrl, "<EBS username>", "
<password>");
    System.out.println("AOL Token : Name - " + token[0] + ", Value -
" + token[1]);
    //Invoke REST service using the Access Token
    postJSON_AolToken(svcUrlStr1, token[0], token[1]);
}
}

```

デプロイ済 WADL から以前に記録されたリソース情報が、baseUrl と svcUrlStr1 の要素に置かれたこと注意してください。

注意: TLS が有効化された環境で Oracle E- Business Suite インスタンスが実行されている場合、baseUrl で **https** (http ではなく)を使用します。また、クライアントの JVM のキーストアに TLS 証明書をインポートする必要があります。

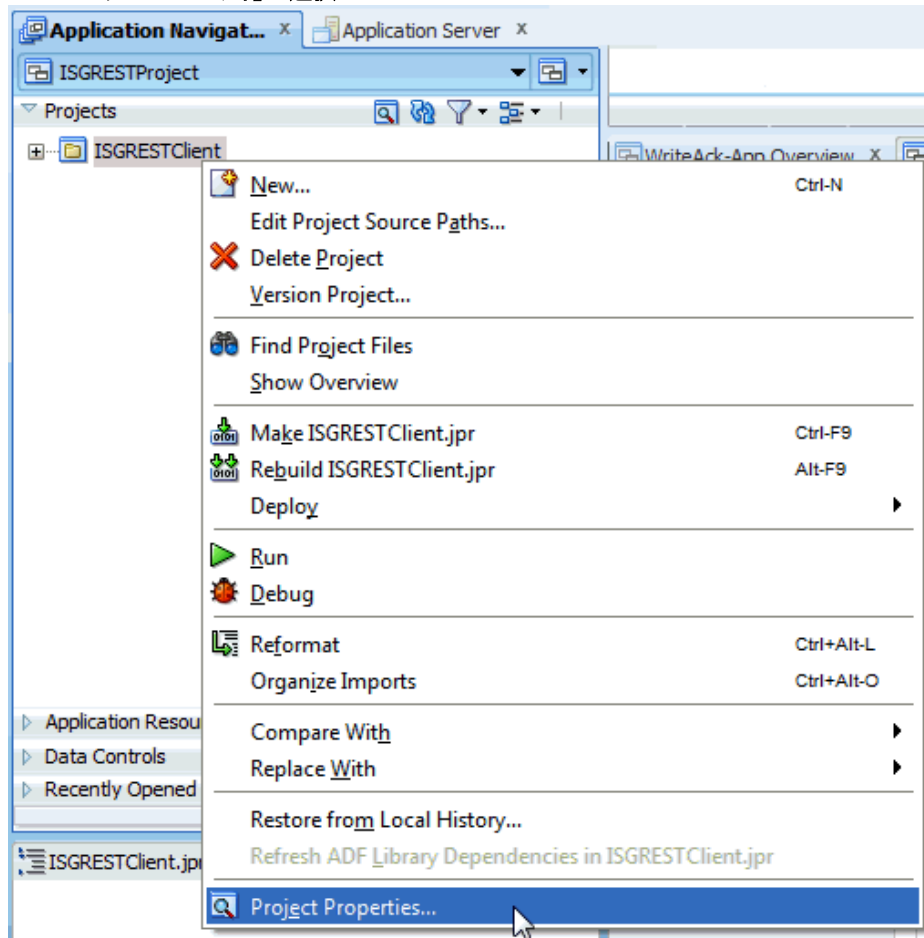
9. コードで、<server hostname>:<port>、<EBS username>および<password>を実際の値に置き換えます。

10. JSON ペイロードの処理に必要なライブラリを追加します。

次のステップを使用して、必要なライブラリ・ファイルをプロジェクト・プロパティに追加します。

1. 作成したばかりのプロジェクトの名前を選択して右クリックし、選択メニューを開きます。
2. メニューから「プロジェクト・プロパティ」を選択します。

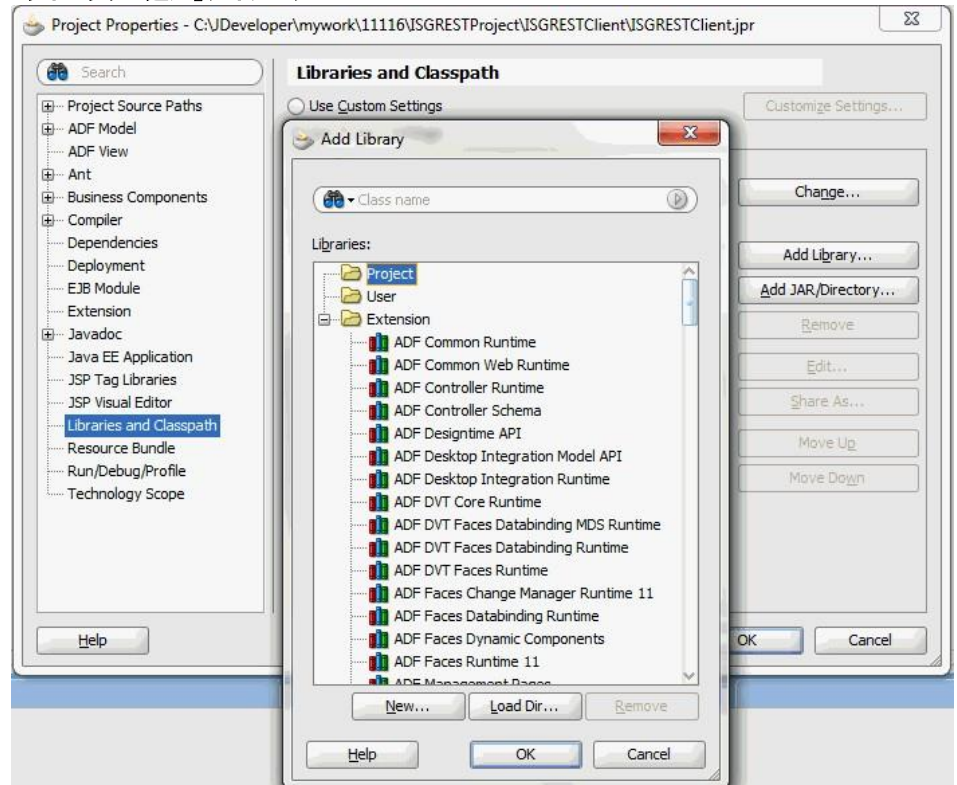
「プロジェクト・プロパティ」の選択



「デフォルト・プロパティ」ダイアログ・ボックスが開きます。

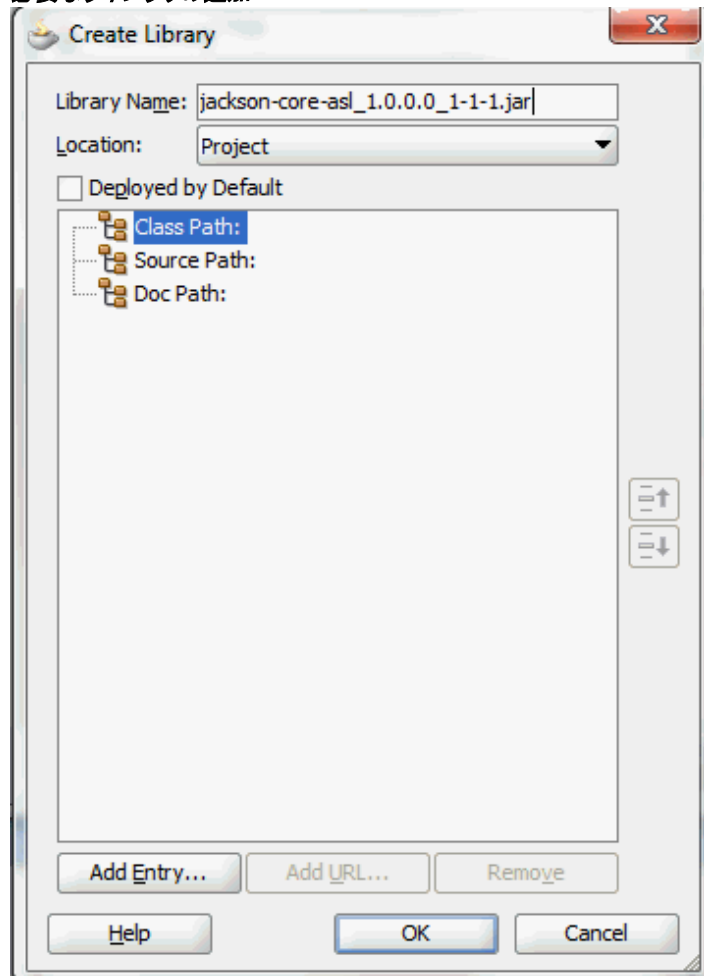
3. 「ライブラリとクラスパス」を選択して、「ライブラリの追加」をクリックします。「ライブラリの追加」ダイアログ・ボックスが開きます。

「ライブラリの追加」ダイアログ



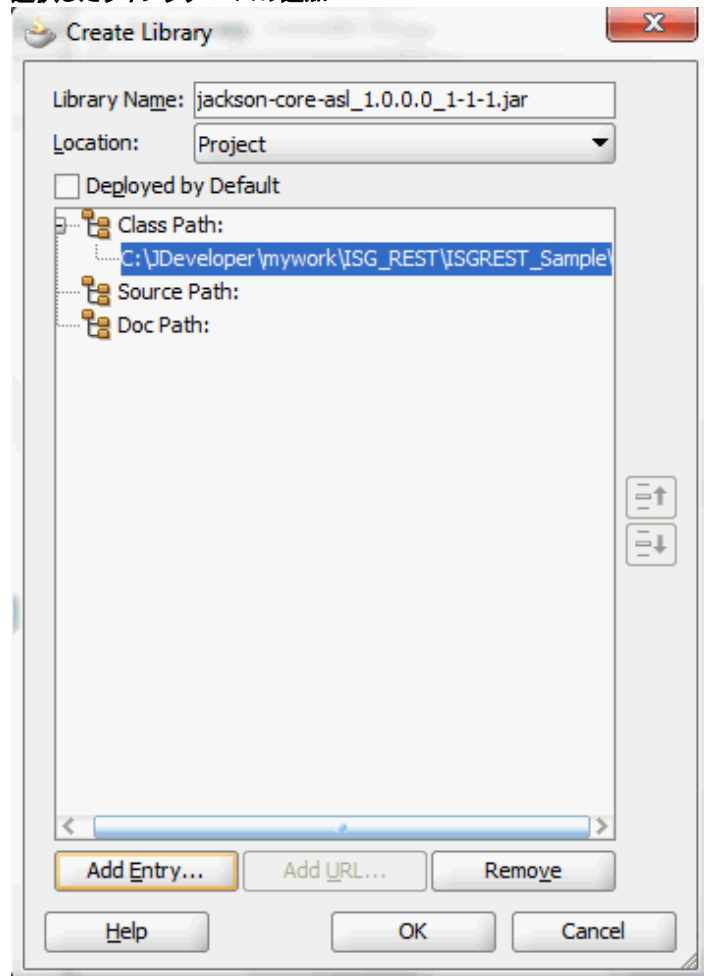
4. 「ライブラリの追加」ダイアログ・ボックスで、「プロジェクト」フォルダを選択して、「新規」をクリックします。「ライブラリの作成」ダイアログ・ボックスが開きます。
5. 「ライブラリ名」フィールドに、`jackson-core-asl_1.0.0.0_1-1-1.jar` と入力します。「エントリの追加」をクリックします。「パス・エントリの選択」ダイアログ・ボックスが表示されます。

必要なライブラリの追加



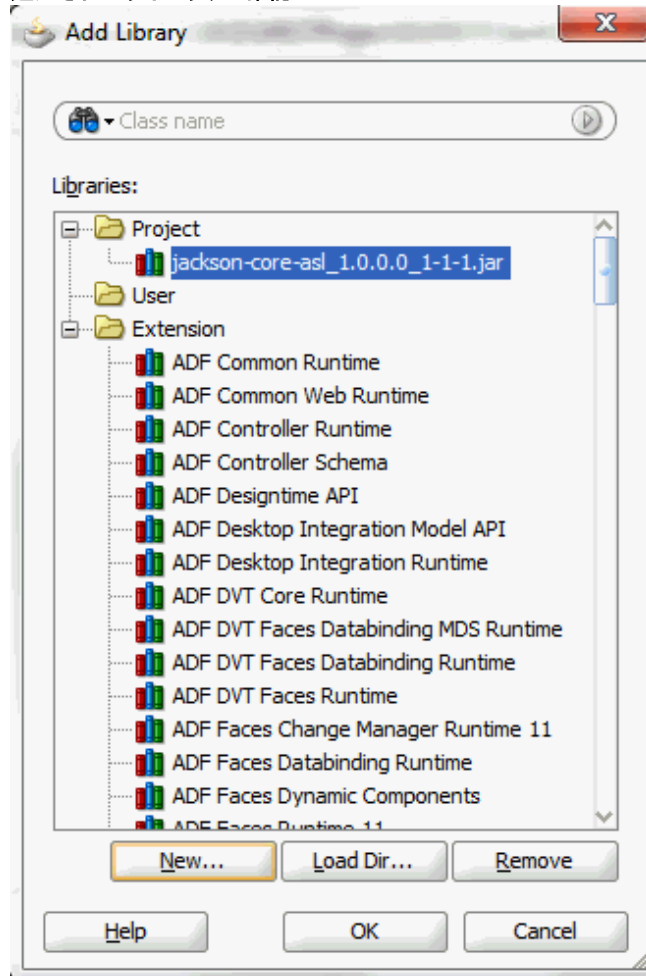
6. 「パス・エントリの選択」ダイアログ・ボックスで、ダウンロードした jackson-core-asl_1.0.0.0_1-1-1.jar ファイルを見つけて選択します。こうすることでクラスパスに追加されます。

選択したライブラリ・パスの追加



「OK」をクリックします。ackson-core-asl_1.0.0.0_1-1-1.jar が「プロジェクト」フォルダに追加されます。

追加されたライブラリの確認



7. ステップ 4、5 および 6 を繰り返して、次の 2 つの jar ファイルを「プロジェクト」フォルダに追加します。

- jersey-bundle_1.0.0.0_1-1-5-1.jar
- jackson-mapper-asl_1.0.0.0_1-1-1.jar

これらの 3 つの jar ファイルが「プロジェクト」フォルダに表示されます。「OK」をクリックします。

8. 「プロジェクト・プロパティ」ダイアログ・ボックスが表示されます。「OK」をクリックします。このプロジェクトが、必要なライブラリで設定されました。

必要なライブラリの確認



11. 「ファイル」→「すべて保存」を選択して、作業内容を保存します。

Java クライアントを使用した REST サービスの起動

Java クラス `RestInvocationWithLogin.java` を使用してプロジェクトを作成した後、プロセスをコンパイルおよび実行して `FND_USER_PKG` REST サービスを起動する必要があります。

次のステップを使用して、Java クラスをコンパイルして実行します。

1. アプリケーション・ナビゲータで、設計時に作成したばかりの `RestInvocationWithLogin.java` Java クラスを右クリックします。メニューから「メイク」を選択します。
2. `RestInvocationWithLogin.java` Java クラスを右クリックし、メニューから「実行」を選択します。

「ログ」ウィンドウで、このプロセスをモニターして、コンパイルが成功したことを確認します。
「ログ」ウィンドウで、実行が成功したことを確認します。

出力メッセージの表示

`FND_USER_PKG` REST サービスが正常に起動されると、次の出力が表示されます。

- ログイン・サービスからの応答は次のようになります。
AOL Token : Name - isgdemo, Value -
5xjE6DVfi7NlEGnS70xX1fvrdt

- サービス起動からの応答は次のようになります。

```
Response code - 200
Response is :
{
  "OutputParameters" : {
    "@xmlns:xsi" : "http://www.w3.org/2001/XMLSchema-instance",
    "@xmlns" : "http://xmlns.oracle.com/apps/fnd/rest/fndMessageSvc
/testusername/",
    "TESTUSERNAME" : "2"
  }
}
```

この例では、プラスの数値 2 が返され、これはペイロードで渡されたユーザー名が Oracle E-Business Suite に存在しないことを示します。

サービスの配置時に入力したサービス別名情報 **fndMessageSvc** が、サービス・エンドポイントの一部として表示されていることに注目してください。

- ログアウト・サービスからの応答は次のようになります。

```
Response is :
{
  "data" : {
    "accessToken" : "-1",
    "accessTokenName" : "isgdemo",
    "ebsVersion" : null
  }
}
```

RESTサービスとしてのJava APIの使用

概要

Java API は、Java クラスに基づくビジネス・インタフェースです。シリアライズ可能な Java Bean、または String、Int などの単純なデータ型のパラメータをメソッドで使用する必要がある一部の特殊化された Java API は、**Java Bean サービス**として分類されることがあります。Java クラスの中には、OA フレームワーク・ベースのコンポーネントおよびページを管理するビジネス・ロジックへのアクセスを提供するものがあります。そのような Java クラスは**アプリケーション・モジュール・サービス**と呼ばれ、これらも Java インタフェースのサブタイプとして分類されます。

Java インタフェースのこのようなサブタイプを探すには、「カテゴリ」として「インタフェース・サブタイプ」を指定し、「カテゴリ値」として「Java Bean サービス」または「アプリケーション・モジュール・サービス」を指定して、検索を実行します。Java Bean サービスとアプリケーション・モジュール・サービスは、どちらも REST サービスとしてのみ公開できます。

Java Bean サービスまたはアプリケーション・モジュール・サービスが REST サービスとして正常に配置されると、統合開発者は、Java、PHP、Javascript、Python などの言語を使用して、配置された REST サービスをクライアント・プログラムから起動できます。

サービス起動の例

Java API を REST サービスとして使用してアプリケーション・データをフェッチおよび使用する方法について理解を深めるために、この章には、インタフェース・タイプが Java Bean サービスおよびアプリケーション・モジュール・サービスの場合のサービス起動の例が含まれています。これらの例では、Oracle JDeveloper 11g (11.1.1.6.0)を使用して、Java クラスを含むプロジェクトを作成し、サービスを起動します。

- HTTP GET メソッドを使用した Java Bean サービスの起動、ページ 4-2
- カスタム Java Bean サービスへの注釈付けと起動、ページ 4-16
- トークン・ベース認証および XML ペイロードを使用したアプリケーション・モジュール・サービスの起動、ページ 4-48

HTTP GET メソッドを使用した Java Bean サービスの起動

REST サービスの起動シナリオ

REST サービスの起動を説明するための例として、Java Bean サービス「REST Service Locator」(`oracle.apps.fnd.rep.ws.service.EbsRestLocator`)を考えてみましょう。REST Service Locator は、配置されている Oracle E-Business Suite REST サービスの詳細を取得するメソッドで構成されるサンプル Java API です。

Java クライアントを使用して、`getRestInterface` サービス操作への HTTP GET 要求が行われます。`getRestInterface` サービス操作は、内部名によって識別された REST サービスの詳細を返します。

この例では、HTTP Basic 認証スキームを使用して、ユーザー名およびパスワード情報が HTTP 要求ヘッダーで提供されます。ユーザー資格証明は、ISG の REST サービス・プロバイダによって認証および承認されます。検証後、`getRestInterface` サービス操作は、認証されたユーザーの要求を実行します。

サービスが正常に起動すると、クライアントは、サービスの起動中、実行時に内部名が HTTP URL で渡された REST サービスの詳細を含む REST 応答メッセージを受け取ります。

Java を使用した REST サービスの起動

この項には、REST サービスの起動シナリオに基づく次のタスクが含まれています。

1. REST サービスの配置、ページ 4-2
2. セキュリティ付与の作成、ページ 4-4
3. 配置済 WADL からのリソース情報の記録、ページ 4-6
4. Java クラスを含むプロジェクトの作成、ページ 4-9
5. Java クラスを使用した REST サービスの起動、ページ 4-15

REST サービスの配置

次のステップを使用して、REST Service Locator という Java Bean サービスを配置します。

1. 「統合管理者」ロールを持つユーザーとして、Oracle E-Business Suite にログインします。
職責「統合 SOA ゲートウェイ」を使って、ナビゲーション・メニューから「統合リポジトリ」リンクを選択します。
2. 「統合リポジトリ」タブで、「検索」をクリックしてメインの「検索」ページにアクセスします。
3. 「詳細検索オプションの表示」をクリックして、詳細検索オプションを表示します。

検索基準として、次のキー検索値を入力します。

- カテゴリ: インタフェース・サブタイプ
- カテゴリ値: Java Bean サービス

4. 「進む」をクリックして、検索を実行します。

REST Service Locator のインタフェース名リンクをクリックして、インタフェースの詳細ページを開きます。

5. 「REST Web サービス」タブで、次の情報を入力します。

REST サービスとしての Java API の配置

Overview **REST Web Service** Grants

* Service Alias: servicelocator x

REST Service Status: Not Deployed

Service Operations

Display Name	Internal Name	GET	POST	Grant
Rest Service Locator	oracle.apps.fnd.rep.ws.service.EbsRestLocator	☒	☐	
ADDGRANT	addGrant	☐	☒	
ADDGRANTS	addGrants	☐	☒	
GETOPERATIONS	getOperations	☒	☐	
LISTGRANTS	listGrants	☒	☐	
REVOKEGRANT	removeGrant	☐	☒	
getRestInterface	getRestInterface	☒	☐	
getRestInterfaces	getRestInterfaces	☒	☐	

REST Service Security

REST Web Service is secured by HTTP Basic Authentication at HTTP Transport level. Send either of the following in "Authorization" header as per HTTP Basic scheme:

- Username:Password
- Security Token.

Tip: Use [Login Service](#) to obtain Security Token for given user credentials.

Deploy

- サービス別名: servicelocator

この別名が、WADL のサービス・エンドポイントと、選択されたメソッドまたはオペレーションのスキーマとして利用されます。

- 必要なサービス操作の選択

「サービス操作」リージョンでは、HTTP メソッドのチェック・ボックスがあらかじめ選択されています。

Java メソッドに特定の HTTP メソッドが注釈付けされている場合、そのメソッドについては、対応する HTTP メソッドのチェック・ボックスがあらかじめ選択されることに注意してください。

管理者は、サービスを配置する前に HTTP メソッドのチェック・ボックスの選択を変更できます。

Java Bean サービスのアノテーションに関する詳細なガイドラインは、「Java Bean サービスに対するアノテーション」(ページ A-6)を参照してください。

この例では、getRestInterface サービス操作に GET HTTP メソッドが注釈付けされているため、「GET」チェック・ボックスが自動的に選択されます。

6. 「**配置**」をクリックして、Oracle E-Business Suite WebLogic 環境にサービスを配置します。

REST サービスが正常に配置されたら、「**WADL の表示**」リンクとともに、「REST サービス・ステータス」フィールドに「配置済」が表示されます。「**WADL の表示**」リンクをクリックして、配置されたサービス WADL の記述を表示します。

REST サービスの配置の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「REST Web サービスの配置」を参照してください。

セキュリティ付与の作成

REST Service Locator を REST サービスとして配置したら、統合管理者はセキュリティ付与を作成して、サービスまたはメソッドのアクセス権限を特定のユーザー、ユーザー・グループまたはすべてのユーザーに与えることができます。

次のステップを使用して、セキュリティ付与を作成します。

1. 「統合管理者」ロールを持つユーザーとして、Oracle E-Business Suite にログインします。職責「統合 SOA ゲートウェイ」を使って、ナビゲーション・メニューから「統合リポジトリ」リンクを選択します。
2. 検索を実行して、管理者が先ほど配置した REST Service Locator サービスを探します。
3. 選択したカスタム Java Bean サービスのインタフェースの詳細ページで、「付与」タブをクリックします。
4. getRestInterface メソッドのチェック・ボックスを選択し、「**付与の作成**」をクリックします。

選択したメソッドのセキュリティ付与の作成

The screenshot shows the 'Integration Repository Administration' page for 'Java Details : Rest Service Locator'. The 'Grants' tab is selected, displaying a table of existing grants. The 'getRestInterface' grant is highlighted with a blue row.

Select	Name	Internal Name	REST Service Operation	Grant
☐	ADDGRANT	addGrant		
☐	ADDGRANTS	addGrants		
☐	GETOPERATIO...	getOperations		
☐	LISTGRANTS	listGrants		
☐	REVOKEGRANT	removeGrant		
☒	getRestInterface	getRestInterface		
☐	getRestInterfaces	getRestInterfaces		

5. 「付与の作成」ページで、「被付与者タイプ」として「全ユーザー」を選択します。

注意: この例では、getRestInterface サービス操作はすべてのユーザーに許可されます。実際の実装では、厳格なセキュリティ・ルールを定義する必要があります。ユーザーに対する付与、またはより適切な方法として、ロールおよび職責によって定義されたユーザーのグループに対する付与を作成します。

すべてのユーザーに対する付与の作成

Integration Repository Administration

Integration Repository > Java Details : Rest Service Locator >

Create Grants Cancel Create Grant

Selected Methods

Name	Internal Name
getRestInterface	getRestInterface

Grant All Selected

Grantee Type All Users ▼

Copyright (c) 1998, 2015, Oracle and/or its affiliates. All rights reserved. Privacy Statement

「付与の作成」をクリックします。これにより、すべての Oracle E-Business Suite ユーザーに、選択したメソッドのアクセス権限が付与されます。

配置済 WADL からのリソース情報の記録

REST Service Locator サービスの配置済 WADL からサービス・リソース情報を取得するには、統合開発者が「REST Web サービス」タブで「**WADL の表示**」リンクをクリックします。

WADL の記述の表示

Overview **REST Web Service**

Service Alias servicelocator

REST Service Status Deployed | [View WADL](#)

Service Operations

Expand All Collapse All				
Display Name	Internal Name	GET	POST	Grant
Rest Service Locator	oracle.apps.fnd.rep.ws.service.EbsRestLocator			
ADDGRANT	addGrant		✓	
ADDGRANTS	addGrants		✓	
GETOPERATIONS	getOperations	✓		
LISTGRANTS	listGrants	✓		
REVOKEGRANT	removeGrant		✓	
getRestInterface	getRestInterface	✓		
getRestInterfaces	getRestInterfaces	✓		

次の WADL 記述が表示されます。

```

<xml version="1.0" encoding="UTF-8">
<application name="EbsRestLocator" targetNamespace="http://xmlns.oracle.
com/apps/fnd/soapprovider/pojo/ebsrestlocator/"
fnd/soapprovider/pojo/ebsrestlocator/"
xmlns="http://w3.org/2001/XMLSchema" xmlns:xsd="http://www.w3.
org/2001/XMLSchema"
xmlns:tns1="http://xmlns.oracle.com/apps/fnd/rest/locatorsvc/addgrant/"
xmlns:tns2="http://xmlns.oracle.com/apps/fnd/rest/locatorsvc/addgrants/"

xmlns:tns3="http://xmlns.oracle.
com/apps/fnd/rest/locatorsvc/getoperations/"
xmlns:tns4="http://xmlns.oracle.
com/apps/fnd/rest/locatorsvc/getrestinterface/"
xmlns:tns5="http://xmlns.oracle.
com/apps/fnd/rest/locatorsvc/getrestinterfaces/"
xmlns:tns6="http://xmlns.oracle.
com/apps/fnd/rest/locatorsvc/removegrant/">
<grammars>
  <include href="http://<hostname>:
<port>/webservices/rest/servicelocator/?XSD=addgrant.xsd" xmlns="http:
//www.w3.org/2001/XMLSchema" />
  <include href="http://<hostname>:
<port>/webservices/rest/servicelocator/?XSD=addgrants.xsd" xmlns="http:
//www.w3.org/2001/XMLSchema" />
  <include href="http://<hostname>:
<port>/webservices/rest/servicelocator/?XSD=getoperations.xsd" xmlns="
http://www.w3.org/2001/XMLSchema" />
  <include href="http://<hostname>:
<port>/webservices/rest/servicelocator/?XSD=getrestinterface.xsd"
xmlns="http://www.w3.org/2001/XMLSchema" />
  <include href="http://<hostname>:
<port>/webservices/rest/servicelocator/?XSD=getrestinterfaces.xsd"
xmlns="http://www.w3.org/2001/XMLSchema" />
  <include href="http://<hostname>:
<port>/webservices/rest/servicelocator/?XSD=removegrant.xsd" xmlns="
http://www.w3.org/2001/XMLSchema" />
</grammars><resources base="http://<hostname>:
<port>/webservices/rest/servicelocator/">
  <resource path="addGrant/">
    /
    ...
  </resource path>
  ...
  <resource path="/getRestInterface/{irepClassName}"/>
    <param name="irepClassName" style="template" required="true" type="
xsd:string" />
    <method id="getRestInterface" name="GET">
      <request>
        <param name="ctx_responsibility" type="xsd:string" style="query"
required="false" />
        <param name="ctx_respapplication" type="xsd:string"
style="query" required="false" />
        <param name="ctx_securitygroup" type="xsd:string" style="query"
required="false" />
        <param name="ctx_nlslanguage" type="xsd:string" style="query"
required="false" />
        <param name="ctx_orgid" type="xsd:int" style="query" required="
false" />
      </request>
      <response>
        <representation mediaType="application/xml" type="tns4:
getRestInterface_Output" />
        <representation mediaType="application/json" type="tns4:
getRestInterface_Output" />
      </response>
    </method>
  </resource>
</resources>
</application>

```

```

</method>
</resource>
<resource path="removeGrant/">
/
...
</resource path>
</application>

```

後で Java クライアントを定義するときには使用される次の情報をコピーまたは記録します。

- `<resources base="http://<hostname>:<port>/webservices/rest/servicelocator/">`
 この情報の `http://<hostname>:<port>/webservices/rest/servicelocator/` は、後で Java クライアント・プログラムで `baseUrl` として使用されます。
- `<resource path="/getRestInterface/{irepClassName}"/>`
 この情報の `/getRestInterface/{irepClassName}/` は、後でサービス URL の後ろの部分形成するために使用されます。入力の `{irepClassName}` はパス変数であり、インタフェースの内部名に置き換えられます。

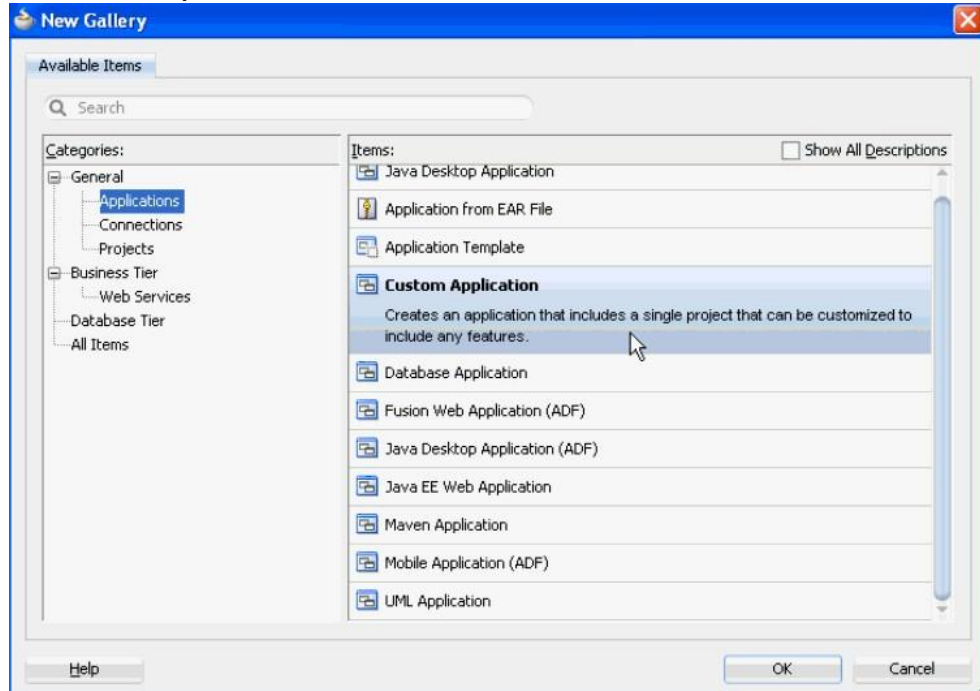
Java クラスを含むプロジェクトの作成

この項では、REST Service Locator サービスを起動するために使用される Java クラスを含むプロジェクトを作成する方法について説明します。

プロジェクトと Java クラスを作成するには:

1. Oracle JDeveloper を開きます。
2. メイン・メニューから「ファイル」→「新規」を選択します。
 「新規ギャラリー」ウィンドウで、「一般」カテゴリを展開して「アプリケーション」を選択します。「アイテム」リストで「カスタム・アプリケーション」を選択します。

Oracle JDeveloper でのプロジェクトの作成



「OK」をクリックします。「カスタム・アプリケーションの作成 - アプリケーションの名前付け」ページが表示されます。

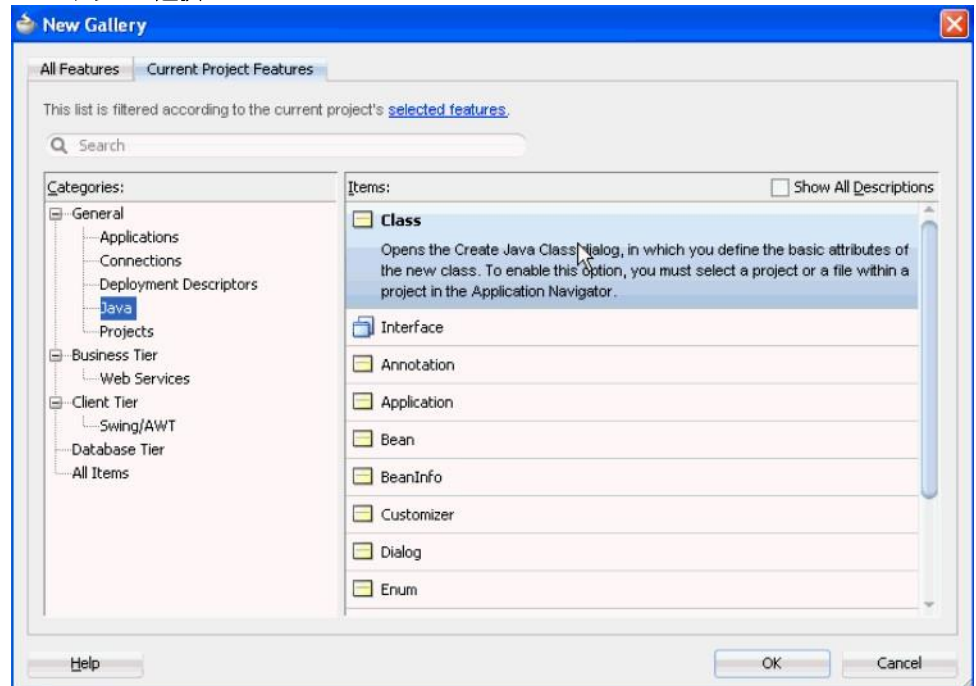
3. 「アプリケーション名」フィールドにアプリケーションの適切な名前を入力します(例: ISGJavaRESTProject)。「次へ」をクリックします。
4. 「カスタム・アプリケーションの作成 - プロジェクトの名前付け」ページが表示されます。「プロジェクト名」フィールドにプロジェクトの適切な名前を入力します(例: ISGJavaRESTClient)。

「プロジェクトの機能」タブで、「使用可能」リストから「Java」を選択します。右矢印ボタンを使用して、「使用可能」ウィンドウから「選択済」ウィンドウに、選択した機能を移動します。

「次へ」をクリックします。

5. 「Java 設定の構成」ダイアログ・ボックスで「終了」をクリックします。
新しく作成したプロジェクトが「プロジェクト」ワークスペースに表示される必要があります。
6. アプリケーション・ナビゲータで、作成したばかりのプロジェクトの名前を選択して右クリックし、ドロップダウン選択メニューから「新規」を選択します。
7. 「新規ギャラリー」ウィンドウで、「一般」カテゴリを展開して「Java」を選択します。「アイテム」リストで「Java クラス」を選択します。「OK」をクリックします。

Java クラスの選択



8. 「Java クラスの作成」ダイアログで、デフォルトのクラス名を `RestInvocationGETMethod` に変更します。その他の項目はデフォルトのまま、**「OK」**をクリックします。
9. ソース・エディタで新しいクラスが自動的に開き、スケルトン・クラス定義が表示されます。次の Java コードでスケルトン・クラス定義を置き換えます。

```

package isgrestget;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import com.sun.jersey.core.util.Base64;

public class RestInvocationGETMethod {

    /**
     * This Method invokes the a rest service using HTTP GET Method
     with path parameter in URL
     */
    public static void invokeREST(String svcUrlStr,String username,
String passwd,String pathParam) throws IOException {
        String getURL = svcUrlStr + "/" + pathParam;
        URL url = new URL(getURL);
        //Obtaining connection to invoke the service
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        String auth = username + ":" + passwd;
        byte[] bytes = Base64.encode(auth);
        String authStr = new String(bytes);

        //Setting Http header values
        conn.setRequestMethod("GET");
        conn.setRequestProperty("Authorization", "Basic " + authStr);
        conn.setRequestProperty("Accept", "application/json");
        conn.setRequestProperty("Content-Language", "en-US");
        conn.setUseCaches(false);
        conn.setDoInput(true);
        conn.setDoOutput(true);
        conn.connect();
        System.out.println("\n 'GET' request sent to URL : " +
url);
        System.out.println("\n Response code - " + conn.
getResponseCode());
        //Get Response
        String response = null;
        try {
            response = readHttpResponse(conn);
        } finally {
            if (conn != null)
                conn.disconnect();
        }
        //Show Response
        System.out.println("Response is : \n" + response);
    }

    /**
     * This method reads response sent by the server and returns it
     in a string representation.
     */
    private static String readHttpResponse(HttpURLConnection conn) {
        InputStream is = null;
        BufferedReader rd = null;
        StringBuffer response = new StringBuffer();
        try {
            if (conn.getResponseCode() >= 400) {
                is = conn.getErrorStream();
            } else {
                is = conn.getInputStream();
            }
        }
    }
}

```

```

rd = new BufferedReader(new InputStreamReader(is));
String line;
while ((line = rd.readLine()) != null) {
    response.append(line);
    response.append('\n');
}
} catch (IOException ioe) {
    response.append(ioe.getMessage());
} finally {
    if (is != null) {
        try {
            is.close();
        } catch (Exception e) {
        }
    }
}
if (rd != null) {
    try {
        rd.close();
    } catch (Exception e) {
    }
}
}
return (response.toString());
}

public static void main(String[] args) throws Exception {
    String baseUrl = "http://<hostname>:
<port>/webservices/rest/servicelocator";
    String svcUrlStr1 = baseUrl + "/getRestInterface";
    //Invoke REST service
    invokeREST(svcUrlStr1, "<EBS username>", "<password>", "PLSQL:
FND_PROFILE");
}
}

```

デプロイ済 WADL から以前に記録されたリソース情報が、baseUrl と svcUrlStr1 の要素に置かれたこと注意してください。PLSQL:FND_PROFILE はインタフェースの内部名であり、インスタンスにデプロイされると想定されるものです。

注意: TLS が有効化された環境で Oracle E- Business Suite インスタンスが実行されている場合、baseUrl で **https** (http ではなく)を使用します。また、クライアントの JVM のキーストアに TLS 証明書をインポートする必要があります。

10. コードで、<EBS hostname>:<port>、<EBS username>および<password>を実際の値に置き換えます。
11. 「ファイル」→「すべて保存」を選択して、作業内容を保存します。
12. JSON 形式に必要なライブラリを追加します。

次のステップを使用して、必要なライブラリ・ファイルをプロジェクト・プロパティに追加します。

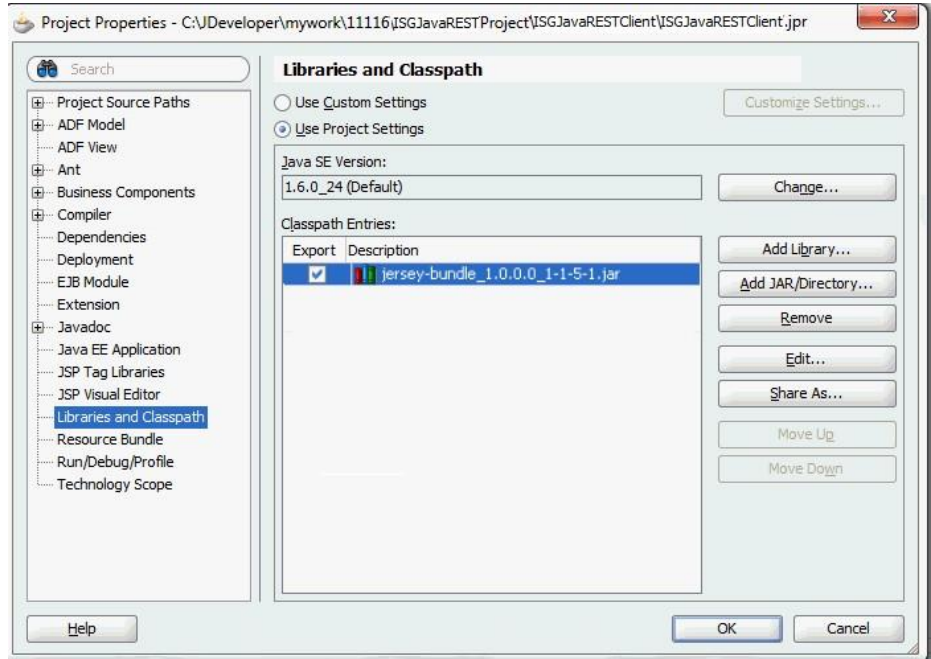
1. 作成したばかりのプロジェクトの名前を選択して右クリックし、選択メニューを開きます。
2. メニューから「プロジェクト・プロパティ」を選択します。

「デフォルト・プロパティ」ダイアログ・ボックスが開きます。

3. 「ライブラリとクラスパス」を選択して、「**ライブラリの追加**」をクリックします。「ライブラリの追加」ダイアログ・ボックスが開きます。
4. 「ライブラリの追加」ダイアログ・ボックスで、「プロジェクト」フォルダを選択して、「**新規**」をクリックします。「ライブラリの作成」ダイアログ・ボックスが開きます。
5. 「ライブラリ名」フィールドに、`Jersey-bundle_1.0.0.0_1-1-5-1.jar` と入力します。「**エントリの追加**」をクリックします。「パス・エントリの選択」ダイアログ・ボックスが表示されます。
6. 「パス・エントリの選択」ダイアログ・ボックスで、ダウンロードした `Jersey-bundle_1.0.0.0_1-1-5-1.jar` ファイルを探して選択します。こうすることでクラスパスに追加されます。

「**OK**」をクリックします。`Jersey-bundle_1.0.0.0_1-1-5-1.jar` が「プロジェクト」フォルダに追加されます。
7. 「プロジェクト・プロパティ」ダイアログ・ボックスが表示されます。「**OK**」をクリックします。このプロジェクトが、必要なライブラリで設定されました。

追加されたライブラリの確認



Java クラスを使用した REST サービスの起動

Java クラス `RestInvocationGETMethod.java` を含むプロジェクトを作成したら、REST サービスを起動するプロセスをコンパイルして実行する必要があります。

次のステップを使用して、Java クラスをコンパイルして実行します。

1. アプリケーション・ナビゲータで、設計時に作成した `RestInvocationGETMethod.java` Java クラスを右クリックします。メニューから「**メイク**」を選択します。
2. `RestInvocationGETMethod.java` Java クラスを右クリックします。メニューから「**実行**」を選択します。

「ログ」ウィンドウで、このプロセスをモニターして、コンパイルが成功したことを確認します。

「ログ」ウィンドウで、実行が成功したことを確認します。

要求ヘッダー情報

この例では、`getRestInterface` Java メソッドは、GET メソッドを使用する REST サービス操作として公開されています。GET メソッドの入力ペイロードはありません。パス変数 `{irepClassName}` は、`getRestInterface` REST サービス操作が起動されたときに、次に示す HTTP URL の一部として送信された実際の値 `PLSQL: FND_PROFILE` に置き換えられます。

```
URL = http://<hostname>:  
<port>/webservices/rest/servicelocator/getRestInterface/PLSQL:
```

FND_PROFILE

注意: GET 要求では、JSON がデフォルトの出力応答形式です。Accept ヘッダー application/xml を使用して、XML 形式の応答を受け入れます。Content-Type ヘッダーが GET HTTP 要求で送信された場合は無視されます。

Request Headers

```
Authorization: Basic c3lzYWRtaW46c3lzYWRtaW4=
Accept: application/json
Content-Language: en-US
```

出力メッセージの表示

REST サービスが正常に起動されると、次に示す JSON 形式の出力が「ログ」ウィンドウに表示されます。

```
{
  "OutputParameters" : {
    "EbsRestServiceBean" : [ {
      "alternateAlias" : "plssql/PLSQL:FND_PROFILE",
      "serviceAlias" : "profile",
      "serviceName" : "PLSQL:FND_PROFILE",
      "wadlUrl" : "http://<hostname>:
<port>/webservices/rest/profile?WADL"
    } ],
    "ControlBean" : [ {
      "fields" : "",
      "filter" : "",
      "limit" : "",
      "offset" : ""
    } ]
  }
}
```

内部名 PLSQL: FND_PROFILE によって識別されたサービス情報が返されていることに注目してください。たとえば、WADL URL、サービス名、サービス別名が応答メッセージの一部として含まれています。

カスタム Java Bean サービスへの注釈付けと起動

統合リポジトリに登録されているすべての Java API を REST サービスとして公開できるわけではないため、適切なアノテーションが付いた、前述の一部の特殊化された Java API のみを REST サービスとして公開できます。それらの特殊化された Java API に Java Bean サービスとして注釈付けする方法、および REST サービス・セキュリティを使用して Java Bean サービスを起動する方法について理解を深めるために、この項では、カスタム Java API への注釈付けから HTTP GET メソッドを使用したサービスの起動までのプロセス全体について説明します。

REST サービスの起動シナリオ

この例では、Employee Service (oracle.apps.per.sample.service.EmployeeInfo) というカスタム Java API を使用して、アノテーションとアップロードのプロセス全体とともにサービスの起動について説明します。2 つの起動シナリオがあります。一方は、従業員の詳細を取得する場合で、もう一方は、ログイン・ユーザー

の直属の部下を取得する場合です。

従業員の詳細を取得するために、Javascript を含む HTML ページを使用して、カスタム API に含まれている `getPersonInfo` サービス操作への HTTP GET 要求が行われます。関連する従業員名と従業員のマネージャ名をサービスが取得できるように、実行時に従業員 ID が提供されます。サービスが正常に起動すると、従業員 ID に対応する従業員の詳細が HTML ページに表示されます。

ログイン・ユーザーの直属の部下を取得するために、このシナリオでは、Javascript を含む別の HTML ページを使用して、`getDirectReports` サービス操作への HTTP GET 要求が行われます。`getDirectReports` サービス操作の起動時に、ログイン・ユーザーの資格証明が提供されます。サービスが正常に起動すると、ログイン・ユーザーの部下またはユーザーの直属の部下が HTML ページに表示され、要求された数のレコードが各ページに表示されます。

カスタム Java Bean サービスを作成するための大まかなプロセス・フロー

カスタム Java Bean サービスを開発するには、統合開発者はカスタム Java API を作成し、Java Bean サービスに関する統合リポジトリのアノテーション標準に基づいて注釈付けする必要があります。インタフェースの作成後、統合管理者は、注釈付けされた API を検証する必要があります。検証中にエラーが発生しなかった場合、管理者はカスタム API を Oracle Integration Repository にアップロードします。そこで、Oracle E-Business Suite Integrated SOA Gateway を介して、それらを REST サービスとして公開できます。

アノテーションの詳細は、「Java Bean サービスに対するアノテーション」(ページ A-6)を参照してください。

カスタム Java Bean サービスへの注釈付けと Javascript を含む HTML ページからの起動

この項には、REST サービスの起動シナリオに基づく次のタスクが含まれています。

1. カスタム Java API の作成およびコンパイル、ページ 4-17
2. カスタム Java クラスおよびソース・ファイルの配置、ページ 4-34
3. 注釈付けされたカスタム Java Bean サービスの解析と統合リポジトリへのアップロード、ページ 4-34
4. カスタム Java Bean サービスの配置、ページ 4-36
5. セキュリティ付与の作成、ページ 4-37
6. 配置済 WADL からのリソース情報の記録、ページ 4-38
7. Javascript を含む HTML ページからのカスタム REST サービスの起動、ページ 4-41

カスタム Java API の作成およびコンパイル

この項では、Employee Information

(`oracle.apps.per.sample.service.EmployeeInfo`)というカスタム Java Bean サービスを作成し、注釈付けする方法について説明します。

カスタム Java API の開発に関するガイドライン

計画段階では、次のガイドラインを使用して、Oracle E-Business Suite Integrated SOA Gateway を介して REST サービスとして公開されるカスタム Java API を計画および開発してください。

1. `public` メソッドがビジネス機能を提供する Java クラスを開発します。これらの `public` メソッドにビジネス・ロジックを埋め込む必要があります。

- カスタム Java API およびメソッド名が Web サービス URL で使用されます。そのため、必ず、わかりやすい名前を指定してください。
- 初期化された Oracle E-Business Suite コンテキストが Java メソッド内で必要な場合は、必ず、次のようにしてください。

- ISG コンテキスト(`IContext`)を使用して、ランタイム情報へのハンドルを取得します。

例: `IContext ctx = ContextManager.getContext();`

- 要求ヘッダー/セキュリティ・トークンに基づいて、完全に初期化された Oracle E-Business Suite `WebAppsContext` を取得します。

例: `WebAppsContext wctx = (WebAppsContext) ctx.getExternalContext();`

`WebAppsContext` は、ファイングレイイン・アクセス管理など、アプリケーション固有の検証に後で使用できます。

- Oracle E-Business Suite Integrated SOA Gateway データベース接続を Java API 内で使用できます。

例: `conn = DBConnectionManager.getConnection();`

- Oracle E-Business Suite Integrated SOA Gateway では、インフラストラクチャから標準の例外処理を実行できます。エラー条件が発生した場合は、常に、Java API によってスロー可能例外がスローまたは再スローされるようにしてください。

2. 前述の Java メソッドで複雑なデータ・オブジェクトを入力および出力パラメータとして交換する必要がある場合は、Java Bean を開発します。

次のような Java Bean にしてください。

- `java.io.Serializable` を実装します。
- 引数なしのコンストラクタがあります。
- プライベート属性について、`get` および `set` 命名規則に準拠したアクセサ・メソッドがあります。

カスタム Java API の作成

この例では、ログイン・ユーザーの階層に属する特定の個人の詳細を返すとともに、ログイ

ン・ユーザーのすべての部下を返すサービスが必要です。

この目標を達成するには、次の Java ファイルを作成します。

- 次の 3 つのメソッドを含む Java クラス `EmployeeInfo.java`。シナリオに基づいて、これらは読み取り専用メソッドであり、それぞれの Java メソッドを HTTP GET 動詞にマップします。

- `getAllReports`: このメソッドは、要求ユーザーのすべての部下の配列を返します。

- `getDirectReports`: このメソッドは、要求ユーザーの直属の部下のリストを返します。

- `getPersonInfo`: このメソッドは、特定の個人 ID の個人詳細を返します。

このメソッドには、個人 ID のキー識別子パラメータが必要です。そのため、`personId` に `key_param` として注釈付けします。

```
...
* @param personId Person Identifier
* @rep:paraminfo {@rep:required} {@rep:key_param}
...
```

- Oracle E-Business Suite Integrated SOA Gateway の `ServiceException` を拡張する Java クラス `PerServiceException.java`。

- 個人情報を取得する Java Bean `PersonBean.java`。

カスタム Java API を作成してコンパイルするには:

次のステップを使用して、カスタム Java API を作成し、コンパイルします。

1. Oracle JDeveloper を開きます。

2. メイン・メニューから「ファイル」→「新規」を選択します。

「新規ギャラリー」ウィンドウで、「一般」カテゴリを展開して「アプリケーション」を選択します。「アイテム」リストで「**カスタム・アプリケーション**」を選択します。

「OK」をクリックします。「カスタム・アプリケーションの作成 - アプリケーションの名前付け」ページが表示されます。

3. 「アプリケーション名」フィールドにアプリケーションの適切な名前を入力します。「次へ」をクリックします。

4. 「カスタム・アプリケーションの作成 - プロジェクトの名前付け」ページが表示されます。

「プロジェクト名」フィールドにプロジェクトの適切な名前を入力します(例:

`ISGJava_RESTDemo`)。

「プロジェクトの機能」タブで、「使用可能」リストから「**Java**」を選択します。右矢印ボタンを使用して、「使用可能」ウィンドウから「選択済」ウィンドウに、選択した機能を移動します。

「次へ」をクリックします。

5. 「Java 設定の構成」ダイアログ・ボックスで「終了」をクリックします。

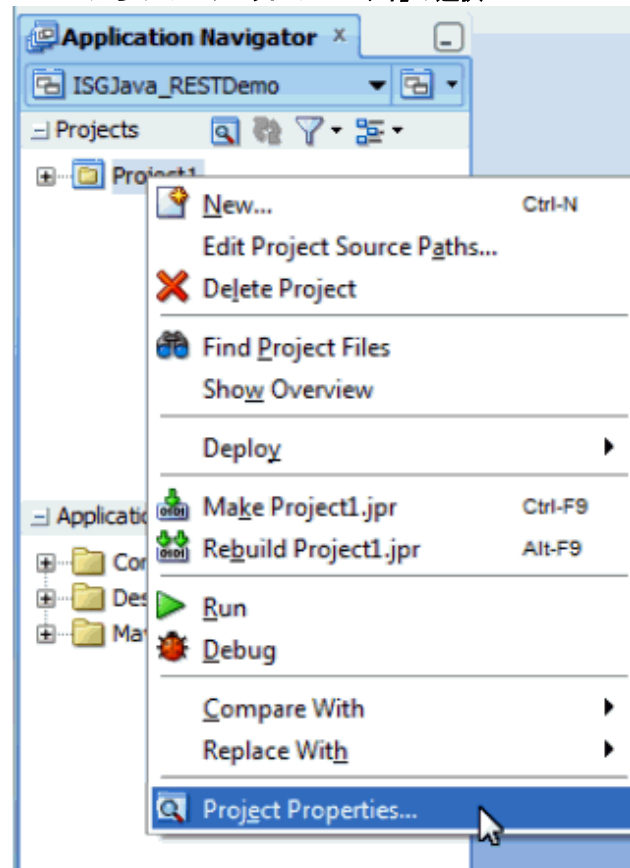
新しく作成したプロジェクトが「プロジェクト」ワークスペースに表示される必要があります。

6. 必要なライブラリを追加します。

次のステップを使用して、必要なライブラリ・ファイルをプロジェクト・プロパティに追加します。

1. 先ほど作成したプロジェクトの名前「ISGJava_RESTDemo」を選択して右クリックし、選択メニューを開きます。
2. メニューから「プロジェクト・プロパティ」を選択します。

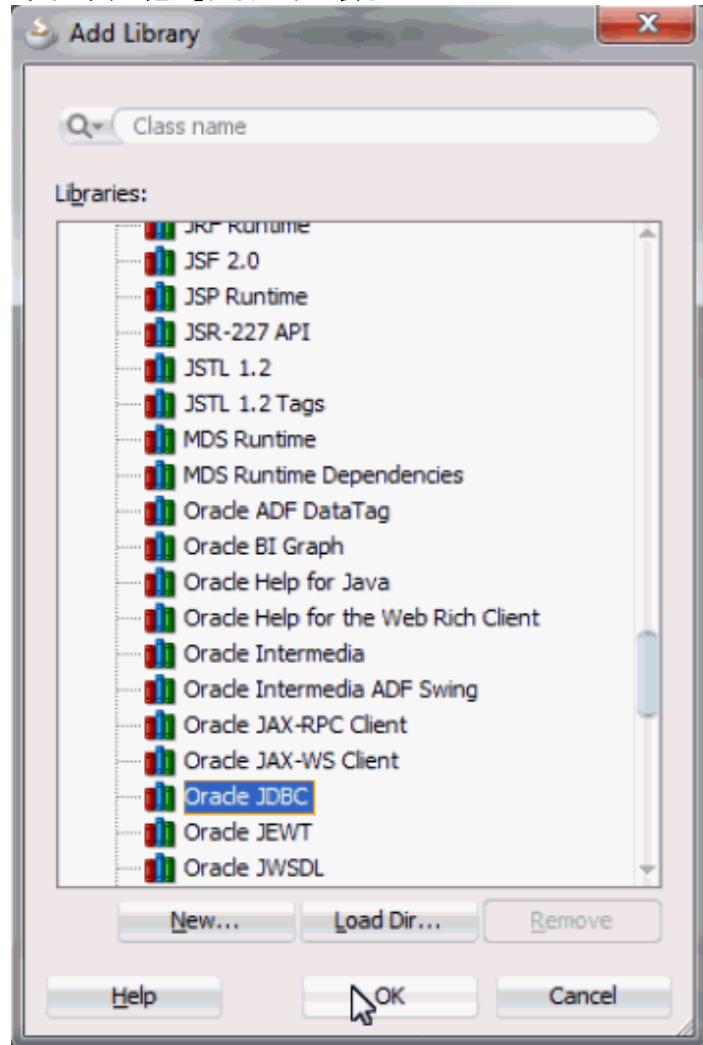
メニューからの「プロジェクト・プロパティ」の選択



「デフォルト・プロパティ」ダイアログ・ボックスが開きます。

3. 「ライブラリとクラスパス」を選択して、「ライブラリの追加」をクリックします。「ライブラリの追加」ダイアログ・ボックスが開きます。

「ライブラリの追加」ダイアログの表示



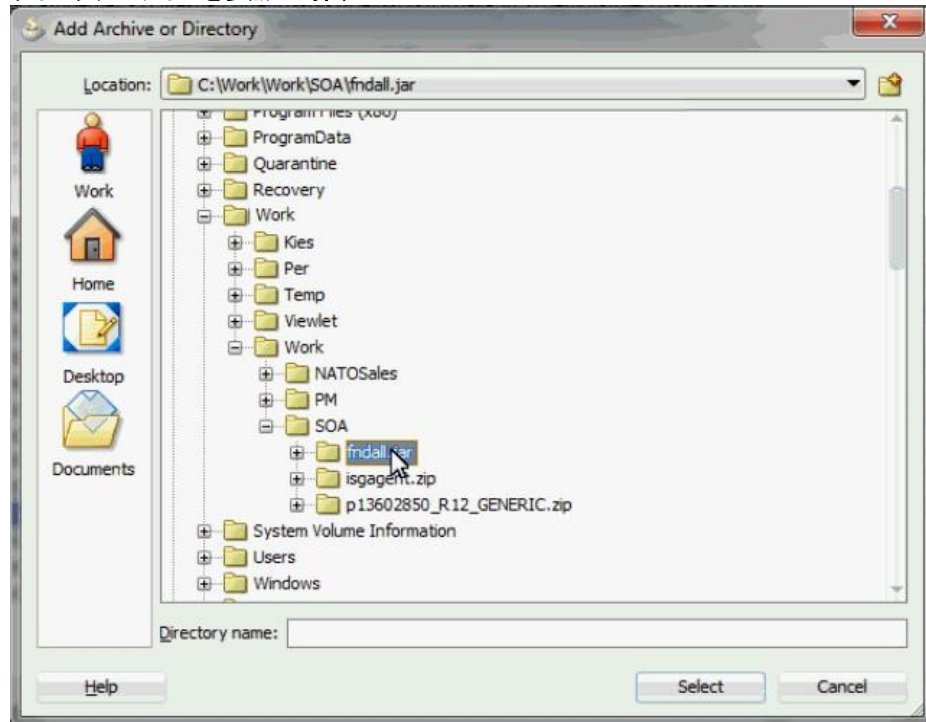
4. 「ライブラリの追加」ダイアログ・ボックスで、「拡張」フォルダを選択して展開し、リストから「Oracle JDBC」を選択します。

「OK」をクリックします。Oracle JDBC が「クラスパス・エントリ」セクションに追加されます。

5. 「JAR/ディレクトリの追加」をクリックします。「アーカイブまたはディレクトリの追加」ダイアログ・ボックスが表示されます。

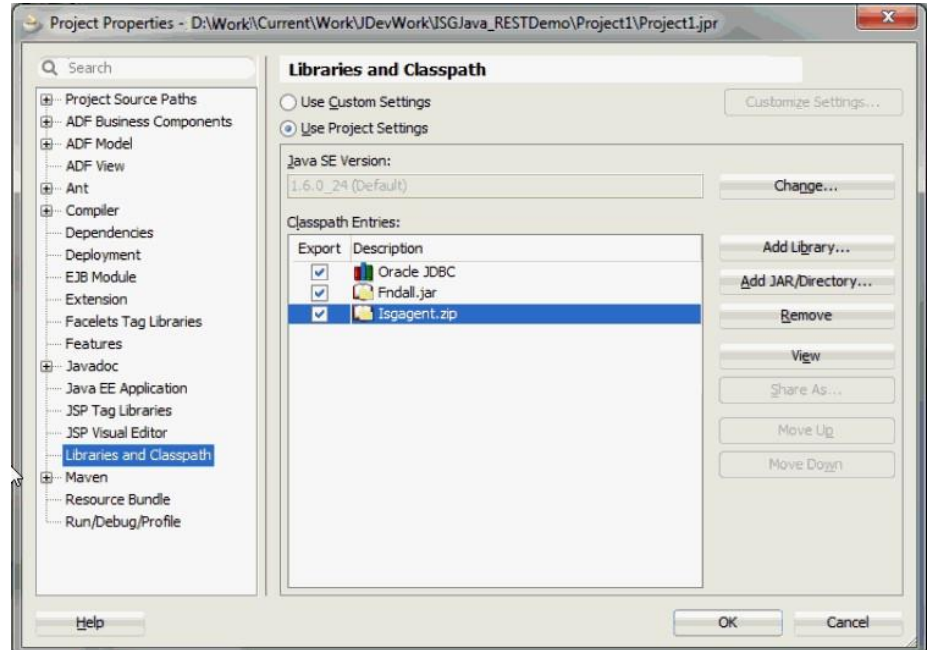
ディレクトリを参照し、fndall.jar ファイルを探します。「選択」をクリックして、選択した fndall.jar ファイルを追加します。

ライブラリ・ファイルを参照して探す



同じアプローチを使用して、「プロジェクト・プロパティ」ダイアログの「クラスパス・エントリ」リージョンに isgagent.zip ファイルを追加します。

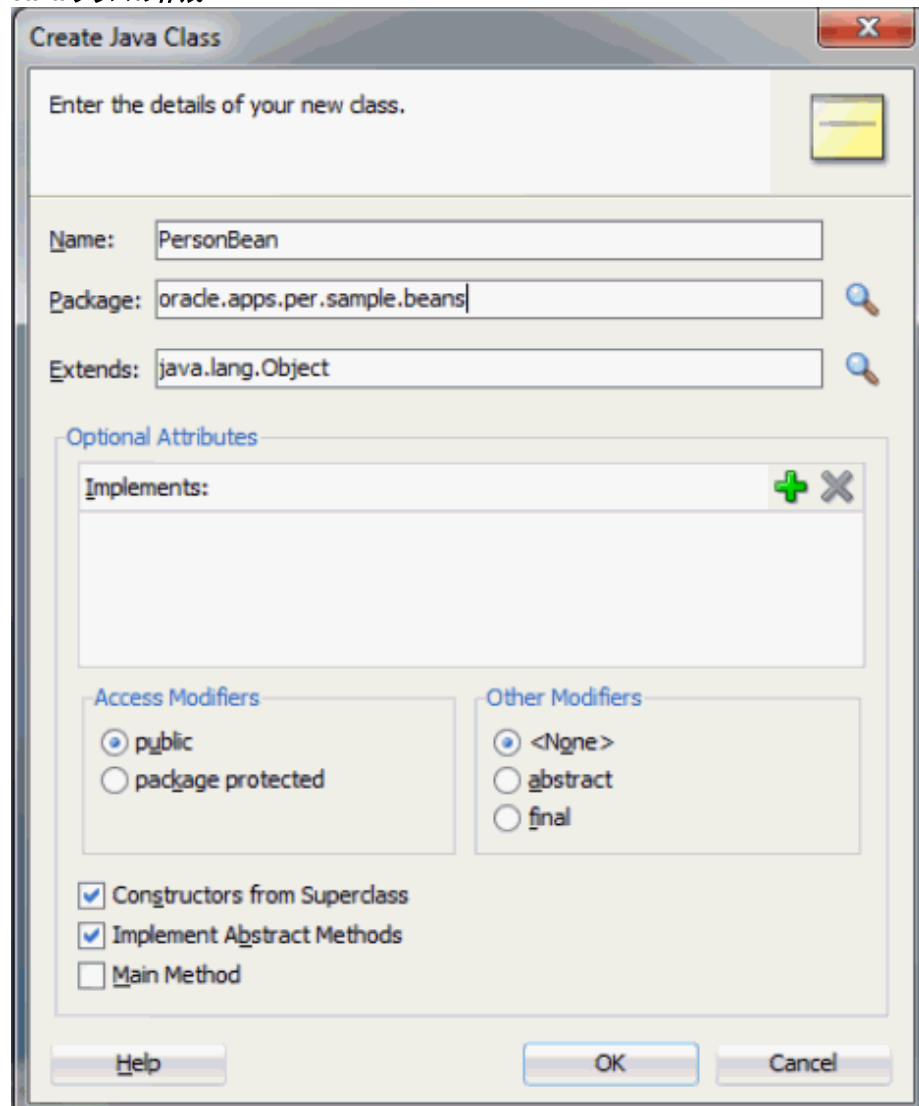
追加されたライブラリの確認



「OK」をクリックします。このプロジェクトが、必要なライブラリで設定されました。

7. 作成したばかりのプロジェクトの名前を選択して右クリックし、ドロップダウン選択メニューから「新規」を選択します。
8. 「新規ギャラリー」ウィンドウで、「一般」カテゴリを展開して「Java」を選択します。「アイテム」リストで「Java クラス」を選択します。「OK」をクリックします。
9. 「Java クラスの作成」ダイアログで、次の情報を入力して `PersonBean.java` という Java クラスを作成します。
 - 名前: `PersonBean`
 - パッケージ: `oracle.apps.per.sample.beans`
 - 拡張: `java.lang.Object`

Java クラスの作成



「OK」をクリックします。ソース・エディタで新しいクラスが自動的に開きます。
PersonBean.java に次の Java コードを使用します。

```

package oracle.apps.per.sample.beans;

import java.io.Serializable;
import java.util.Date;

public class PersonBean implements Serializable {
    private static final long serialVersionUID = 1L;
    private int personId;
    private String firstName;
    private String lastName;
    private String salutaion;
    private String fullName;
    private String nameSuffix;
    private String emailAddress;
    private String employeeNumber;
    private String workPhone;
    private Date startDate;
    private int reportingLevel;
    private int supervisorPersonId;
    private String supervisorEmployeeNumber;
    private String supervisorFullName;
    private String reportingHierarchy;

    public void setPersonId(int personId) {
        this.personId = personId;
    }

    public int getPersonId() {
        return personId;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getFirstName() {
        return firstName;
    }
}

```

```

public void setLastName(String lastName) {
    this.lastName = lastName;
}

public String getLastName() {
    return lastName;
}

public void setSalutaion(String salutaion) {
    this.salutaion = salutaion;
}

public String getSalutaion() {
    return salutaion;
}

public void setFullName(String fullName) {
    this.fullName = fullName;
}

public String getFullName() {
    return fullName;
}

public void setEmailAddress(String emailAddress) {
    this.emailAddress = emailAddress;
}

public String getEmailAddress() {
    return emailAddress;
}

public void setEmployeeNumber(String employeeNumber) {
    this.employeeNumber = employeeNumber;
}

public String getEmployeeNumber() {
    return employeeNumber;
}

public void setStartDate(Date startDate) {
    this.startDate = startDate;
}

public Date getStartDate() {
    return startDate;
}

public void setNameSuffix(String nameSuffix) {
    this.nameSuffix = nameSuffix;
}

public String getNameSuffix() {
    return nameSuffix;
}

public void setWorkPhone(String workPhone) {
    this.workPhone = workPhone;
}

public String getWorkPhone() {
    return workPhone;
}

public void setReportingLevel(int reportingLevel) {
    this.reportingLevel = reportingLevel;
}

public int getReportingLevel() {
    return reportingLevel;
}

public void setSupervisorPersonId(int supervisorPersonId) {

```

```

this.supervisorPersonId = supervisorPersonId;
}

public int getSupervisorPersonId() {
    return supervisorPersonId;
}

public void setSupervisorEmployeeNumber(String
supervisorEmployeeNumber) {
    this.supervisorEmployeeNumber = supervisorEmployeeNumber;
}

public String getSupervisorEmployeeNumber() {
    return supervisorEmployeeNumber;
}

public void setSupervisorFullName(String supervisorFullName) {
    this.supervisorFullName = supervisorFullName;
}

public String getSupervisorFullName() {
    return supervisorFullName;
}

public void setReportingHierarchy(String reportingHierarchy) {
    this.reportingHierarchy = reportingHierarchy;
}

    public String getReportingHierarchy() { return
reportingHierarchy;
}
}

```

10. ステップ 7、8 および 9 を繰り返して、次の 2 つの Java クラスを作成します。

- PerServiceException.java については、次の情報を入力します。
 - 名前: PerServiceException
 - パッケージ: oracle.apps.per.sample.common
 - 拡張: java.lang.Object

PerServiceException.java に次の Java コードを使用します。

```

package oracle.apps.per.sample.common;

import oracle.apps.fnd.isg.app.ebs.rt.common.ServiceException;

public class PerServiceException extends ServiceException {

    public PerServiceException(String message) {
        super(message);
    }

    public PerServiceException(Throwable t) {
        super(t);
    }

    public PerServiceException(String message, Throwable t) {
        super(message, t);
    }

    public PerServiceException(String errorCode, String message,
        Throwable t) {
        super(errorCode, message, t);
    }
}

```

- EmployeeInfo.java については、次の情報を入力します。
 - 名前: EmployeeInfo
 - パッケージ: oracle.apps.per.sample.service
 - 拡張: java.lang.Object

EmployeeInfo.java に次の Java コードを使用します。

```

package oracle.apps.per.sample.service;

import java.sql.Connection;
import java.sql.SQLException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import oracle.apps.fnd.common.WebAppsContext;
import oracle.apps.fnd.isg.common.IContext;
import oracle.apps.fnd.isg.common.mgr.ContextManager;
import oracle.apps.fnd.isg.common.mgr.DBConnectionManager;
import oracle.apps.per.sample.beans.PersonBean;
import oracle.apps.per.sample.common.PerServiceException;

import oracle.jdbc.OraclePreparedStatement;
import oracle.jdbc.OracleResultSet;

/**
 * A sample class to demonstrate how Java API can use the ISG
 * REST framework. This class provides
 * methods to retrieve list of direct reports, all reports of a
 * person. It also has methods to
 * retrieve personal details and accrual balance of a person.
 * @rep:scope public
 * @rep:product PER
 * @rep:displayname Employee Information
 * @rep:category IREP_CLASS_SUBTYPE JAVA_BEAN_SERVICES
 */
public class EmployeeInfo {

    public EmployeeInfo() {
        super();
    }

    /**
     * This method returns a list of direct reports of the
     * requesting user.
     *
     * @return List of person records who are direct reports
     * @rep:paraminfo {@rep:innertype oracle.apps.per.sample.
     beans.PersonBean}
     * @rep:scope public
     * @rep:displayname Get Direct Reports
     * @rep:httpverb get
     * @rep:category BUSINESS_ENTITY sample
     */
    // Demonstration of list return type
    public List<PersonBean> getDirectReports() throws
    PerServiceException {

        // Get the ISG context, which has runtime information
        IContext ctx = ContextManager.getContext();
        // Retrieve fully initialized webappscontext, as per the
        request header / security token
        WebAppsContext wctx = (WebAppsContext) ctx.
        getExternalContext();

        // Use webappscontext for apps specific validation e.g.
        fine grained access control etc.
        int userId = wctx.getUserId();
        String userName = wctx.getUserName().toUpperCase();

```

```

        // Here we intend to filter based on the specific user who
is invoking the service
        String filter = " AND LEVEL = 2 START WITH USER_NAME = :1";

        List<PersonBean> personList = null;
        try {
            Map<Integer, PersonBean> map = makeBean(filter, userName);
            if (map != null)
                personList = new ArrayList<PersonBean>(map.values());
        } catch (SQLException sqle) {
            throw new PerServiceException("SQL error while getting the
all reports", sqle);
        }

        return(personList);
    }
}
/**
 * This method returns an array of all reports of the
requesting user.
 *
 * @return Array of person records who are reporting into
the requesting user's organization hierarchy
 * @rep:scope public
 * @rep:displayname Get All Reports
 * @rep:httpverb get
 * @rep:category BUSINESS_ENTITY sample
 */
// Demonstration of array return type
public PersonBean[] getAllReports() throws PerServiceException {

    // Get the ISG context, which has runtime information
    IContext ctx = ContextManager.getContext();
    // Retrieve fully initialized webappscontext, as per the
request header / security token
    WebAppsContext wctx = (WebAppsContext) ctx.
getExternalContext();

    // Use webappscontext for apps specific validation e.g.
fine grained access control etc.
    int userId = wctx.getUserId();
    String userName = wctx.getUserName().toUpperCase();

    // Here we intend to filter based on the specific user who
is invoking the service
    String filter = " START WITH USER_NAME = :1 ";

    PersonBean[] array = null;

    try {
        Map<Integer, PersonBean> map = makeBean(filter, userName);
        if (map != null)
            array = map.values().toArray(new PersonBean[map.size()]);
    } catch (SQLException sqle) {
        throw new PerServiceException("SQL error while getting
the direct reports", sqle);
    }

    return(array);
}
}
/**
 * This method returns the person details for a specific
person id. Throws error if the person

```



```

* is not in requesting user's org hierarchy.
*
* @return Details of a person in the logged on user's org
hierarchy.
* @param personId Person Identifier
* @rep:paraminfo {@rep:required} {@rep:key_param}
* @rep:scope public
* @rep:displayname Get Person Details
* @rep:httpverb get
* @rep:category BUSINESS_ENTITY sample
*/
// Demonstration of simple navigation using path param public
PersonBean getPersonInfo(int personId) throws
PerServiceException {

    // Get the ISG context, which has runtime information
    IContext ctx = ContextManager.getContext();
    // Retrieve fully initialized webappscontext, as per the
request header / security token
    WebAppsContext wctx = (WebAppsContext) ctx.
getExternalContext();

    // Use webappscontext for apps specific validation e.g.
fine grained access control etc.
    int userId = wctx.getUserId();
    String userName = wctx.getUserName().toUpperCase();

    String filter = " START WITH USER_NAME = :1 ";
    Map<Integer, PersonBean> map = null;
    try {
        map = makeBean(filter, userName);
    } catch (SQLException sqle) {
        throw new PerServiceException("SQL error while getting
the direct reports", sqle);
    }

    if (map == null)
        throw new PerServiceException(PerServiceException.
AUTHORIZATION_FAILURE, "No org hierarchy found for user - " +
userName, null);

    boolean doesExist = map.containsKey(personId); if
(!doesExist)
        throw new PerServiceException(PerServiceException.
AUTHORIZATION_FAILURE, "The given person " + personId + "
either does not exist or does not belong to the current user's
- " + userName + " hierarchy", null);

    PersonBean bean = map.get(personId);

    return(bean);

}

/***** private members *****/

private Map<Integer, PersonBean> makeBean(String sqlFilter,
String userName) throws SQLException {

    Connection conn = null;
    OraclePreparedStatement stmt = null;
    OracleResultSet rs = null;
    HashMap<Integer, PersonBean> personMap = new
HashMap<Integer, PersonBean>();

```

```

IContext ctx = ContextManager.getContext();

try {
    String sql = GET_PERSON_INFO1 + sqlFilter;

    // preferred way of obtaining connection, rather than
    from WebAppsContext
    conn = DBConnectionManager.getConnection();
    stmt = (OraclePreparedStatement) conn.prepareStatement
(sql);
    stmt.setString(1, userName);
    rs = (OracleResultSet) stmt.executeQuery();

    while (rs != null && rs.next()) {

        int personId = rs.getInt(1);
        String firstName = rs.getString(2);
        String lastName = rs.getString(3);
        String title = rs.getString(4);
        String fullName = rs.getString(5);
        String nameSuffix = rs.getString(6);
        String empNumber = rs.getString(7);
        String emailAddr = rs.getString(8);
        String workPhone = rs.getString(9);
        int supervisorId = rs.getInt(10);
        java.util.Date startDate = rs.getDate(12);
        int level = rs.getInt(16);
        String supervisorName = rs.getString(17);
        String supervisorEmpNumber = rs.getString(18);
        String reporting = rs.getString(19);

        // Do any data validation / transformation if
        necessary, here

        PersonBean personBean = new PersonBean();

        // Set the information into the serializable bean
        personBean.setPersonId(personId);
        personBean.setFirstName(firstName);
        personBean.setLastName(lastName);
        personBean.setSalutaion(title);
        personBean.setFullName(fullName);
        personBean.setNameSuffix(nameSuffix);
        personBean.setEmployeeNumber(empNumber);
        personBean.setEmailAddress(emailAddr);
        personBean.setWorkPhone(workPhone);
        personBean.setSupervisorPersonId(supervisorId);
        personBean.setStartDate(startDate);
        personBean.setReportingLevel(level);
        personBean.setSupervisorFullName(supervisorName);
        personBean.setSupervisorEmployeeNumber
(supervisorEmpNumber);
        personBean.setReportingHierarchy(reporting);

        personMap.put(personId, personBean);

    }
} finally {
    if (rs != null) {
        try { rs.close(); } catch (Exception e) {};
        rs = null;
    }
}

```

```

        if (stmt != null) {
            try { stmt.close(); } catch (Exception e) {};
            stmt = null;
        }

        // preferred way of closing the connection
        DBConnectionManager.closeConnection(conn);
    }

    return (personMap);
}

private static final String GET_PERSON_INFO1 = "SELECT
DISTINCT E.PERSON_ID, E.FIRST_NAME, E.LAST_NAME, E.TITLE, E.
FULL_NAME, E.SUFFIX,\n" +
                                                "E.EMPLOYEE_NUMBER, E.
EMAIL_ADDRESS, E.WORK_TELEPHONE, E.SUPERVISOR_ID,\n" +
                                                "E.
SUPERVISOR_ASSIGNMENT_ID, E.EFFECTIVE_START_DATE, E.
EFFECTIVE_END_DATE,\n" +
                                                "E.USER_ID, E.
USER_NAME, LEVEL, PRIOR E.FULL_NAME, PRIOR E.EMPLOYEE_NUMBER,\n"
+
                                                "SYS_CONNECT_BY_PATH(E.
FULL_NAME, '/')\n" +
\n" +
                                                "PPF.FIRST_NAME,\n" +
                                                "PPF.LAST_NAME,\n" +
                                                "PPF.TITLE,\n" +
                                                "PPF.FULL_NAME,\n" +
                                                "PPF.SUFFIX,\n" +
                                                "PPF.EMPLOYEE_NUMBER,
\n" +
                                                "PPF.EMAIL_ADDRESS,\n"
+
                                                "PPF.WORK_TELEPHONE,\n"
+
                                                "PAF.SUPERVISOR_ID,\n"
+
                                                "PAF.
SUPERVISOR_ASSIGNMENT_ID,\n" +
                                                "GREATEST (PPF.
EFFECTIVE_START_DATE, PAF.EFFECTIVE_START_DATE)
EFFECTIVE_START_DATE, \n" +
                                                "LEAST (PPF.
EFFECTIVE_END_DATE, PAF.EFFECTIVE_END_DATE)
EFFECTIVE_END_DATE,\n" +
                                                "FUR.USER_ID,\n" +
                                                "FUR.USER_NAME\n" +
                                                "FROM FND_USER FUR,
PER_ALL_PEOPLE_F PPF,\n" +
                                                "PER_ALL_ASSIGNMENTS_F
PAF\n" +
                                                "WHERE PPF.PERSON_ID =
PAF.PERSON_ID (+)\n" +
                                                "AND PPF.
BUSINESS_GROUP_ID = PAF.BUSINESS_GROUP_ID (+)\n" +
                                                "AND (SYSDATE BETWEEN
PPF.EFFECTIVE_START_DATE\n" +
                                                "AND PPF.
EFFECTIVE_END_DATE\n" +

```

```

"AND SYSDATE BETWEEN PAF.EFFECTIVE_START_DATE\ " +
"AND PAF.EFFECTIVE_END_DATE)\ " +
"AND FUR.EMPLOYEE_ID =
PPF.PERSON_ID\ " +
") E\ " +
"CONNECT BY
SUPERVISOR_ID = PRIOR PERSON_ID";
}

```

11. 「ファイル」→「すべて保存」を選択して、作業内容を保存します。
12. プロジェクトの名前「ISGJava_RESTDemo」を選択して右クリックし、ドロップダウン選択メニューから「**Project1.jpr のメイク**」を選択します。コンパイル・プロセスが開始されます。
「メッセージ - ログ」ウィンドウで、コンパイル・プロセスが成功したことを確認します。

カスタム Java クラスおよびソース・ファイルの配置

カスタム Java API が正常に作成されたら、統合開発者は、新しく作成されたこれらの Java クラスおよびソース・ファイルをターゲット・インスタンスにコピーする必要があります。

これらのカスタム Java クラスを配置する方法の詳細は、My Oracle Support のナレッジ・ドキュメント 1577661.1『Deploying Customizations in Oracle E- Business Suite Release 12.2』の第 5 項「Deploying Custom Application Tier Objects」を参照してください。

ターゲット・インスタンスに正常に配置したら、管理対象サーバーを停止して再起動します。

注釈付けされたカスタム Java Bean サービスの解析と統合リポジトリへのアップロード

カスタム Java クラスがターゲット・インスタンスに正常に配置されたら、統合管理者は、統合リポジトリ・パーサーを使用して、注釈付けされたカスタム・インタフェース定義 Employee Information を Java Bean サービスのアノテーション標準に照らして検証する必要があります。検証中にエラーが発生しなかった場合、iLDT ファイルが生成されます。その後、管理者は、生成された iLDT ファイルを Oracle Integration Repository にアップロードします。

次のステップを実行して、注釈付けされたカスタム・インタフェース定義 Employee Information を解析し、Oracle Integration Repository にアップロードします。

1. RUN ファイル・システムの環境をソースとして使用し、カスタム Java API で使用されるすべてのライブラリおよび JAR ファイルを含むように CLASSPATH を設定します。
2. 注釈付けされたカスタム・インタフェース定義および関連する Java クラスがターゲット・インスタンスに配置されていることを確認します。

注釈付けされたカスタム・インタフェース定義の確認

```
-bash-3.2$ source /u01/R122_EBS/fs1/EBSapps/appl/isgl224_rws3270149.env
-bash-3.2$ export JAVA_HOME=/u01/R122_EBS/fs1/EBSapps/comn/util/jdk32
-bash-3.2$ export CLASSPATH=/u01/R122_EBS/fs1/EBSapps/comn/java/lib/lsagent.zip:$CLASSPATH
-bash-3.2$ export CLASSPATH=/u01/R122_EBS/fs1/EBSapps/comn/clone/jlib/ojdbc6.jar:$CLASSPATH
-bash-3.2$ cd $JAVA_TOP
-bash-3.2$ cd oracle/apps/per/sample/service/
-bash-3.2$ ll
total 84
-rw-r--r-- 1 oracle dba 8358 Jul 3 11:20 EmployeeInfo.class
-rw-r--r-- 1 oracle dba 11463 Jul 3 11:21 EmployeeInfo.java
-rw-r--r-- 1 oracle dba 1281 Jul 3 04:01 L7572911.log
-rw-r--r-- 1 oracle dba 15191 Jul 2 05:36 PersonResource.class
-rw-r--r-- 1 oracle dba 19311 Jul 2 23:33 PersonResource.java
-rw-r--r-- 1 oracle dba 17587 Jul 3 03:51 PersonResource_java.ildt
-bash-3.2$
```

3. 次のコマンドを使用して統合リポジトリ・パーサーを実行し、注釈付けされたカスタム・インタフェース定義をアノテーション標準に照らして検証します。

```
$IAS_ORACLE_HOME/perl/bin/perl $FND_TOP/bin/irep_parser.pl -g -v -username=sysadmin
per:patch/115/java:EmployeeInfo.java:
12.2=EmployeeInfo.java
```

検証のための統合リポジトリ・パーサーの実行

```
-bash-3.2$ $IAS_ORACLE_HOME/perl/bin/perl $FND_TOP/bin/irep_parser.pl -g -v -username=sysadmin per:patch/115/java:EmployeeInfo.java:12.2=EmployeeInfo.java
# Interface Repository Annotation Processor, 12.0.0
#
# Generating annotation output.
# Processing file 'EmployeeInfo.java'.
# Using YAPP-based parser.
# Warning: Carriage return(s) seen in text.
# Java code is in package 'oracle.apps.per.sample.service'.
# Defining the class 'EmployeeInfo'.
# Found a class annotation for 'EmployeeInfo'.
# Found a member annotation for the method named getDirectReports
# Found a member annotation for the method named getAllReports
# Found a member annotation for the method named getPersonInfo
# Javadoc at line 158 with no IR annotation tags ignored.
# Done all files.
-bash-3.2$ ll
total 100
-rw-r--r-- 1 oracle dba 8358 Jul 3 11:20 EmployeeInfo.class
-rw-r--r-- 1 oracle dba 11463 Jul 3 11:21 EmployeeInfo.java
-rw-r--r-- 1 oracle dba 14765 Jul 3 11:34 EmployeeInfo_java.ildt
-rw-r--r-- 1 oracle dba 1281 Jul 3 04:01 L7572911.log
-rw-r--r-- 1 oracle dba 15191 Jul 2 05:36 PersonResource.class
-rw-r--r-- 1 oracle dba 19311 Jul 2 23:33 PersonResource.java
-rw-r--r-- 1 oracle dba 17587 Jul 3 03:51 PersonResource_java.ildt
-bash-3.2$ clear
```

検証後、EmployeeInfo_java.ildt ファイルが正常に生成されたことを確認します。

4. 次の FNDLOAD コマンドを使用して、生成された EmployeeInfo_Java.ildt ファイルを統合リポジトリにアップロードします。

```
$FND_TOP/bin/FNDLOAD <APPS username> @TWO_TASK 0 Y UPLOAD
$FND_TOP/patch/115/import/wfirep.lct EmployeeInfo_java.ildt ORACLE
Password:
```

5. 生成されたログ・ファイルを検証して、アップロードの詳細を確認します。

アップロードの詳細の確認

```
+-----+
Application Object Library: Version : 12.2

Copyright (c) 1998, 2013, Oracle and/or its affiliates. All rights reserved.

FNDLOAD: Generic Loader
+-----+

Current system time is 03-JUL-2014 11:36:46

+-----+

Uploading from the data file EmployeeInfo_java.ildt
Altering database NLS_LANGUAGE environment to AMERICAN
Dump from LCT/LDT files (/u01/R122_EBS/fs1/EBSSapps/appl/fnd/12.0.0/patch/115/import/wfirep.lct(120.8.12020000.3), EmployeeIn
fo_java.ildt) to stage tables
Dump LCT file /u01/R122_EBS/fs1/EBSSapps/appl/fnd/12.0.0/patch/115/import/wfirep.lct(120.8.12020000.3) into FND_SEED_STAGE_CO
NFIG
Dump LDT file EmployeeInfo_java.ildt into FND_SEED_STAGE_ENTITY
Dumped the batch (IREP_OBJECT JAVA:oracle.apps.per.sample.service.EmployeeInfo C , PARAMS 0 0 ) into FND_SEED_STAGE_ENTITY
Upload from stage tables

+-----+

Concurrent request completed successfully
Current system time is 03-JUL-2014 11:36:52

+-----+
/
```

アップロード・プロセスが正常に完了したことを確認します。

- アップロードしたカスタム Java Bean サービス・インタフェース「Employee Information」を統合リポジトリで検索します。

Employee Information の名前リンクをクリックして、インタフェースの詳細ページを開きます。

カスタム Java Bean サービスの配置

次のステップを使用して、カスタム・インタフェースを REST サービスとして配置します。

- 「統合管理者」ロールを持つユーザーとして、Oracle E-Business Suite にログインします。
職責「統合 SOA ゲートウェイ」を使って、ナビゲーション・メニューから「統合リポジトリ」リンクを選択します。
- 「統合リポジトリ」タブで、「検索」をクリックしてメインの「検索」ページにアクセスします。
- 「詳細検索オプションの表示」をクリックして、詳細検索オプションを表示します。「インタフェース・ソース」フィールドで「カスタム」を選択します。
- 「進む」をクリックして、検索を実行します。

Employee Information のインタフェース名リンクをクリックして、インタフェースの詳細ページを開きます。

- 「REST Web サービス」タブで、次の情報を入力します。

- サービス別名: empinfo

この別名が、WADL のサービス・エンドポイントと、選択されたメソッドまたはオペレーションのスキーマとして利用されます。

- Get Person Details Java メソッドの「POST」チェック・ボックスを選択します。事前選択されている残りのチェック・ボックスは変更しません。

6. 「**配置**」をクリックして、Oracle E-Business Suite WebLogic 環境にサービスを配置します。

REST サービスが正常に配置されたら、「**WADL の表示**」リンクとともに、「REST サービス・ステータス」フィールドに「配置済」が表示されます。「**WADL の表示**」リンクをクリックして、配置されたサービス WADL の記述を表示します。

REST サービスの配置の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「REST Web サービスの配置」を参照してください。

セキュリティ付与の作成

カスタム・インタフェース「Employee Information」を REST サービスとして配置したら、統合管理者はセキュリティ付与を作成して、サービスのアクセス権限をすべてのユーザーに与えることができます。

次のステップを使用して、セキュリティ付与を作成します。

1. 「統合管理者」ロールを持つユーザーとして、Oracle E-Business Suite にログインします。職責「統合 SOA ゲートウェイ」を使って、ナビゲーション・メニューから「統合リポジトリ」リンクを選択します。
2. 検索を実行して、管理者が先ほど配置した Employee Information サービスを探します。
3. 検索結果の表から Employee Information の名前リンクを選択して、インタフェースの詳細ページを表示します。「付与」タブをクリックします。
4. Employee Information サービスを選択し、「**付与の作成**」をクリックします。
5. 「付与の作成」ページで、「被付与者タイプ」として「全ユーザー」を選択します。

注意: この例では、Employee Information サービスはすべてのユーザーに許可されます。実際の実装では、厳格なセキュリティ・ルールを定義する必要があります。ユーザーに対する付与、またはより適切な方法として、ロールおよび職責によって定義されたユーザーのグループに対する付与を作成します。

「**付与の作成**」をクリックします。これにより、選択したサービスのアクセス権限がすべての Oracle E-Business Suite ユーザーに付与されます。

配置済 WADL からのリソース情報の記録

Employee Information カスタム・サービスの配置済 WADL からサービス・リソース情報を取得するには、「REST Web サービス」タブで「**WADL の表示**」リンクをクリックします。

次の WADL 記述が表示されます。


```

<xml version="1.0" encoding="UTF-8">
<application name="EmployeeInfo" targetNamespace="http://xmlns.oracle.
com/apps/per/soapprovider/pojo/employeeinfo/"
  xmlns:tns="http://xmlns.oracle.
com/apps/per/soapprovider/pojo/employeeinfo/"
  xmlns="http://w3.org/2001/XMLSchema"
  xmlns:tns1="http://xmlns.oracle.
com/apps/fnd/rest/empinfo/getallreports/"
  xmlns:tns2="http://xmlns.oracle.
com/apps/fnd/rest/empinfo/getdirectreports/"
  xmlns:tns3="http://xmlns.oracle.
com/apps/fnd/rest/empinfo/getpersoninfo/">

<grammars>
  <include href="http://<hostname>:<port>/web services/rest/empinfo/?
XSD=getallreports.xsd" xmlns="http://www.w3.org/2001/XMLSchema" />
  <include href="http://<hostname>:<port>/web services/rest/empinfo/?
XSD=getdirectreports.xsd" xmlns="http://www.w3.org/2001/XMLSchema" />
  <include href="http://<hostname>:<port>/web services/rest/empinfo/?
XSD=getpersoninfo.xsd" xmlns="http://www.w3.org/2001/XMLSchema" />
</grammars><resources base="http://<hostname>:
<port>/web services/rest/empinfo/">
  <resource path="/getAllReports/">
    <method id="getAllReports" name="GET">
      <request>
        <param name="ctx_responsibility" type="xsd:string" style="query"
required="false" />
        <param name="ctx_respapplication" type="xsd:string" style="
query" required="false" />
        <param name="ctx_securitygroup" type="xsd:string" style="query"
required="false" />
        <param name="ctx_nlslanguage" type="xsd:string" style="query"
required="false" />
        <param name="ctx_orgid" type="xsd:int" style="query" required="
false" />
      </request>
      <response>
        <representation mediaType="application/xml" type="tns1:
getAllReports_Output" />
        <representation mediaType="application/json" type="tns1:
getAllReports_Output" />
      </response>
    </method>
  </resource>
  <resource path="/getDirectReports/">
    <method id="getDirectReports" name="GET">
      <request>
        <param name="ctx_responsibility" type="xsd:string" style="query"
required="false" />
        <param name="ctx_respapplication" type="xsd:string"
style="query" required="false" />
        <param name="ctx_securitygroup" type="xsd:string" style="query"
required="false" />
        <param name="ctx_nlslanguage" type="xsd:string" style="query"
required="false" />
        <param name="ctx_orgid" type="xsd:int" style="query" required="
false" />
      </request>
      <response>
        <representation mediaType="application/xml" type="tns2:
getDirectReports_Output" />
        <representation mediaType="application/json" type="tns2:
getDirectReports_Output" />
      </response>
    </method>
  </resource>
</resources>
</application>

```

```

</resource>
<resource path="/"getPersonInfo/ {personId}/">
  <param name="personId" style="template" required="true" type="xsd:int"
/>
  <method id="getPersonInfo" name="GET">
    <request>
      <param name="ctx_responsibility" type="xsd:string" style="query"
required="false" />
      <param name="ctx_respapplication" type="xsd:string"
style=" query" required="false" />
      <param name="ctx_securitygroup" type="xsd:string" style="query"
required="false" />
      <param name="ctx_nlslanguage" type="xsd:string" style="query"
required="false" />
      <param name="ctx_orgid" type="xsd:int" style="query" required="
false" />
    </request>
    <response>
      <representation mediaType="application/xml" type="tns3:
getPersonInfo_Output" />
      <representation mediaType="application/json" type="tns3:
getPersonInfo_Output" />
    </response>
  </method>
</resource>
<resource path="/getPersonInfo/">
  <method id="getPersonInfo" name="POST">
    <request>
      <representation mediaType="application/xml" type="tns3:
getPersonInfo_Input" />
      <representation mediaType="application/xml" type="tns3:
getPersonInfo_Output" />
    </request>
    <response>
      <representation mediaType="application/xml" type="tns3:
getPersonInfo_Input" />
      <representation mediaType="application/xml" type="tns3:
getPersonInfo_Output" />
    </response>
  </method>
</resource>
</resource path>
</application>

```

後で Java クライアントを定義するときに使用される次の情報をコピーまたは記録します。

- <resources base>="**http://<hostname>:**
 <port>/webservices/rest/empinfo/">
 この情報の http://<hostname>:<port>/webservices/rest/empinfo
 は、後で Java クライアント・プログラムで baseUrl として使用されます。
- <resource path>="/getPersonInfo/{**personId**}/">
 この情報の /getPersonInfo/{personId}/ は、後でサービス URL の後ろの部分
 を形成するために使用されます。
{personId} は、<param> タグを使用して定義されたパス変数です。HTTP GET メソッド
 が起動されると、実行時に実際の値(この例では従業員 ID 13137)に置き換えられます。
- <resource path>="/getDirectReports/">

この情報の `/getDirectReports/` は、後でサービス URL の後ろの部分形成するために使用されます。

Javascript を使用して HTML からのカスタム REST サービスを起動する

この例には、2 つの起動シナリオが含まれています。

1. `getPersonInfo` REST サービス操作を起動して、従業員の個人情報を取得する
2. ページ区切り制御パラメータを指定して `getDirectReports` REST サービス操作を起動して、ログイン・ユーザーの直属の部下を取得する

シナリオ 1: 従業員の個人情報の取得

次のステップを使用して、配置済の `getPersonInfo` REST サービス操作を起動します。

1. テキスト・エディタを使用して、次のコンテンツを含む HTML ファイルを作成します。
コードで、`<hostname>:<port>`、`<EBS username>` および `<password>` を実際の値に置き換えます。

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>Oracle E-Business Suite Integrated SOA Gateway - REST
Services Sample</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8" >
    <style>
      table,th,td
      {
        border:1px solid black;
        border-collapse:collapse;
      }
      th,td
      {
        padding:5px;
      }
    </style>
    <script language="javascript">
      function getEmpDetails()
      {
        var xmlhttp;
        if (window.XMLHttpRequest)
        { // code for IE7+, Firefox, Chrome, Opera, Safari
          xmlhttp=new XMLHttpRequest();
        }
        else
        { // code for IE6, IE5
          xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
        }
        xmlhttp.onreadystatechange=function()
        {
          if (xmlhttp.readyState==4 && xmlhttp.status==200)
          {
            var det = eval( "(" + xmlhttp.responseText + ")" );
            document.getElementById('div1').innerHTML=det.OutputParameters
. PersonBean[0].fullName;
            document.getElementById('div2').innerHTML=det.OutputParameters
. PersonBean[0].supervisorFullName;
          }
        }

        var empno = document.getElementById("empno");
        var url = "http://<hostname>:<port>/web services/rest/empinfo/
getPersonInfo/" + empno.value;

        xmlhttp.open('GET',url,true,"<EBS username>", "<password>");
        xmlhttp.send();
      }
    </script>
  </head>
  <body>
    <center><h1>Oracle E-Business Suite Integrated SOA Gateway - REST
Services Sample</h1>
      <h2>Path Parameter Example</h2>
    </center>
    <h3>This example demonstrates the use of Path Parameters in REST
Service URL.
    ISG REST Service expects personId as input. personId was annotated
as key_param. Hence, it can be sent as part of GET request URL.
    This example does not set Accept header in HTTP Request. REST
Service returns response in JSON format (JSON is default message
format).<br><br>
    Enter employee id and click Get Details button. ISG REST Service
will be invoked. Employee Name and his / her Manager's Name will be

```

```

displayed.<br>
</h3>
<center><table>
<tr>
<td>Enter Employee ID</td>
<td><input type="text" id="empno" size="10"/> <input
type="button" value="Get Details" onclick="getEmpDetails()" />
</td>
</tr>
<tr>
<td>Employee Name</td>
<td><div id="div1"/></div></td>
</tr>
<tr>
<td>Manager Name</td>
<td><div id="div1"/></div></td>
</tr>
</table>
</body>
</html>

```

この起動シナリオでは、パス・パラメータ{personId}を使用して、実行時に従業員 ID を取得します。その後、HTTP GET 要求によって getPersonInfo サービス操作が起動されます。

```

<resource path="/"getPersonInfo/{personId}"/>
  <param name="personId" style="template" required="true"
type="xsd: int"/>
  <method id="getPersonInfo" name="GET">

```

パス・パラメータ{personId}は前に@rep:key_param で注釈付けされており、GET 要求 URL として送信できます。

この例では、GET 要求に Accept ヘッダーは設定されません。REST サービスは、デフォルトの JSON 形式で応答メッセージを返します。

2. ブラウザ・ウィンドウで HTML ページを表示します。

HTML ページからのサービスの起動

Oracle E-Business Suite Integrated SOA Gateway - REST Services Sample

Path Parameter Example

This example demonstrates the use of Path Parameters in REST Service URL. ISG REST Service expects personId as input. personId was annotated as key_param. Hence, it can be sent as part of GET request URL. This example does not set Accept header in HTTP Request. REST Service returns response in JSON format (JSON is default message format).

Enter employee id and click Get Details button. ISG REST Service will be invoked. Employee Name and his / her Manager's Name will be displayed.

Enter Employee ID	13137	Get Details
Employee Name		
Manager Name		

3. HTML ページで従業員 ID (パス・パラメータ{personId})を入力し(例: 13137)、「Get Details」をクリックします。これにより、getPersonInfo REST サービス操作が起動し、関連付けられている従業員の名前とそのマネージャの名前が返されます。

この例では、従業員 ID が 13137 の「Taylor, Mr. Steve」が「Employee Name」として表示され、Taylor のマネージャである「Bennett, Terrence G (Terry)」が「Manager Name」として表示されます。

従業員の個人情報の表示

Oracle E-Business Suite Integrated SOA Gateway - REST Services Sample

Path Parameter Example

This example demonstrates the use of Path Parameters in REST Service URL. ISG REST Service expects personId as input. personId was annotated as key_param. Hence, it can be sent as part of GET request URL. This example does not set Accept header in HTTP Request. REST Service returns response in JSON format (JSON is default message format).

Enter employee id and click Get Details button. ISG REST Service will be invoked. Employee Name and his / her Manager's Name will be displayed.

Enter Employee ID	13137	Get Details
Employee Name	Taylor, Mr. Steve	
Manager Name	Bennett, Terrence G (Terry)	

シナリオ 2: ログイン・ユーザーの直属の部下の取得

次のステップを使用して、配置済の getDirectReports REST サービス操作を起動します。

1. テキスト・エディタを使用して、次のコンテンツを含む HTML ファイルを作成します。

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>Oracle E-Business Suite Integrated SOA Gateway - REST
Services Sample</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" >

    <style>
table,th,td
{
  border:1px solid black;
  border-collapse:collapse;
}
th,td
{
  padding:5px;
}
</style>
<script language="javascript">
  function getPrev()
  {
    offset = parseInt(offset) - 3;
    getDirectReports(offset);
  }

  function getNext()
  {
    offset = parseInt(offset) + 3;
    getDirectReports(offset);
  }

  function getDirectReports(offsetP)
  {
    var xmlhttp;
    if (window.XMLHttpRequest)
    { // code for IE7+, Firefox, Chrome, Opera, Safari
      xmlhttp=new XMLHttpRequest();
    }
    else
    { // code for IE6, IE5
      xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
    xmlhttp.onreadystatechange=function()
    {
      if (xmlhttp.readyState==4 && xmlhttp.status==200)
      {

        xmlDoc=xmlhttp.responseXML;
        txt="";
        retRows=0;
        x=xmlDoc.getElementsByTagName("PersonBean");

        if (x.length > 0)
        {
          txt = "<br><table border = \"1\"><tr><th>Employee
Id</th><th>Employee Name</th></tr>";
        }
        else
        {
          txt = "No records to display";
        }

        for (i=0;i<3;i++)
        {
          txt = txt + "<tr><td>" + xmlDoc.getElementsByTagName

```

```

("personId")[i].firstChild.nodeValue + " + xmlDoc.
getElementsByTagName("personId")[i].firstChild.nodeValue + "</td>";
    txt = txt + "<td>" + xmlDoc.getElementsByTagName("fullName")
[i].firstChild.nodeValue + " + xmlDoc.getElementsByTagName
("fullName")[i].firstChild.nodeValue + "</td></tr>";
    }

    if (x.length > 0)
    {
        txt = txt + "</table><br>";
    }

    y=xmlDoc.getElementsByTagName("ControlBean");
    rOffset = xmlDoc.getElementsByTagName("offset")[0].firstChild.
nodeValue;
    rFrom = 1 + parseInt(rOffset);
    rTo = 3 + parseInt(rOffset);
    txt = txt + "Displaying " + rFrom;
    txt = txt + " to " + rTo + " records.<br>";
    document.getElementById("div1").innerHTML=txt;

    if(offsetP > 2)
    { document.getElementById("div2").innerHTML="<input type=\"
button\" value=\"Previous\" onclick=\"getPrev()\"/>";
    }
    else
    {
        document.getElementById("div2").innerHTML="";
    }

    if(x.length == 4) {
        document.getElementById("div3").innerHTML="<input type=\"
button\" value=\"Next\" onclick=\"getNext()\"/>";
    }
    else
    {
        document.getElementById("div3").innerHTML="";
    }

    }
}

    var url = "http://<hostname>:
<port>/webservices/rest/empinfo/getDirectReports?ctx_orgid=204&
limit=4&offset" + offsetP;

    xmlhttp.open('GET',url,true);
    xmlhttp.setRequestHeader("Accept", "application/xml");
    xmlhttp.send();
}
</script>
</head>
<body>
    <center><h1>Oracle E-Business Suite Integrated SOA Gateway -
REST Services Sample</h1>
        <h2>Pagination Control Parameters Example</h2>
    </center>
    <h3>This example demonstrates the use of Pagination Control
Parameters. When you click Get Directs button, ISG REST Service
will be invoked. Provide EBS username
    and password. On successful invocation, the first 3 direct
reports (employees) of the logged in user will be displayed. Next
and Previous buttons will be displayed
    based on the number of records.

```



```

</h3>
<center>
  <input type="button" value="Get Directs" onclick="getDirectReports
(0)"/>
<div id="div1"/></div>
<div id="div2"/></div>
<div id="div3"/></div>
</center>

<script type="text/javascript"><!--
  offset=0;
  //--></script>

</body>
</html>

```

この起動シナリオでは、HTTP GET 要求によって getDirectReports サービス操作が起動されたときに、Javascript で定義されたページ区切り制御パラメータ **offset** を使用して、実行時に返されるレコードの数を制限し、ページを区切って表示します。

```

<script language="javascript">
...

    var url = "http://<hostname>:
<port>/webservices/rest/empinfo/getDirectReports/?ctx_orgid=204&
limit=4&offset" + offsetP;

    xmlhttp.open('GET',url,true);
    xmlhttp.setRequestHeader("Accept", "application/xml");
    xmlhttp.send();
  }
</script>

```

この例では、GET 要求に Accept ヘッダーが設定されます。REST サービスは、XML 形式で応答メッセージを返します。

2. ブラウザ・ウィンドウで HTML ページを表示します。

HTML ページからのサービスの起動

Oracle E-Business Suite Integrated SOA Gateway - REST Services Sample

Pagination Control Parameters Example

This example demonstrates the use of Pagination Control Parameters. When you click Get Directs button, ISG REST Service will be invoked. Provide EBS username and password. On successful invocation, the first 3 direct reports (employees) of the logged in user will be displayed. Next and Previous buttons will be displayed based on the number of records.

Get Directs

3. HTML ページで「Get Directs」をクリックします。これにより、getPersonInfo REST サービス操作が起動します。ユーザー名とパスワードの入力を要求するポップアップ・ウィンドウが表示されます。

ユーザー名とパスワードを入力し、「OK」をクリックすると、入力したユーザーの最初の 3 人の直属の部下(従業員)が表内に返されます。

「Next」をクリックすると、直属の部下のリストから取得された次の 3 人のレコードが表示されます。「Previous」をクリックすると、現在の直属の部下のリストから直前の 3 人のレコードが表示されます。

結果としての直属の部下の表示

Oracle E-Business Suite Integrated SOA Gateway - REST Services Sample

Pagination Control Parameters Example

This example demonstrates the use of **Pagination Control Parameters**. When you click **Get Directs** button, ISG REST Service will be invoked. Provide EBS username and password. On successful invocation, the first 3 direct reports (employees) of the logged in user will be displayed. Next and Previous buttons will be displayed based on the number of records.

Get Directs

Employee Id	Employee Name
276	Watson, Mr. Phillip
283	Evans, George
281	Rhodes, Mr. Connor

Displaying 4 to 6 records.

Previous
Next

トークン・ベース認証および XML ペイロードを使用したアプリケーション・モジュール・サービスの起動

REST サービスの起動シナリオ

REST サービスの起動を説明するための例として、Approvals Management (oracle.apps.fnd.wf.worklist.service.rt.server.WFWorklistServiceAMImpl) というアプリケーション・モジュール・サービスを使用します。この例では、このインターフェースに含まれている Get Open Notifications (getOpenNotifications) REST サービス操作を使用して、現在のユーザーのすべてのオープン通知のリストを返します。

結果の HTTP 要求を受け取ると、getOpenNotifications サービス操作が起動して、REST 要求として XML 形式で送信された入力ペイロードに基づいて、現在のユーザーのすべてのオープン・ワークフロー通知がフェッチされます。要求でページ・サイズとページ番号を提供することにより、結果をバッチで取得できます。

この要求にパスワードが指定されない場合、ユーザー資格証明を認証するためにトークン・ベースのセキュリティ・メソッドが使用されます。Oracle E-Business Suite ユーザー・セッションを作成するためにセキュリティ・ログイン・サービスが起動されて、ユーザー認証のパスワードのかわりに、Cookie としてのセッション ID が戻されます。

検証後、getOpenNotifications REST サービス操作が起動して、出力応答メッセージの一部としてすべてのオープン通知を XML 形式で返すことができます。

アプリケーション・モジュール・サービスを使用するための前提条件

設計時タスクを実行する前に、次のタスクが実行されるようにします。

この例に必要なライブラリの入手

XML 形式のペイロードを使用して REST サービスを正常に起動するには、この例のために <\$COMMON_TOP>/java/lib ディレクトリに用意されている次のライブラリを入手する必要があります。

- jersey-bundle_1.0.0.0_1-1-5-1.jar

このライブラリ・ファイルは、プロジェクトの作成中、後で設計時にプロジェクトに追加します。

HTTP 要求の *RESTHeader* の変数の設定

REST サービスの起動前に、REST サービスで、アプリケーション・コンテキストの値を *RESTHeader* 要素で渡すことができます。

これらの *RESTHeader* の要素は、*Responsibility*、*RespApplication*、*SecurityGroup*、*NLSLanguage* および *Org_Id* です。

RESTHeader の要素の詳細は、「アプリケーション・コンテキストの REST ヘッダー」(ページ 2-46)を参照してください。

Java を使用した REST サービスの起動

この項には、REST サービスの起動シナリオに基づく次のタスクが含まれています。

1. REST サービスの配置、ページ 4-49
2. 配置済 WADL URL の記録、ページ 4-50
3. セキュリティの作成、ページ 4-51
4. Java クラスを含むプロジェクトの作成、ページ 4-52
5. Java クラスを使用した REST サービスの起動、ページ 4-60

REST サービスの配置

次のステップを使用して、Approvals Management というアプリケーション・モジュール・サービスを配置します。

1. 「統合管理者」ロールを持つユーザーとして、Oracle E-Business Suite にログインします。ナビゲーション・メニューから、「統合 SOA ゲートウェイ」の職責を選択して、「統合リポジトリ」リンクを選択します。
2. 「統合リポジトリ」タブで、「検索」をクリックしてメインの「検索」ページにアクセスします。
3. 「詳細検索オプションの表示」をクリックして、詳細検索オプションを表示します。検索基準として、次のキー検索値を入力します。
 - カテゴリ: インタフェース・サブタイプ

- カテゴリ値: アプリケーション・モジュール・サービス
4. 「進む」をクリックして、検索を実行します。

Approvals Management のインタフェース名リンクをクリックして、インタフェースの詳細ページを開きます。

5. 「REST Web サービス」タブで、次の情報を入力します。

「REST Web サービス」タブ

Overview **REST Web Service** Grants

* Service Alias

REST Service Status Not Deployed

Service Operations

Display Name	Internal Name	GET	POST	Grant
Approvals Management	oracle.apps.fnd.wf.worklist.service.rt.server.WFWorklistServiceAMImpl	☒	☒	
Get Approval Types	getApprovalTypes	☐	☒	
Get Open Notifications	getOpenNotifications	☐	☒	
Get Pending Approvals By Approval Type	getPendingApprovalsByType	☐	☒	

- サービス別名: notification
- 選択したメソッドまたはサービス操作(この例では getOpenNotifications)の WADL およびスキーマのサービス・エンドポイントとして別名が表示されます。
- 「サービス操作」リージョンで、「Get Open Notifications」(getOpenNotifications)を選択します。
- 選択されたメソッドは、REST サービス操作として公開されます。
6. 「配置」をクリックして、Oracle E-Business Suite WebLogic 環境にサービスを配置します。

REST サービスが正常に配置されたら、WADL の記述を表示できる「**WADL の表示**」リンクとともに、「REST サービス・ステータス」フィールドに「配置済」が表示されます。

REST サービスの配置の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「REST Web サービスの配置」を参照してください。

配置済 WADL URL の記録

Approvals Management REST サービスの配置済 WADL からサービス・リソース情報を取得するには、統合開発者が「REST Web サービス」タブで「**WADL の表示**」リンクをクリックします。

「REST Web サービス」タブ: 「WADL の表示」リンク

Service Alias notification

REST Service Status Deployed [View WADL](#)

Service Operations

Display Name	Internal Name	GET	POST	Grant
Approvals Management	oracle.apps.fnd.wf.worklist.service.rt.server.WFWorklistServiceAMImpl			
Get Approval Types	getApprovalTypes		✓	
Get Open Notifications	getOpenNotifications		✓	
Get Pending Approvals By Approval Type	getPendingApprovalsByType		✓	

REST Service Security

REST Web Service is secured by HTTP Basic Authentication at HTTP Transport level. Send either of the following in "Authorization" header as per HTTP Basic scheme:

- Username:Password
- Security Token.

Tip: Use [Login Service](#) to obtain Security Token for given user credentials.

[Undeploy](#)

後で Java クライアントを定義するとき使用される次の情報を WADL の記述からコピーまたは記録します。

- `<resources base>="http://<hostname>:
<port>/webservices/rest/notification/">`

この情報の

`http://<hostname>:<port>/webservices/rest/notification` は、後で Java クライアント・プログラムでベース URL として使用されます。

- `<resource path>="/getOpenNotifications/">`

この情報は、後でサービス URL の後ろの部分形成するために使用されます。

セキュリティ付与の作成

Approvals Management インタフェースを REST サービスとして配置したら、統合管理者はセキュリティ付与を作成して、サービスまたはメソッドのアクセス権限を特定のユーザー、ユーザー・グループまたはすべてのユーザーに与えることができます。

次のステップを使用して、セキュリティ付与を作成します。

1. 「統合管理者」ロールを持つユーザーとして、Oracle E-Business Suite にログインします。職責「統合 SOA ゲートウェイ」を使って、ナビゲーション・メニューから「統合リポジトリ」リンクを選択します。
2. 検索を実行して、管理者が先ほど配置した Approvals Management サービスを探します。

「アイテム」リストで「**カスタム・アプリケーション**」を選択します。

「**OK**」をクリックします。「カスタム・アプリケーションの作成 - アプリケーションの名前付け」ページが表示されます。

2. 「アプリケーション名」フィールドにアプリケーションの適切な名前を入力します。「**次へ**」をクリックします。
3. 「カスタム・アプリケーションの作成 - プロジェクトの名前付け」ページが表示されます。「プロジェクト名」フィールドにプロジェクトの適切な名前を入力します(例: ISGAMClient)。
「プロジェクトの機能」タブで、「使用可能」リストから「**Java**」を選択します。右矢印ボタンを使用して、「使用可能」ウインドウから「選択済」ウインドウに、選択した機能を移動します。
「**次へ**」をクリックします
4. 「Java 設定の構成」ダイアログ・ボックスで「**終了**」をクリックします。
5. アプリケーション・ナビゲータで、作成したばかりのプロジェクトを右クリックし、ドロップダウン・メニューから「**新規**」を選択します。
6. 「新規ギャラリー」ウインドウで、「一般」カテゴリを展開して「**Java**」を選択します。「アイテム」リストで「**Java クラス**」を選択します。「**OK**」をクリックします。
7. 「Java クラスの作成」ダイアログで、デフォルトのクラス名を `AMClient` に変更します。その他の項目はデフォルトのままで、「**OK**」をクリックします。
8. ソース・エディタで新しいクラスが自動的に開き、スケルトン・クラス定義が表示されます。
次の **Java** コードでスケルトン・クラス定義を置き換えます。

```

package sample;

import java.net.HttpURLConnection;
import java.net.URL;
import com.sun.jersey.core.util.Base64;
import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;
import org.w3c.dom.Element;

public class AMClient {
    public AMClient() {
        super();
    }

    private static final String xmlRequest1 = "<?xml version=\"1.0\"
encoding=\"utf-8\"?>\n" +
"<getOpenNotifications_Input>\n" +
"  <RESTHeader>\n" +
"    <NLSLanguage>AMERICAN</NLSLanguage>\n" +
"    <Org_Id>202</Org_Id>\n" +
"    <RespApplication>SYSADMIN</RespApplication>\n" +
"    <Responsibility>SYSTEM ADMINISTRATOR</Responsibility>\n" +
"    <SecurityGroup>STANDARD</SecurityGroup>\n" +
"  </RESTHeader>\n" +
"<InputParameters>\n" +
"  <pageSize>3</pageSize>\n" +
"  <pageNumber>1</pageNumber>\n" +
"</InputParameters>\n" +
"</getOpenNotifications_Input>";

    /**
     * This method invokes the EBS Login service. It
     authenticates login credentials and returns accessTokenName and
     * accessToken values after successful validation of login
     credentials.
     */
    private static String[] getAccessToken(String baseUrl, String
username,String passwd) throws Exception {

        String loginUrl = baseUrl + "/login";
        URL url = new URL(loginUrl);
        //Obtaining connection to invoke login service.
        HttpURLConnection conn = (HttpURLConnection) url.
openConnection();
        String auth = username + ":" + passwd;
        byte[] bytes = Base64.encode(auth);
        String authStr = new String(bytes);
        //Setting Http request method
        conn.setRequestMethod("GET");
        //Setting the Http header values
        conn.setRequestProperty("Authorization", "Basic " +
authStr);
        conn.setRequestProperty("Accept", "application/xml");
        conn.setUseCaches(false);
        conn.setDoInput(true);
        conn.setDoOutput(true);
        conn.connect();
        String response = null;

```



```

//Get Response
try {
    response = readHttpResponse(conn);
} finally {
    if (conn != null)
        conn.disconnect();
}

//Parsing Response to obtain
DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
ByteArrayInputStream input = new ByteArrayInputStream
(response.getBytes("UTF-8"));
Document doc = builder.parse(input);
Element eAccessToken = (Element) doc.
getElementsByTagName("accessToken").item(0);
String strAccessToken = eAccessToken.getTextContent();
Element eTokenName = (Element) doc.getElementsByTagName
("accessTokenName").item(0);
String strTokenName = eTokenName.getTextContent();

return (new String[] { strTokenName, strAccessToken });
}

/**
 * This method reads response sent by the server and returns
it in a string representation.
 */
private static String readHttpResponse(HttpURLConnection conn) {
    InputStream is = null;
    BufferedReader rd = null;
    StringBuffer response = new StringBuffer();
    try {
        if (conn.getResponseCode() >= 400) {
            is = conn.getErrorStream();
        } else {
            is = conn.getInputStream();
        }
        rd = new BufferedReader(new InputStreamReader
(is));

        String line;
        while ((line = rd.readLine()) != null) {
            response.append(line);
            response.append('\n');
        }
    } catch (IOException ioe) {
        response.append(ioe.getMessage());
    } finally {
        if (is != null) {
            try {
                is.close();
            } catch (Exception e) {
            }
        }
        if (rd != null) {
            try {
                rd.close();
            } catch (Exception e) {
            }
        }
    }
    return (response.toString());
}

/**

```

```

* This method invokes Workflow Notification rest service using the
accessTokenName and accessToken values returned by EBS login service
*/
    public static void invokeRESTService(String svcUrlStr, String
tokenName,String tokenValue) throws IOException {

        URL url = new URL(svcUrlStr);
        //Obtaining connection to invoke the service
        HttpURLConnection conn = (HttpURLConnection) url.
openConnection();
        //Setting Http header values
        conn.setRequestMethod("POST");
        conn.setRequestProperty("Content-Type", "application/xml");
        //Adding the accessTokenName and accessToken as Cookies
        conn.addRequestProperty("Cookie", tokenName + "=" +
tokenValue);
        conn.setRequestProperty("Accept", "application/xml");
        conn.setUseCaches(false);
        conn.setDoInput(true);
        conn.setDoOutput(true);
        //Send request
        OutputStreamWriter wr = new OutputStreamWriter(conn.
getOutputStream());
        wr.write(xmlRequest1.toCharArray());
        wr.flush();
        wr.close();
        conn.connect();
        System.out.println("Response code - " + conn.
getResponseCode());
        //Get Response
        String response = null;
        try {
            response = readHttpResponse(conn);
        } finally {
            if (conn != null)
                conn.disconnect();
        }
        //Show Response
        System.out.println("Response is : \n" + response);
    }

    public static void main(String[] args) throws Exception {
        AMClient aMClient = new AMClient();
        String baseUrl = "http://<EBS hostname>:
<port>/webservices/rest";
        String svcUrlStr1 = baseUrl + "/"
notification/getOpenNotifications/";
        System.out.println("Calling EBS Login Service");
        //Get Access Token by invoking AOL Login Service
        String[] token = getAccessToken(baseUrl, "<EBS username>",
" <password>");
        System.out.println("Obtained EBS Access Token");
        //Invoke REST service using the Access Token
        invokeRESTService(svcUrlStr1, token[0], token[1]);
    }
}

```

デプロイ済 WADL から以前に記録されたリソース情報が、baseUrl と svcUrlStr1 の要素に置かれたこと注意してください。

注意: TLS が有効化された環境で Oracle E- Business Suite インスタンスが実行されている場合、baseUrl で **https** (http ではなく)を使用します。また、クラ

イアントの JVM のキースタに TLS 証明書をインポートする必要があります。

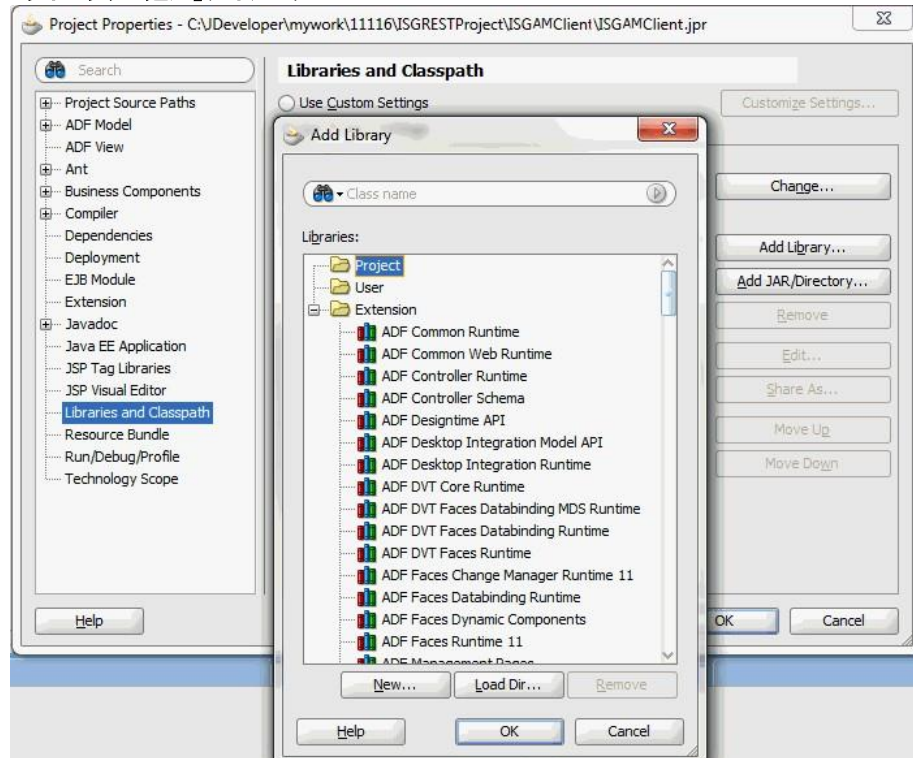
9. コードで、`<EBS hostname>:<port>`、`<EBS username>`および`<password>`を実際の値に置き換えます。

10. XML ペイロードを処理するために必要なライブラリを追加します。

次のステップを使用して、必要なライブラリ・ファイルをプロジェクト・プロパティに追加します。

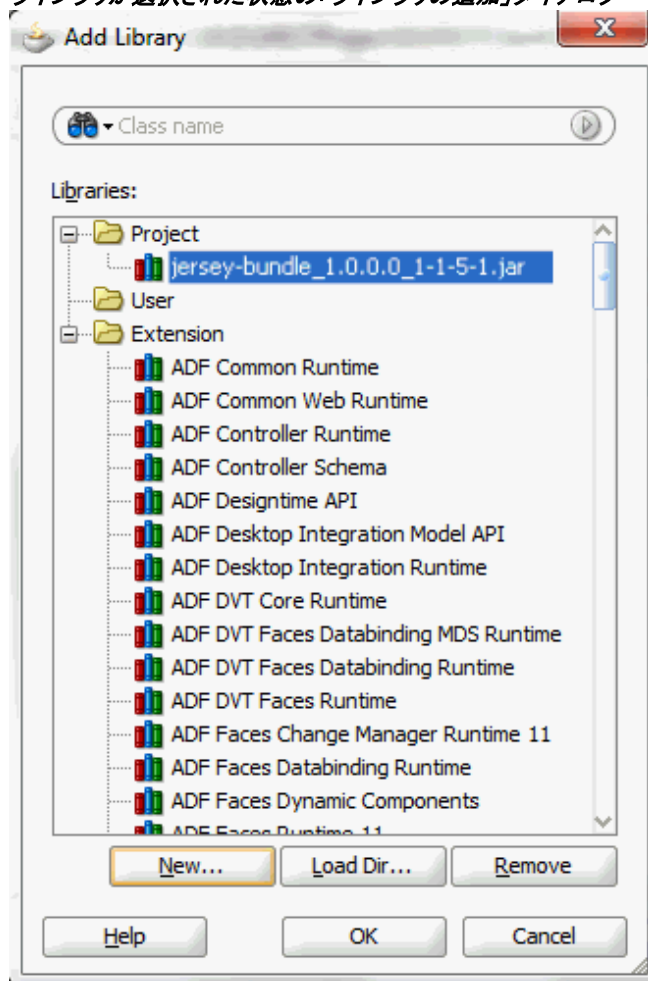
1. 作成したばかりのプロジェクトの名前を選択して右クリックし、選択メニューを開きます。
2. メニューから「プロジェクト・プロパティ」を選択します。「デフォルト・プロパティ」ダイアログ・ボックスが開きます。
3. 「ライブラリとクラスパス」を選択して、「ライブラリの追加」をクリックします。「ライブラリの追加」ダイアログ・ボックスが開きます。

「ライブラリの追加」ダイアログ



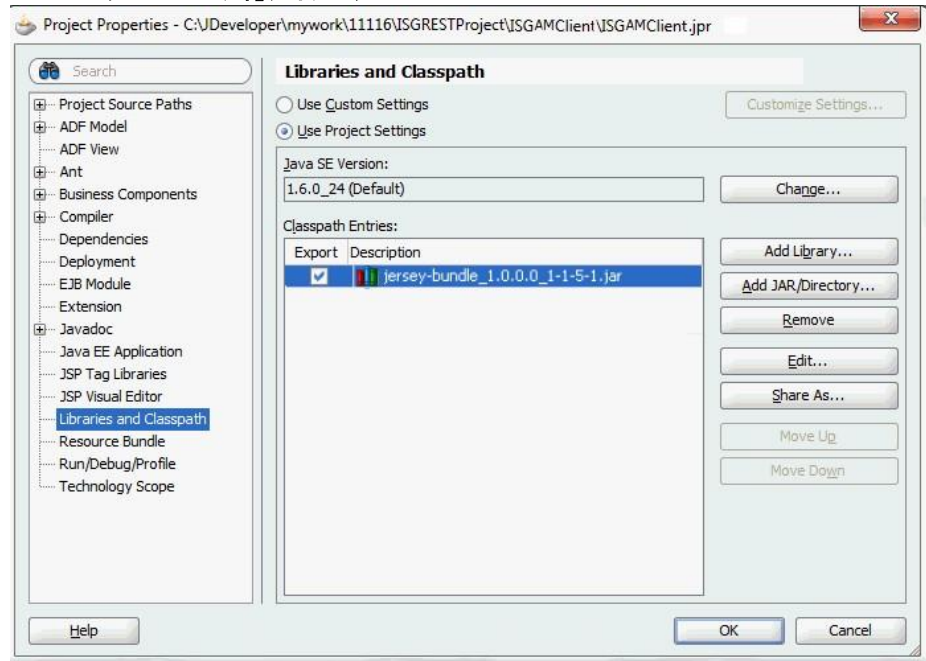
4. 「ライブラリの追加」ダイアログ・ボックスで、「プロジェクト」フォルダを選択して、「新規」をクリックします。「ライブラリの作成」ダイアログ・ボックスが開きます。
5. 「ライブラリ名」フィールドに、jersey-bundle_1.0.0.0_1-1-5-1.jar と入力します。「エントリの追加」をクリックします。「パス・エントリの選択」ダイアログ・ボックスが表示されます。
6. 「パス・エントリの選択」ダイアログ・ボックスで、ダウンロードした jersey-bundle_1.0.0.0_1-1-5-1.jar ファイルを探して選択します。こうすることでクラスパスに追加されます。
「OK」をクリックします。jersey-bundle_1.0.0.0_1-1-5-1.jar が「プロジェクト」フォルダに追加されます。

ライブラリが選択された状態の「ライブラリの追加」ダイアログ



7. 「プロジェクト・プロパティ」ダイアログ・ボックスが表示されます。「OK」をクリックします。このプロジェクトが、必要なライブラリで設定されました。

「プロジェクト・プロパティ」ダイアログ



11. 「ファイル」→「すべて保存」を選択して、作業内容を保存します。

Java クラスを使用した REST サービスの起動

Java クラス `AMClient.java` を含むプロジェクトを作成したら、Approvals Management REST サービスを起動するプロセスをコンパイルして実行する必要があります。

次のステップを使用して、Java クラスをコンパイルして実行します。

1. アプリケーション・ナビゲータで、設計時に作成した `AMClient.java` Java クラスを右クリックします。メニューから「メイク」を選択します。
2. `AMClient.java` Java クラスを右クリックし、メニューから「実行」を選択します。

「ログ」ウィンドウで、このプロセスをモニターして、コンパイルが成功したことを確認します。

「ログ」ウィンドウで、実行が成功したことを確認します。

出力メッセージの表示

Approvals Management REST サービスが正常に起動されると、次の応答が表示されます。

```

Calling EBS Login Service
Obtained EBS Access Token
Response code - 200
Response is :
<?xml version = '1.0' encoding = 'UTF-8'?>
<getOpenNotifications_Output xmlns:ns="http://xmlns.oracle.
com/apps/fnd/rest/output">
  <ns:OutputParameters>
    <ns:Output>
      <ns:WFApprovalsBean
xmlns:ns2="http://xmlns.oracle.
com/apps/fnd/wf/worklist/service/rt/artifacts/fault/"
xmlns:ns1="http://xmlns.oracle.
com/apps/fnd/wf/worklist/service/rt/artifacts/notificationdetails/"
xmlns:ns="http://xmlns.oracle.
com/apps/fnd/rest/notification/getOpenNotifications"
xmlns:ns0="http://xmlns.oracle.
com/apps/fnd/wf/worklist/service/rt/artifacts/directoryservices/">
        <ns:output>
          <notifications count="23">
            <notification>
              <NotificationId>7857291</NotificationId>
              <Subject>Standard Purchase Order 6562 has been
approved</Subject>
              <Language>US</Language>
              <BeginDate>30-Sep-2010 01:25:03</BeginDate>
              <Status>Open</Status>
              <FromUser>Brown, Casey</FromUser>
              <Name>POAPPRV</Name>
              <SentTime>30-Sep-2010</SentTime>

<NotificationLink>OA.jsp?OAFunc=FND_WFNTF_DETAILS&NtfId=7857291&amp;..
.. </NotificationLink>

              <ns1:result/>
            </notification>
          <notification>
            <NotificationId>7857248</NotificationId>
            <Subject>Standard Purchase Order 6561 has been
approved</Subject>
            <Language>US</Language>
            <BeginDate>30-Sep-2010 01:20:27</BeginDate>
            <Status>Open</Status>
            <FromUser>Brown, Casey</FromUser>
            <Name>POAPPRV</Name>
            <SentTime>30-Sep-2010</SentTime>

<NotificationLink>OA.jsp?
OAFunc=FND_WFNTF_DETAILS&NtfId=7857248&amp;.....</NotificationLink>

            <ns1:result/>
          </notification>
        <notification>
          <NotificationId>7747986</NotificationId>
          <Subject>Standard Purchase Order 6549 has been
approved</Subject>
          <Language>US</Language>
          <BeginDate>13-Sep-2010 17:00:54</BeginDate>
          <Status>Open</Status>
          <FromUser>Stock, Pat</FromUser>
          <Name>POAPPRV</Name>
          <SentTime>13-Sep-2010</SentTime>

<NotificationLink>OA.jsp?
OAFunc=FND_WFNTF_DETAILS&NtfId=7747986&amp;.....</NotificationLink>

```

```
<ns1:result/>
  </notification>
  </notifications>
  </ns:output>
</ns:WFApprovalsBean>
</ns:Output>
<ns:ControlBean>
  <ns:filter/>
  <ns:limit/>
  <ns:offset/>
  <ns:sort/>
  </ns:ControlBean>
</ns:OutputParameters>
</getOpenNotifications_Output>
```

この例では、XML 形式の応答メッセージによって最初のページの 3 つのオープン・ワークフロー通知が提供されます。

XMLゲートウェイ・インバウンドおよびアウトバウンド・インタフェースの使用

概要

Oracle E-Business Suite Integrated SOA Gateway は、Oracle E-Business Suite と Web コンシューマの間の通信インフラストラクチャを提供します。インバウンドおよびアウトバウンド XML データは、コンシューマと Oracle E-Business Suite の間で Oracle XML Gateway を介して交換されます。

Oracle XML Gateway は、XML 統合のための標準ベースの共通アプローチを提供します。XML は、データを検索、交換および表示する方法を標準化し、それによってサプライ・チェーン全体の相互運用性を実現するため、統合ソリューションの鍵となります。

Oracle XML Gateway には、XML メッセージングをサポートするために Oracle E-Business Suite と簡単に統合できる一連のサービスが備えられています。Oracle XML Gateway は、Oracle Advanced Queuing のメッセージ伝播機能を使用して、アウトバウンド XML メッセージを配信し、ビジネス・パートナーからインバウンド XML メッセージまたは取引を受け入れる Oracle Transport Agent と統合されます。

Oracle E-Business Suite とコンシューマの間の双方向の統合を実現するために、Oracle E-Business Suite Integrated SOA Gateway は、次のアプローチで XML ゲートウェイ・マップ・インタフェース・タイプをサポートします。

- インバウンド XML ゲートウェイ・マップ・インタフェースの場合、インバウンド XML ゲートウェイ・インタフェースの Web サービスをデプロイすると、インバウンド・データを Oracle E-Business Suite に挿入する BPEL プロセスによる SOA コンポジット・アプリケーションを作成するとき、WSDL で表されたデプロイ済のサービスを使用できます。
- アウトバウンド XML ゲートウェイ・マップ・インタフェースの場合、アウトバウンド・メッセージは最初に ECX_OUTBOUND キューにエンキューされるため、Oracle E-Business Suite Integrated SOA Gateway は、サブスクリプション・モデルを介してそれをサポートします。これを行うために、メッセージをデキューして Oracle E-Business Suite からアウトバウンド・データを取得します。

その後、取得したデータを、メッセージを購読している取引先やコンシューマに渡すことができます。

インバウンド XML ゲートウェイ・インタフェースのデプロイ済 Web サービスを使用する方法について理解を深めるとともに、アウトバウンド XML ゲートウェイにおけるサブスクリプション・モデルの仕組みを理解するために、この章では、次のトピックについて説明します。

- XML ゲートウェイ・インバウンド Web サービスの使用、ページ 5-2
 - 設計時における XML ゲートウェイ・インバウンド・サービスの使用、ページ 5-2
 - 実行時における BPEL プロセスによる SOA コンポジット・アプリケーションのデプロイおよびテスト、ページ 5-22
- サブスクリプション・モデルを介した XML ゲートウェイ・アウトバウンドの使用、ページ 5-32
 - 設計時に BPEL プロセスによる SOA コンポジット・アプリケーションを作成する場合の XML ゲートウェイ・アウトバウンド・メッセージの使用、ページ 5-32
 - 実行時における BPEL プロセスによる SOA コンポジット・アプリケーションのデプロイおよびテスト、ページ 5-26

この後の各項で説明する例では、BPEL プロセスによる SOA コンポジット・アプリケーションを作成するためのデザインタイム・ツールとしては Oracle JDeveloper 11g (11.1.1.6.0)を使用し、プロセスのデプロイには Oracle SOA Suite 11g (11.1.1.6.0)を使用します。

注意: Oracle JDeveloper とその他の Oracle Fusion Middleware コンポーネント (Oracle SOA Suite など)を使用している場合、SOA テクノロジを有効にするには、SOA テクノロジ用の JDeveloper の拡張機能である Oracle SOA Suite コンポジット・エディタを手動でダウンロードする必要があります。追加の Oracle Fusion Middleware 設計時コンポーネントのインストールの詳細は、*Oracle Fusion Middleware Oracle JDeveloper インストレーション・ガイド*を参照してください。

XML ゲートウェイ・インバウンド・サービスの使用

この項では、次のトピックを取り上げます。

- 設計時における XML ゲートウェイ・インバウンド・サービスの使用、ページ 5-2
- 実行時における BPEL プロセスによる SOA コンポジットのデプロイおよびテスト、ページ 5-22

設計時における XML ゲートウェイ・インバウンド・サービスの使用

BPEL プロセスによる SOA コンポジット・アプリケーションのシナリオ

BPEL プロセスによる SOA コンポジット・アプリケーションの作成を説明するための例として、XML ゲートウェイ・インバウンド・プロセス「発注 XML 取引」について考えます。この例では、XML ゲートウェイ・インバウンド・メッセージ・マップが PROCESS_PO_007 インバウンド・マップを介して Web サービスとして公開されます。これにより、関連する発注が作成されたときに、ヘッダーおよび明細品目を含む受注データを Order Management システムに挿入できるようになります。

取引先から発注が送信されると、MESSAGE_TYPE、MESSAGE_STANDARD、TRANSACTION_TYPE、TRANSACTION_SUBTYPE、PARTY_SITE_ID、DOCUMENT_NUMBER などの ECX ヘッダー・プロパティとともに発注データが BPEL プロセスへの入力として使用されます。その後、BPEL プロセスによって、この発注が ECX_INBOUND キューにプッシュされます。ECX_INBOUND キューに対してエージェント・リスナーが実行されている場合は、実行エンジンによってさらに処理できます。Oracle XML Gateway がこの XML メッセージを捕捉し、取引先の検証を行って、受注データを Order Management アプリケーションに挿入します。

デプロイメント後、BPEL プロセスによる SOA コンポジット・アプリケーションが正常に実行されると、発注が作成されたときに、同じ受注情報が受注管理表に挿入されます。

XML ゲートウェイ・インバウンド・サービスを使用して BPEL プロセスによる SOA コンポジット・アプリケーションを構成するための前提条件

XML ゲートウェイ・インバウンド・サービスに関する設計時タスクを実行する前に、次のタスクを確実に実施してください。

- 統合管理者または統合開発者は、事前に Web サービスを生成しておく必要があります。その後、管理者が、Oracle SOA Suite WebLogic 管理対象サーバーに、生成されたサービスをデプロイします。
- 統合開発者は、Web サービスとして公開されたインバウンド・メッセージ・マップのデプロイ済 WSDL URL を検索して記録する必要があります。
- XML 取引に必要な XML ゲートウェイ・ヘッダー変数を設定する必要があります。
- XML 文書を受信するために、取引先を使用できるようにする必要があります。
- ECX インバウンド・エージェント・リスナーと ECX 取引エージェント・リスナーが稼働中である必要があります。

XML ゲートウェイ・インバウンド Web サービス・コンポジットのデプロイ

選択した XML ゲートウェイ・インバウンド・マップの Web サービスが正常に作成されたら、統合管理者は、Oracle Integration Repository から Oracle SOA Suite WebLogic 管理対象サーバーにサービスをデプロイする必要があります。

たとえば、統合開発者がデプロイ済 WSDL を使用して BPEL プロセスによる SOA コンポジット・アプリケーションを作成する前に、まず次のステップを実行する必要があります。

1. Web サービスを生成するには、統合管理者または統合開発者がまずインタフェース定義(XML ゲートウェイ・インバウンド・インタフェース INBOUND:Process Purchase Order

XML Transaction (ONT: POI)など)を見つけ、インタフェースの詳細ページで「生成」をクリックします。

サービスが正常に生成されると、「Web サービス」リージョンの「Web サービス・ステータス」フィールドが「未生成」から「生成済」に変わります。

Web サービスの生成方法の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「Web サービスの生成」を参照してください。

2. 生成した Web サービスを配置するには、統合管理者がいずれかの認証タイプを選択してから、「**配置**」をクリックします。Oracle SOA Suite に配置されたサービスは、有効なサービスであり、新しい SOAP 要求を受け入れる準備が整っています。

サービスが正常に配置されると、選択した認証タイプが表示され、「Web サービス・ステータス」フィールドが「配置済」および「有効」状態になります。認証タイプによる Web サービスの保護の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「Web サービス・セキュリティの管理」を参照してください。

Web サービスの配置方法の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「Web サービスの配置および配置解除」を参照してください。

WSDL URL の検索および記録

統合開発者は、インバウンド・メッセージ・マップの配置済 Web サービスの WSDL URL を検索して記録する必要があります。

この WSDL 情報は、設計時における BPEL プロセスの作成中に、Web サービスとして公開されたインバウンド・マップのパートナー・リンクを作成するときに後で使用されます。

インタフェースを検索してインタフェースの詳細をレビューする方法の詳細は、「統合インタフェースの検索および表示」(ページ 2-1)を参照してください。

XML ゲートウェイ・ヘッダー変数の設定

XML ゲートウェイ・ヘッダー情報を Oracle E-Business Suite に提供するには、BPEL PM の特定の変数を設定する必要があります。MESSAGE_TYPE、MESSAGE_STANDARD、TRANSACTION_TYPE、TRANSACTION_SUBTYPE、DOCUMENT_NUMBER および PARTY_SITE_ID は、XML 取引を正常に完了するために設定する必要がある必須のヘッダー変数です。

「Assign アクティビティの追加」(ページ 5-17)を参照してください。

XML 文書を受信するための取引先の設定

次のステップを使用して、取引先を設定します。

1. 「XML ゲートウェイ」職責を持つユーザーとして Oracle E-Business Suite にログインします。
2. ナビゲータから「**取引先の定義**」リンクをクリックします。

3. 「取引先設定」フォームが表示されます。「表示」→「入力して問合せ」→「入力」を選択し、次の情報を入力して既存の取引先を検索します。

- 取引先タイプ: 顧客
- 取引先名: Hilman and Associates
- 取引先サイト: 1230 East 60th Street South Tulsa OK 74146

「表示」→「入力して問合せ」→「入力」を選択して問合せを実行し、取引先 Hilman and Associates を既存の取引先とともに検索します。

目的の取引先の検索

The screenshot shows the 'Trading Partner Setup' window. The 'Operating Unit' is 'Vision Operations'. The 'Trading Partner Type' is 'Customer'. The 'Trading Partner Name' is 'Hilman and Associates'. The 'Trading Partner Site' is '1230 East 60th Street South Tulsa OK 74146'. The 'Company Admin Email' is 'nobody@localhost'. There are buttons for 'User Setup' and 'Code Conversion'. Below the form is a table titled 'Trading Partner Details'.

Transaction Type	Transaction SubType	Standard Code	External Transaction Type	External Transaction SubType	Direction	Map	Connection/Hub	Protocol Type
ONT	POA	OAG	PO	ACKNOWLED	OUT	ONT_3A4A_O	DIRECT	SMTP
ONT	CPO	OAG	PO	CANCEL	IN	ONT_3A9R_O		
ONT	POI	OAG	PO	PROCESS	IN	ONT_3A4R_O		
ONT	SSO	OAG	SALESORDE	SHOW	OUT	PROCESS_P	DIRECT	SMTP

次の取引先詳細を入力します。

- トランザクション・タイプ: ONT
- トランザクション・サブタイプ: POI
- 標準コード: OAG
- 外部トランザクション・タイプ: PO
- 外部トランザクション サブタイプ: PROCESS

- 方向: IN
- マップ: ONT_3A4R_OAG72_IN
- ソース取引先ロケーション・コード: HIL-Tulsa

4. エントリを保存します。

エージェント・リスナーが稼働中であることの確認

ECX_INBOUND および ECX_TRANSACTION キューに対するエージェント・リスナーが稼働中であることを確認します。次のステップを使用して、Oracle E-Business Suite でこれらのリスナーを構成します。

1. 「ワークフロー管理者」職責を持つユーザーとして Oracle E-Business Suite にログインします。
2. ナビゲータから「ワークフロー管理者 Web アプリケーション」リンクをクリックします。
3. 「Oracle Applications Manager」の下にある「ワークフロー・マネージャ」リンクをクリックします。
4. 「エージェント・リスナー」の横にあるステータス・アイコンをクリックします。
5. **ECX インバウンド・エージェント・リスナー**と **ECX 取引エージェント・リスナー**を構成し、スケジュールします。リスナーを選択し、「処理」ボックスから「開始」を選択します。稼働中でない場合は、「進む」をクリックします。

BPEL プロセスによる SOA コンポジット・アプリケーションの作成フロー

XML ゲートウェイ・インバウンド・プロセス「発注 XML 取引」のビジネス・シナリオに基づいて、この章では次の設計時タスクについて説明します。

1. 新しい BPEL プロセスによる SOA コンポジット・アプリケーションの作成、ページ 5-7
このステップを使用して、同期 BPEL プロセス・テンプレートから、XMLGatewayInbound.bpel という、新しい BPEL プロセスによる SOA コンポジット・アプリケーションを作成します。これにより、取引先からの入力を受け入れる受入と、要求アプリケーションに BPEL プロセスの出力を返信する返信の 2 つのダミー・アクティビティが自動的に作成されます。
2. パートナ・リンクの作成、ページ 5-11
このステップを使用して、インバウンド・メッセージを Oracle E-Business Suite に挿入するためのパートナ・リンクを作成します。
3. ファイル・アダプタのパートナ・リンクの追加、ページ 5-12
このステップを使用して、取引先から受け入れた XML ファイルを捕捉して XML メッセージを取得するためのファイル・アダプタのパートナ・リンクを追加します。
4. Invoke アクティビティの追加、ページ 5-16

このステップを使用して、次のことを目的とした 2 つの **Invoke** アクティビティを追加します。

1. 受け入れられた XML メッセージの詳細を受入アクティビティから取得します。
2. 発注情報を ECX_INBOUND キューにエンキューします。

5. Assign アクティビティの追加、ページ 5-17

このステップを使用して、次のことを目的とした 2 つの **Assign** アクティビティを作成します。

1. 取得した XML メッセージを 1 つ目の **Invoke** アクティビティから 2 つ目の **Invoke** アクティビティに渡します。
2. ECX ヘッダー変数を 2 つ目の **Invoke** アクティビティに入力変数として渡します。

BPEL プロセスのサービス・コンポーネントを使用した SOA コンポジット・アプリケーションの一般情報と作成方法の詳細は、*Oracle Fusion Middleware Oracle SOA Suite 開発者ガイド* を参照してください。

新しい BPEL プロセスによる SOA コンポジット・アプリケーションの作成

このステップを使用して、様々な BPEL プロセス・アクティビティによる新規 SOA コンポジット・アプリケーションを作成します。

新しい BPEL プロセスによる SOA コンポジット・アプリケーションを作成するには:

1. Oracle JDeveloper を開きます。
2. アプリケーション・ナビゲータで「**新規アプリケーション**」をクリックします。
「SOA アプリケーションの作成 - アプリケーションの名前付け」ページが表示されます。

「SOA アプリケーションの作成」の「アプリケーションの名前付け」ページ

Create SOA Application - Step 1 of 3

Name your application

Application Name
Project Name
Project SOA Settings

Application Name:
XMLGatewayInbound

Directory:
C:\Developer\mywork\XMLGatewayInbound Browse...

Application Package Prefix:

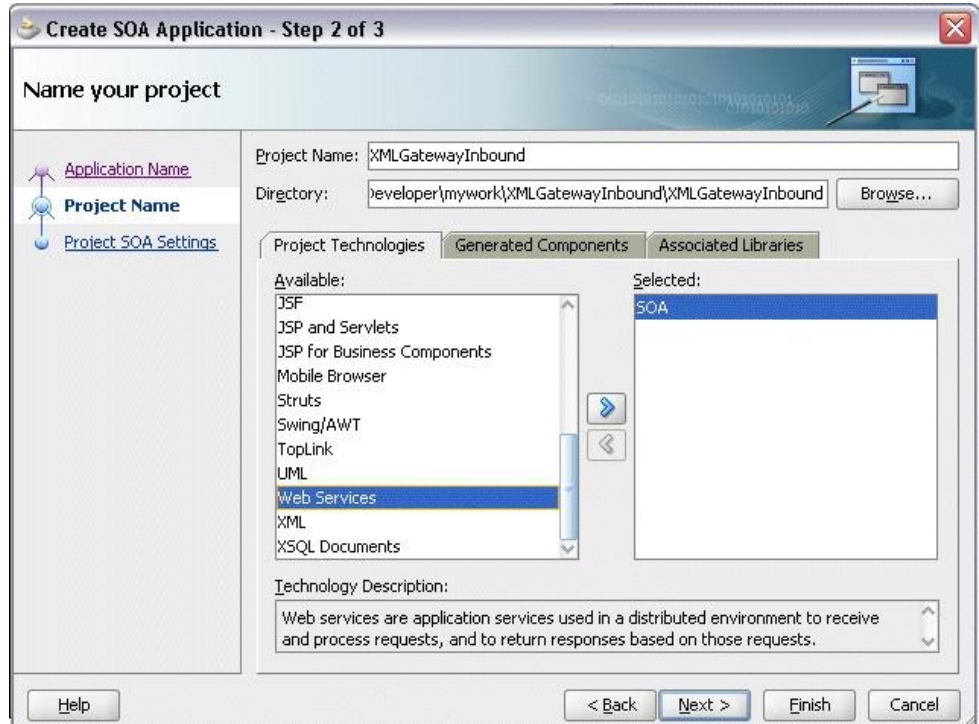
Application Template:

- ☐ Java Desktop Application (ADF)
Creates a databound rich client application. The application consists of one project for the client (ADF Swing), and another project for the ADF Model (ADF Business Components).
- ☐ Java EE Web Application
Creates a databound web application. The application consists of one project for the view and controller components (JSF), and another project for the data model (EJB session beans and JPA entities).
- ☒ **SOA Application**
Creates a SOA (service-oriented architecture) application. The application consists of one SOA project for the SOA composite, components, and adapters.

Help < Back Next > Finish Cancel

3. 「アプリケーション名」フィールドにアプリケーションの適切な名前を入力して、「アプリケーション・テンプレート」リストから「SOA アプリケーション」を選択します。
「次へ」をクリックします。「SOA アプリケーションの作成 - プロジェクトの名前付け」ページが表示されます。

「SOA アプリケーションの作成 - プロジェクトの名前付け」ページ



4. 「プロジェクト名」フィールドにプロジェクトの適切な名前を入力します (XMLGatewayInbound など)。
5. 「プロジェクト・テクノロジー」タブで、「Web サービス」を選択して、「使用可能」のテクノロジー・リストから「選択済」のテクノロジー・リストに「SOA」が選択されるようにします。
「次へ」をクリックします。「SOA アプリケーションの作成 - SOA 設定の構成」ページが表示されます。
6. 「コンポジット・テンプレート」リストから**コンポジット BPEL プロセス**を選択して、「終了」をクリックします。新規アプリケーションと SOA プロジェクトを作成しました。これにより、SOA コンポジットが自動的に作成されます。
「BPEL プロセスの作成」ページが表示されます。

「BPEL プロセスの作成」ページ

Create BPEL Process

BPEL Process

A BPEL process is a service orchestration, based on the BPEL specification, used to describe/execute a business process (or large grained service), which is implemented as a stateful service.

☒ BPEL 1.1 Specification ☐ BPEL 2.0 Specification

Name: XMLGINbound

Namespace: http://xmlns.oracle.com/XMLGatewayInbound1/XMLGatewayInbound/XMLGINbound

Template: Synchronous BPEL Process

Service Name: xmlginbound_client

☒ Expose as a SOAP service

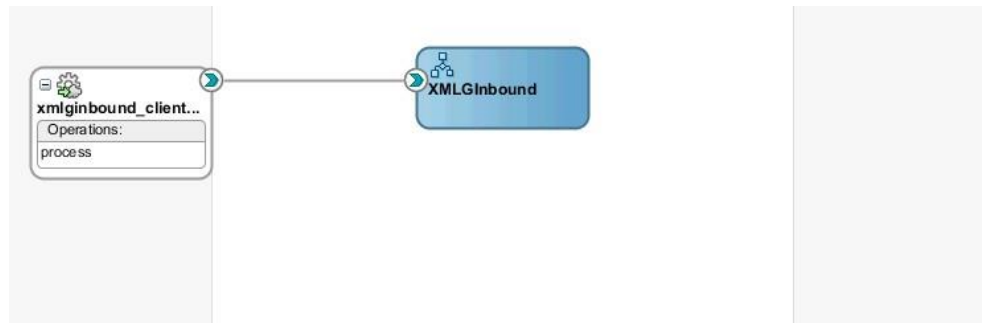
Transaction: required

Input: xmlns.oracle.com/XMLGatewayInbound1/XMLGatewayInbound/XMLGINbound}process

Output: de.com/XMLGatewayInbound1/XMLGatewayInbound/XMLGINbound}processResponse

Help OK Cancel

7. デフォルトの「**BPEL 1.1 Specification**」の選択を変更せずに維持します。これにより、BPEL 1.1 Specification をサポートする BPEL プロジェクトが作成されます。
「名前」フィールドに BPEL プロセスの適切な名前を入力します(例: XMLGINbound)。
「テンプレート」フィールドで「**同期 BPEL プロセス**」を選択します。
「トランザクション」ドロップダウン・リストから「**required**」を選択します。「**OK**」をクリックします。
受入アクティビティと返信アクティビティのある同期 BPEL プロセスが作成されます。また、指定した名前(例: XMLGINbound.bpel および XMLGINbound.wsdl)を使用した bpel および wsdl を含む必要なソース・ファイルと、composite.xml も生成されます。
8. 「SOA コンテンツ」→「ビジネス・ルール」に移動し、composite.xml をクリックしてコンポジット・ダイアグラムを表示します。



XMLGInbound コンポーネントをダブルクリックして BPEL プロセスを開きます。

パートナ・リンクの作成

このステップを使用して、受注データを Oracle E-Business Suite に挿入するための ONT POI というパートナ・リンクを作成します。

受注データを Oracle E-Business Suite に挿入するためのパートナ・リンクを作成するには:

1. Oracle JDeveloper で、「パートナ・リンク」領域の右側にマウスを置いて右クリックし、プルダウン・メニューから「パートナ・リンクの作成」を選択します。または、「BPEL コンストラクト」リストから「パートナ・リンク」を、プロセス・ダイアグラムの右側の「パートナ・リンク」スイムレーンにドラッグ・アンド・ドロップできます。

「パートナ・リンクの作成」ウィンドウが表示されます。

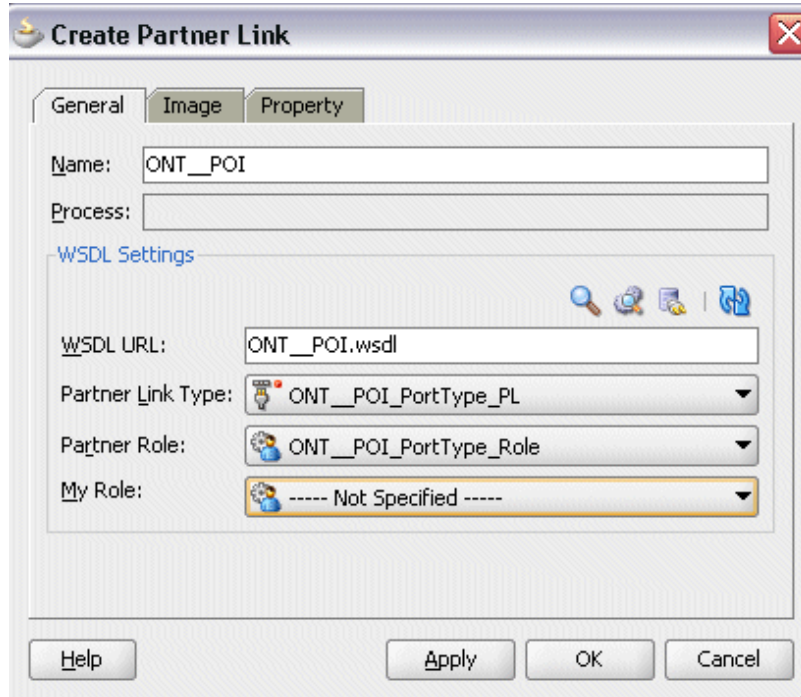
2. 前に統合リポジトリから記録した、XML ゲートウェイ・インバウンド・マップ INBOUND: Process Purchase Order XML Transaction (ONT:POI) に対応する WSDL URL をコピーし、「WSDL ファイル」フィールドに貼り付けます。

[Tab]キーを押します。

3. デフォルトでパートナ・リンク・タイプを作成する新しい WSDL ファイルをシステムで作成するかどうかを尋ねる「パートナ・リンク・タイプ」メッセージ・ダイアログ・ボックスが表示されます。

パートナ名の値が自動的に移入されるように、「はい」をクリックします。パートナ・リンク名を ONT POI として手動で入力できます。

パートナ・リンクの作成



ドロップダウン・リストから「パートナ・リンク・タイプ」フィールドと「パートナ・ロール」フィールドを選択します。「適用」→「OK」をクリックします。

パートナ・リンクは、必要な WSDL 設定で作成され、BPEL プロジェクトでは、プロセス・ダイアグラムの枠線の領域の新しいアイコンによって表現されます。

パートナ・リンクが BPEL プロセス・ダイアグラムの「パートナ・リンク」セクションに追加されます。

ONT_POIWrapper.wsdl をダブルクリックし、「ソース」タブをクリックして、作成したパートナ・リンクのソース WSDL 記述を表示します。参照 WSDL を配置済サービス・エンドポイントの WSDL に置き換えます。「ファイル」→「すべて保存」を選択して、変更を保存します。

ファイル・アダプタのパートナ・リンクの追加

このステップを使用して、XML メッセージを取得するためのファイル・アダプタのパートナ・リンクを追加することにより、BPEL プロセスを構成します。

XML メッセージを取得するためのファイル・アダプタのパートナ・リンクを追加するには:

1. Oracle JDeveloper で、「ファイル・アダプタ」サービスを「BPEL サービス」からプロセス・ダイアグラムの右側の「パートナ・リンク」スイムレーンにドラッグ・アンド・ドロップします。ファイル・アダプタ構成ウィザードへようこそページが表示されます。
2. 「次へ」をクリックします。「サービス名」ダイアログ・ボックスが表示されます。

3. ファイル・アダプタ・サービス名を入力します(例: GetXMLMsg)。サービスに関するオプションの説明を追加できます。
4. 「次へ」をクリックします。「アダプタ・インタフェース」ダイアログ・ボックスが表示されます。
5. 「操作およびスキーマから定義(後で指定)」ラジオ・ボタンを選択して、「次へ」をクリックします。「操作」ダイアログ・ボックスが表示されます。

操作の指定

Operation

The File Adapter supports four operations. There is a Read File operation that polls for incoming files in your local file system, a Write File operation that creates outgoing files, a Synchronous Read File operation that reads the current contents of a file, and a List Files operation that lists file names in specified locations. Specify the Operation type and Operation Name. Only one operation per Adapter Service may be defined using this wizard.

Operation Type: ☐ Read File ☐ Write File ☒ Synchronous Read File ☐ List Files

Operation Name:

Help < Back Next > Finish Cancel

6. 「同期 Read File」などの操作タイプを指定します。これにより、「操作名」フィールドが自動的に設定されます。
- 「次へ」をクリックして、「ファイル・ディレクトリ」ダイアログ・ボックスにアクセスします。

入力ファイル・ディレクトリの指定

Adapter Configuration Wizard - Step 5 of 8

File Directories

Enter directory information for the incoming file of the Synchronous Read File operation.

Directory names are specified as: ☒ Physical Path ☐ Logical Name

Directory for Incoming Files (physical path):

☐ Archive processed files
Archive Directory for Processed Files (physical path):

☒ Delete files after successful retrieval

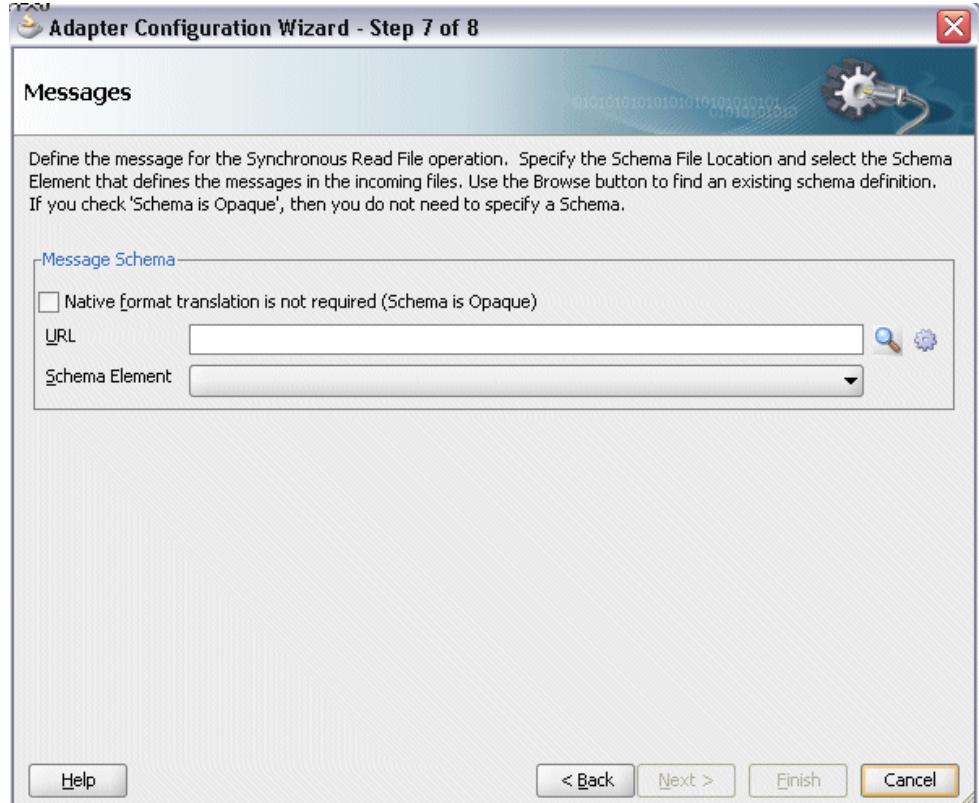
7. 「物理パス」ラジオ・ボタンを選択して、入力ペイロードのファイル・ディレクトリの情報を入力します。たとえば、/usr/tmp/と入力します。

注意: ここで指定した物理ディレクトリからファイルを検索できるようにするには、最初に、指定したディレクトリに入力ペイロード・ファイル(例: order_data_xmlg.xml)を置いておく必要があります。

「次へ」をクリックして、「ファイル名」ダイアログ・ボックスを開きます。

8. 同期 Read File 操作のファイル名を入力します。たとえば、order_data_xmlg.xml と入力します。「次へ」をクリックします。「メッセージ」ダイアログ・ボックスが表示されます。

メッセージ・スキーマの指定



9. 「URL」フィールドの右にある「スキーマ・ファイルを参照」を選択します。「タイプ・チューザ」ウィンドウが表示されます。

「タイプ・チューザ」ウィンドウの右上隅にある「スキーマ・ファイルのインポート」ボタンをクリックします。これにより、「スキーマ・ファイルのインポート」ポップアップ・ウィンドウが開きます。

サービスのスキーマの場所を入力します(例:

```
http://<soa_suite_hostname>:<port>/soa-  
infra/services/default/<jndi_name>_XMLGATEWAY_ONT_POI/ONT  
PO_I_Service?XSD=xsd/PROCESS_PO_007.xsd)
```

サービスのスキーマの場所は、サービスの WSDL URL (例:

```
http://<soa_suite_hostname>:<port>/soa-  
infra/services/default/<jndi_name>_XMLGATEWAY_ONT_POI/ONT  
PO_I_Service?wsdl)から見つけることができます。
```

「プロジェクトにコピー」チェック・ボックスを選択して、「OK」をクリックします。

10. 「ファイルのローカライズ」ウィンドウが表示されます。「インポートされたファイルの元のディレクトリ構造を保持」チェック・ボックスが選択されていることを確認し、「OK」をクリックします。

「タイプ・チューザ」ウィンドウに、「インポートしたスキーマ」フォルダが自動的に追加されます。

11. インポートされたスキーマ・フォルダを展開し、PROCESS_PO_007.xsd から PROCESS_PO_007 を選択します。「OK」をクリックします。選択した PROCESS_PO_007.xsd が URL として表示され、PROCESS_PO_007 要素がスキーマ要素として選択されます。

「次へ」→「終了」をクリックします。ウィザードによって、パートナ・リンクに対応する WSDL ファイルが生成されます。メインの「パートナ・リンクの作成」ダイアログ・ボックスが表示され、新しい WSDL ファイル GetXMLMsg.wsdl が指定されます。

「適用」→「OK」をクリックして構成を完了し、ファイル・アダプタ・サービスに必要な WSDL 設定を指定してパートナ・リンクを作成します。

GetXMLMsg パートナ・リンクが BPEL プロセス・ダイアグラムに表示されます。

アプリケーション・ウィンドウで、ファイル GetXMLMsg_file.jca に移動します。DeleteFile プロパティの値を false に設定します。

Invoke アクティビティの追加

このステップでは、次のことを目的とした 3 つの Invoke アクティビティを構成します。

1. XML ファイルの GetXMLMsg パートナ・リンクを起動して、受け入れた XML メッセージの詳細を受入アクティビティから取得します。
2. XML ファイルの ONT_POI パートナ・リンクを起動して、発注情報を ECX_INBOUND キューにエンキューします。

XML メッセージを取得するためのパートナ・リンクの 1 つ目の Invoke アクティビティを追加するには:

1. Oracle JDeveloper で、コンポーネント・パレットから「BPEL コンストラクト」を展開します。Invoke アクティビティを、プロセス・ダイアグラムの中央のスイムレーンの、receiveInput と replyOutput のアクティビティの間にドラッグ・アンド・ドロップします。
2. Invoke アクティビティを GetXMLMsg サービスにリンクします。「編集 Invoke」ダイアログ・ボックスが表示されます。
3. Invoke アクティビティの名前を入力し(例: Invoke_Msg)、「入力変数」フィールドの横にある「作成」アイコンをクリックして新しい変数を作成します。「変数の作成」ダイアログ・ボックスが表示されます。
4. 入力変数名を入力します。デフォルト名を受け入れることもできます。「グローバル変数」を選択して、「OK」をクリックします。
5. 「出力変数」フィールドの横にある「入力変数の作成」アイコンをクリックして、新しい変数を作成します。「変数の作成」ダイアログ・ボックスが表示されます。
6. 入力変数名を入力します。デフォルト名を受け入れることもできます。「グローバル変数」を選択して、「OK」をクリックします。
7. 「編集 Invoke」ダイアログ・ボックスで「適用」→「OK」をクリックして、Invoke アクティビティ

ティの構成を終了します。

1 つ目の **Invoke** アクティビティがプロセス・ダイアグラムに表示されます。

発注情報をエンキューするためのパートナ・リンクの 2 つ目の **Invoke** アクティビティを追加するには:

1. Oracle JDeveloper で、コンポーネント・パレットから「**BPEL コンストラクト**」を展開します。2 つ目の **Invoke** アクティビティをプロセス・ダイアグラムの中央のスイムレーン(1 つ目の **Invoke** アクティビティと **replyOutput** アクティビティの後)にドラッグ・アンド・ドロップします。
2. **Invoke** アクティビティを **ONT_POI** サービスにリンクします。「編集 **Invoke**」ダイアログ・ボックスが表示されます。
3. **Invoke** アクティビティの名前を入力し、「**入力変数**」フィールドの横にある「**作成**」アイコンをクリックして新しい変数を作成します。「変数の作成」ダイアログ・ボックスが表示されます。
4. 入力変数名を入力します。デフォルト名を受け入れることもできます。「**グローバル変数**」を選択して、「**OK**」をクリックします。
5. 「編集 **Invoke**」ダイアログ・ボックスで「**適用**」→「**OK**」をクリックして、**Invoke** アクティビティの構成を終了します。
6. プロセス・ダイアグラムが表示されます。

Assign アクティビティの追加

このステップでは、次のことを目的とした 2 つの **Assign** アクティビティを構成します。

1. メッセージをエンキューするための **Invoke** アクティビティに XML メッセージを入力として渡します。
2. コンテキスト情報を Oracle E-Business Suite に提供するために、**Invoke** アクティビティに XML ゲートウェイ・ヘッダー変数を入力として渡します。

XML メッセージを入力値として **Invoke** アクティビティに渡すための 1 つ目の **Assign** アクティビティを追加する:

1. Oracle JDeveloper で、コンポーネント・パレットから「**BPEL コンストラクト**」を展開します。**Assign** アクティビティを、プロセス・ダイアグラムの中央のスイムレーンの、先ほど作成したばかりの 2 つの **Invoke** アクティビティの間にドラッグ・アンド・ドロップします。
2. **Assign** アクティビティをダブルクリックして、「編集 **Assign**」ダイアログ・ボックスにアクセスします。
3. 「一般」タブをクリックして、**Assign** アクティビティの名前を入力します(例: **SetOrderXML**)。
4. 「コピー・ルール」タブを選択して、ソース・ツリーとターゲット・ツリーを展開します。
 - 「元」のナビゲーション・ツリーで、「**変数**」→「**プロセス**」→「**変数**」→「**Invoke_Msg_SynchRead_OutputVariable**」に移動し、「**Process_PO_007**」を選択します。

- 「先」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「Invoke_ONT POI_InputVariable」に移動し、「body」を選択します。

ソース・ノード(Process_PO_007)をドラッグして、先ほど識別したターゲット・ノード(body)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

5. 「適用」→「OK」をクリックして、Assign アクティビティの構成を完了します。

Invoke アクティビティに XML ゲートウェイ・ヘッダー変数を渡すための 2 つ目の Assign アクティビティを追加するには:

1. Oracle JDeveloper で、コンポーネント・パレットから「BPEL コンストラクト」を展開します。**Assign** アクティビティをプロセス・ダイアグラムの中央のスイムレーン(先ほど作成した 1 つ目の Assign アクティビティと 2 つ目の **Invoke** アクティビティの間)にドラッグ・アンド・ドロップします。

Assign アクティビティをプロセス・ダイアグラムの中央のスイムレーン(SetOrderXML**Assign** アクティビティと 2 つ目の **Invoke** アクティビティの間)にドラッグ・アンド・ドロップして、2 つ目の Assign アクティビティを追加します。

2. 1 つ目の Assign アクティビティを作成する際に説明したステップ 2 とステップ 3 を繰り返して、SetECXHeader という 2 つ目の Assign アクティビティを追加します。
3. 「コピー・ルール」タブを選択して、ソース・ツリーとターゲット・ツリーを展開します。

- 「式」アイコンをクリックして、「式ビルダー」ダイアログを起動します。

「式」ボックスに'XML'と入力します。「OK」をクリックします。「割当ての編集」ダイアログの中央の、「元」と「先」の各ナビゲーション・ツリー・ノードの間に、式の値(XML)を持つ「式」アイコンが表示されます。

- 「先」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「Invoke_ONT POI_InputVariable」→「header」→「ns1:SOAHeader」→「ns4: ECXMSG」に移動し、「MESSAGE_TYPE」を選択します。

「XPath」フィールドに、選択したエントリが含まれる必要があります。

「式」アイコンをドラッグして、先ほど識別したターゲット・ノード(MESSAGE_TYPE)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

4. ステップ 3 で説明したものと同じメカニズムを使用して、次のパラメータに値を割り当てます。
 - MESSAGE_STANDARD: 'OAG'
 - TRANSACTION_TYPE: 'PO'

- TRANSACTION_SUBTYPE: 'PROCESS'
- DOCUMENT_NUMBER: 'PO-4466-5'
- PARTY_SIDE_ID: 'HIL-Tulsa'

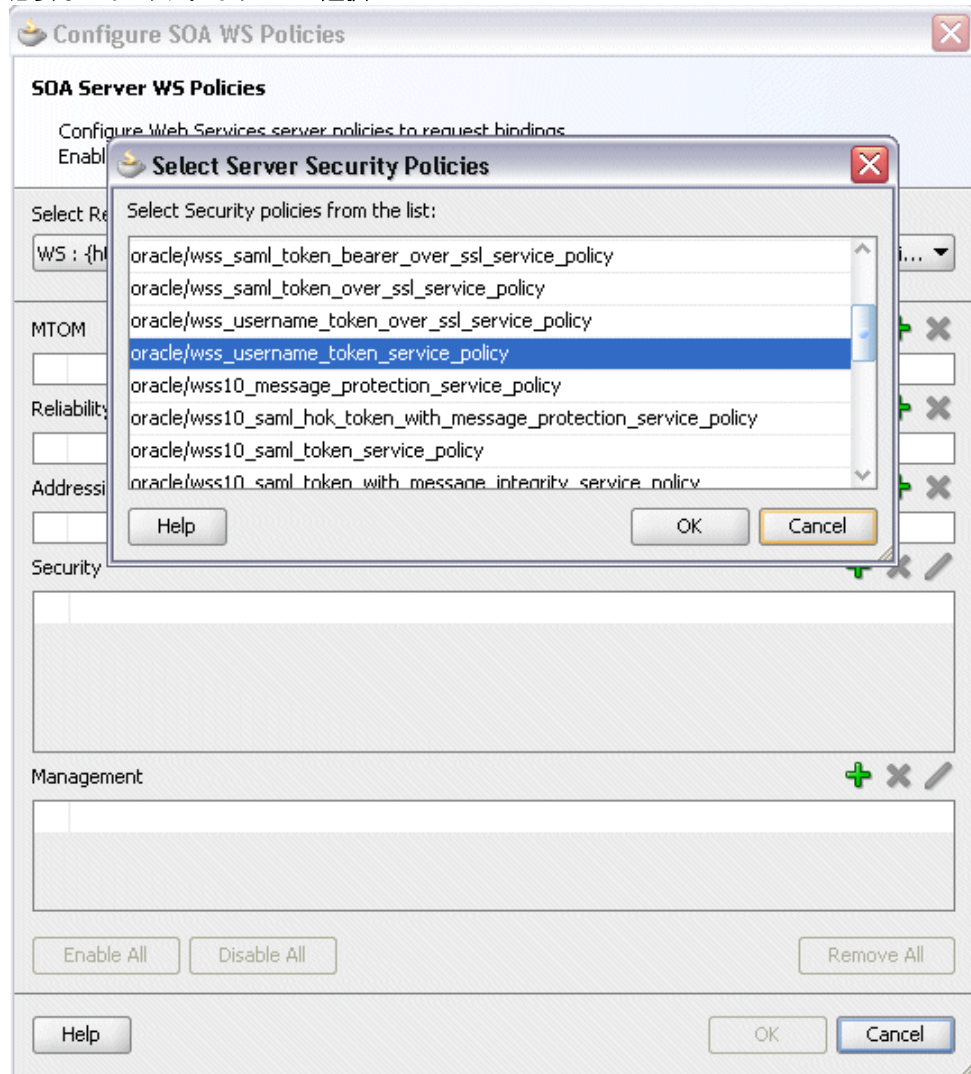
5. 「適用」→「OK」をクリックして、Assign アクティビティの構成を完了します。

Web サービス・ポリシーの構成

次のステップを使用して、設計時にセキュリティ・ポリシーを追加します。

1. 「SOA コンテンツ」→「ビジネス・ルール」→composite.xml に移動します。ONT POI サービスを右クリックし、ドロップダウン・リストから「WS ポリシーの構成」を選択します。
2. 「SOA WS ポリシーの構成」ダイアログが表示されます。
「セキュリティ」セクションで「**セキュリティ・ポリシーの追加**」アイコン(+)をクリックします。
サーバー・セキュリティ・ポリシーの選択ダイアログが表示されます。

必要なセキュリティ・ポリシーの選択



「oracle/wss_username_token_service_policy」を選択して、「OK」をクリックします。

添付されたセキュリティ・ポリシーが「セキュリティ」セクションに表示されます。

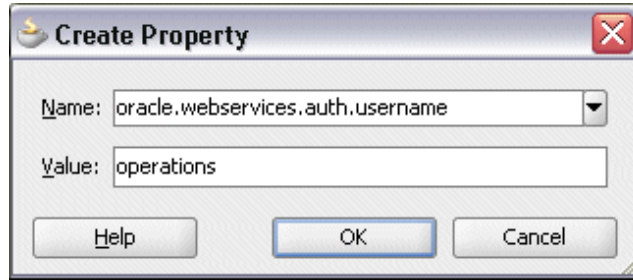
composite.xml の ONT POI サービスに、セキュリティ・ポリシーが正常に添付されたことを示すロック・アイコンが表示されます。

- ナビゲーション・メニューから、「表示」→「プロパティ・インスペクタ」を選択して、ONT POI サービス・コンポーネントの「プロパティ・インスペクタ」ウィンドウを表示します。

「プロパティ」セクションで、バインディング・プロパティの「セキュリティ・ポリシーの追加」アイコン(+)をクリックします。「プロパティの作成」ダイアログが表示されます。

「名前」フィールドに oracle.webservices.auth.username と入力して、値として operations と入力します。

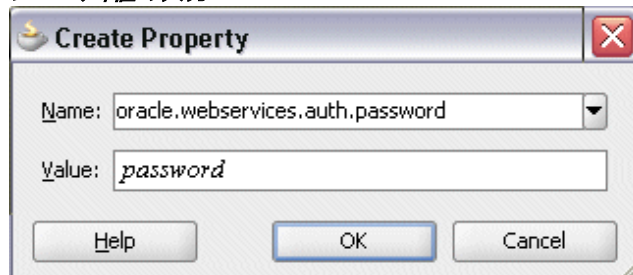
プロパティ値の入力

A screenshot of a 'Create Property' dialog box. The 'Name' field contains 'oracle.webservices.auth.username' and the 'Value' field contains 'operations'. There are 'Help', 'OK', and 'Cancel' buttons at the bottom.

「OK」をクリックします。

4. 「プロパティ」セクションのバインディング・プロパティで「セキュリティ・ポリシーの追加」アイコン(+)を再度クリックする、同じアプローチを使用します。「名前」フィールドに `oracle.webservices.auth.password` と入力します。「値」フィールドに、operations ユーザーに紐づくパスワードを入力します。

プロパティ値の入力

A screenshot of a 'Create Property' dialog box. The 'Name' field contains 'oracle.webservices.auth.password' and the 'Value' field contains 'password'. There are 'Help', 'OK', and 'Cancel' buttons at the bottom.

「OK」をクリックします。

「プロパティ」セクションに、選択したプロパティの名前と値の両方が表示されます。
composite.xml の「ソース」タブをクリックして、
`oracle.webservices.auth.username` と
`oracle.webservices.auth.password` というプロパティ名および関連する値が
ONT__POI 参照に追加されていることを確認します。

プロパティ情報の確認

```
30 <reference name="ONT_POI" ui:wsdlLocation="ONT_POI.wsdl">
31 <interface.wsdl interface="http://xmlns.oracle.com/apps/cin/soapprovider/xmlgateway/ont_poi/#wsdl.interface(ON
32 <binding.ws port="http://xmlns.oracle.com/apps/cin/soapprovider/xmlgateway/ont_poi/#wsdl.endpoint(ONT_POI_Ser
33 location="ONT_POI.wsdl" soapVersion="1.1">
34 <wsp:PolicyReference URI="oracle/wss_username_token_client_policy"
35 orawsp:category="security" orawsp:status="enabled"/>
36 <property name="oracle.webservices.auth.username" type="xs:string"
37 many="false" override="may">operations</property>
38 <property name="oracle.webservices.auth.password" type="xs:string"
39 many="false" override="may">password</property>
40 </binding.ws>
41 </reference>
```

実行時における BPEL プロセスによる SOA コンポジットのデプロイおよびテスト

SOA コンポジットに含まれている BPEL クライアントから同期 XML ゲートウェイ・インバウンド・マップ(PROCESS_PO_007)サービスを起動するには、SOA コンポジットを Oracle WebLogic 管理対象サーバーにデプロイする必要があります。これは、Oracle JDeveloper を使用して達成できます。コンポジットがデプロイされたら、Oracle Enterprise Manager Fusion Middleware Control コンソールからテストできます。

前提条件

Oracle JDeveloper を使用して BPEL プロセスによる SOA コンポジットをデプロイする前に、設計時環境とランタイム・サーバーの間の接続を確立する必要があります。必要なサーバー接続を構成する方法の詳細は、「サーバー接続の構成」(ページ B-1)を参照してください。

注意: WebLogic サーバーのローカル・インスタンスが使用される場合、Oracle JDeveloper から「実行」→「サーバー・インスタンスの起動」を選択して、WebLogic サーバーを起動します。WebLogic 管理サーバーの DefaultServer インスタンスが正常に起動されたら、「実行中:DefaultServer」と「メッセージ」のログに<Server started in Running mode>と DefaultServer started のメッセージが表示されるはずです。

ペイロードの詳細は、「インバウンド・プロセス発注 XML 取引のためのサンプル・ペイロード」(ページ C-3)を参照してください。

次の実行時タスクを実行します。

1. BPEL プロセスによる SOA コンポジットのデプロイ、ページ 5-22
2. SOA コンポジット・アプリケーションのテスト、ページ 5-26

BPEL プロセスによる SOA コンポジットのデプロイ

SOA コンポジットを実行するには、先ほど作成した BPEL プロセス (XMLGatewayInbound.bpel)でデプロイする必要があります。

注意: XML ゲートウェイ・インバウンド・サービスの SOA コンポジットをデプロイする前に、次のことを行う必要があります。

- SOA Suite サーバーの指定したディレクトリ(/usr/tmp/フォルダ)に order_data_xm1g.xml ファイルをロードします。
- <REFERENCEID>および<POID>の値を入力して(例: PO-4466-5)、入力ファイル order_data_xm1g.xml を編集します。

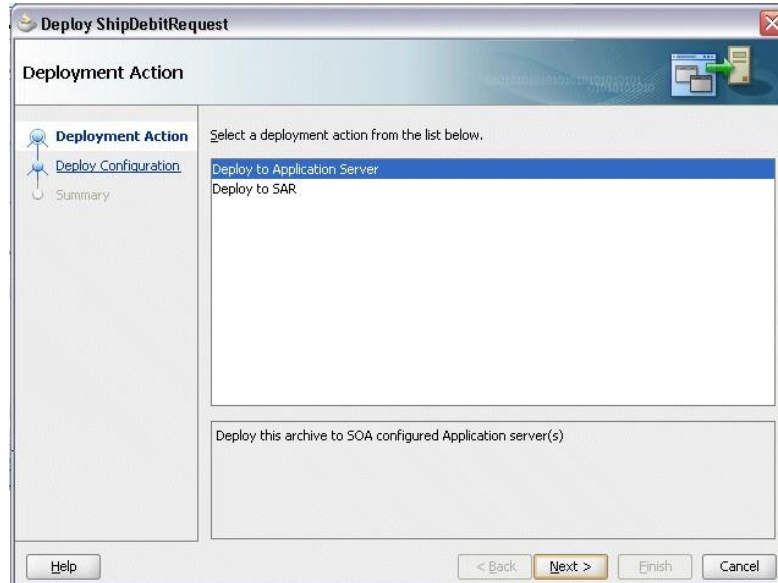
BPEL プロセスによる SOA コンポジットをデプロイするには:

1. JDeveloper のアプリケーション・ナビゲータで、**XMLGInbound** プロジェクトを選択します。
2. プロジェクトを右クリックして、メニューから「**デプロイ**」→「[project name]」→「[serverConnection]」を選択します。

たとえば、接続を適切に設定していれば、「**デプロイ**」→「**XMLGInbound**」→「**SOAServer**」を選択してプロセスをデプロイできます。

注意: サーバー接続を初めて設定する場合は、「デプロイメント・アクション」ダイアログが表示されます。「アプリケーション・サーバーにデプロイ」を選択して、「次へ」をクリックします。

「デプロイメント・アクション」ダイアログ



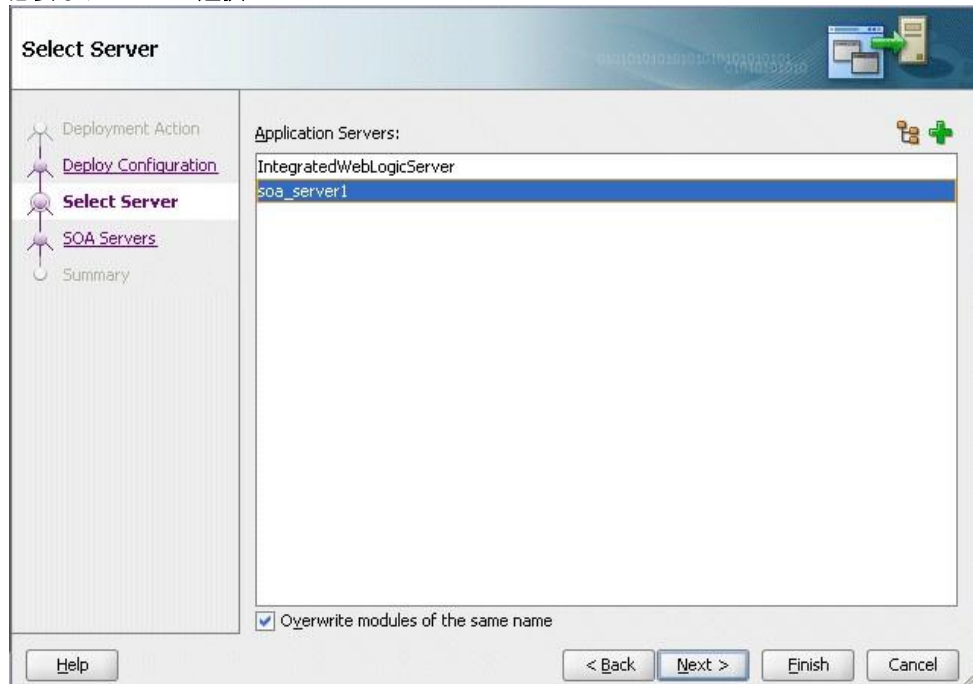
「構成のデプロイ」ダイアログで、「次へ」をクリックして新しいアプリケーション・サーバーを追加する前に、次の情報が選択されるようにします。

- 新規リビジョン ID: 1.0
- コンポジット・リビジョンをデフォルトとしてマークします。: このチェック・ボックスを選択します。
- 同じリビジョン ID で既存のコンポジットを上書きします。: このチェック・ボックスを選択します。

JDeveloper から新しい Oracle WebLogic Server 接続を作成するステップは、「サーバー接続の構成」(ページ B- 1)を参照してください。

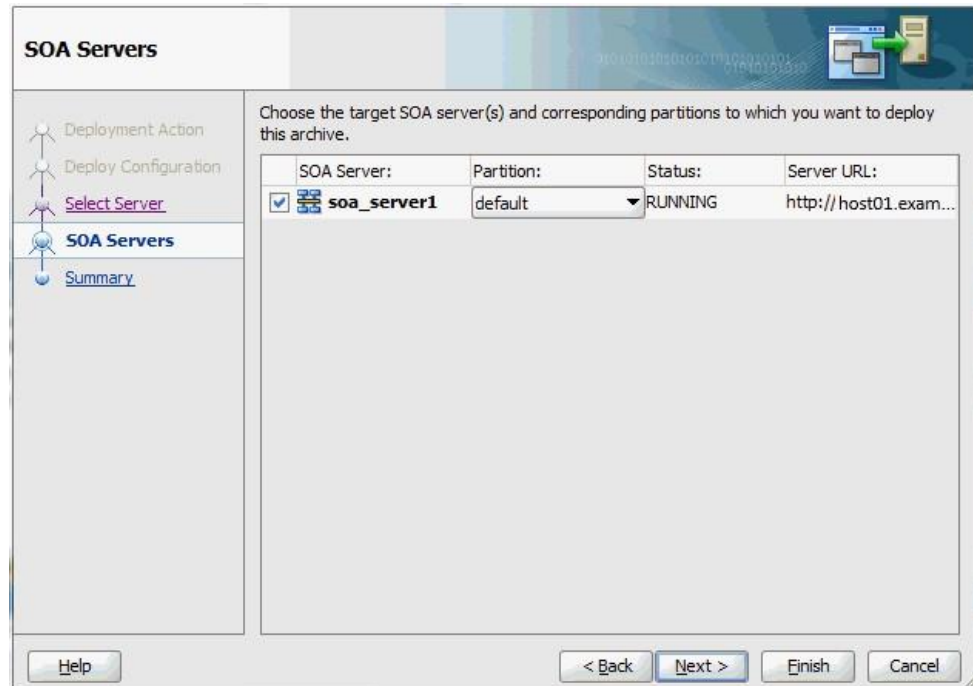
3. 「サーバーの選択」ダイアログで、以前にサーバー接続を確立した soa-server1 を選択します。「次へ」をクリックします。

必要なサーバーの選択



4. 「SOA サーバー」ダイアログで、デフォルトのターゲット SOA サーバー(soa-server1)の選択を受け入れます。

必要な SOA サーバーの選択

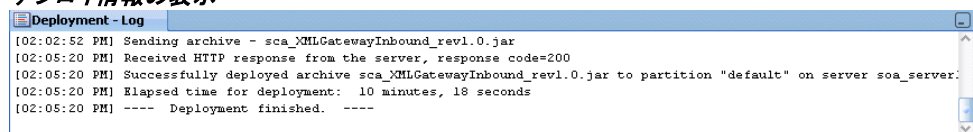


「次へ」→「終了」をクリックします。

5. Oracle JDeveloper セッションから初めてコンポジットをデプロイする場合は、「認証リクエスト」ウィンドウが表示されます。Oracle SOA Suite のインストール時に指定したユーザー名とパスワード情報を入力します。「OK」をクリックします。
6. デプロイ処理が開始します。SOA - ログ・ウィンドウで、デプロイ・プロセスをモニターして、コンパイルが成功したことを確認します。

デプロイメント - ログ・ウィンドウで、デプロイが成功したことを確認します。

デプロイ情報の表示



BPEL プロセスによる SOA コンポジット・アプリケーションのテスト

SOA コンポジット・アプリケーションに含まれる BPEL プロセスが正常にデプロイされたら、Oracle Enterprise Manager Fusion Middleware Control コンソールからプロセスを管理およびモニターできます。

プロセスを手動で開始して、プロセスと統合インタフェースをテストすることもできます。

Oracle SOA Suite の詳細は、*Oracle Fusion Middleware Oracle SOA Suite 管理者ガイド*を参照してください。

BPEL プロセスによる SOA コンポジット・アプリケーションをテストするには:

1. Oracle Enterprise Manager Fusion Middleware Control コンソール
(<http://<hostname>:<port>/em>)にナビゲートします。ログイン・ページが表示されます。

2. インストール時に指定したユーザー名およびパスワード情報を入力し、「ログイン」をクリックしてファームにログインします。デプロイしたコンポジット(XMLGINbound)がアプリケーション・ナビゲーション・ツリーに表示されます。

複数のターゲット Oracle Enterprise Manager Fusion Middleware Control コンソール・ファームがある場合、適切なターゲット・インスタンス・ファームを選択する必要がある場合があります。

3. 「ファーム」のナビゲーション・ペインから、ツリーの「SOA」→「soa-infra」ノードを展開して、「SOA インフラストラクチャ」のホーム・ページおよびメニューを使用して、soa-infra 管理対象サーバーで実行されているデプロイ済の SOA コンポジット・アプリケーションにアクセスします。

XMLGINbound [1.0]のリンクをクリックします。

4. 「ポリシー」タブをクリックし、前に設計時に ONT POI サービス・バインディングに添付した oracle/wss_username_token_service_policy ポリシーがここに表示されていることを確認します。
5. XMLGINbound [1.0]のホーム・ページで、「テスト」をクリックします。
6. インスタンスを開始するための「Web サービスのテスト」ページが表示されます。

注意: 設計時に WS-Security 資格証明が入力されない場合、「リクエスト」タブの上部にある「セキュリティ」セクションの「WSS ユーザー名トークン」オプションを選択して、実行時に資格証明を入力できます。「ユーザー名」フィールドに operations、「パスワード」フィールドに、operations ユーザーに紐づくパスワードを入力します。

WS-Security 資格証明の入力

The screenshot shows a configuration window for WS-Security. It has two tabs: 'Request' and 'Response'. The 'Request' tab is active. Below the tabs is a section labeled 'Security'. Inside this section, there are four radio buttons: 'WSS Username Token' (which is selected), 'HTTP Basic Auth', 'Custom Policy', and 'None'. Below the radio buttons, there are two input fields. The first is labeled 'Username' and contains the text 'operations'. The second is labeled 'Password' and contains a series of dots, indicating a masked password.

7. プロセスに必要な入力文字列(例: test)を入力し、「Web サービスのテスト」をクリックしてプロセスを開始します。

完了時に、「レスポンス」タブにテスト結果が表示されます。

8. 「インスタンス」タブをクリックします。SOA コンポジット・アプリケーションのインスタンス ID、名前、対話 ID、各インスタンスの最新の状態(ページの前のデータ・リフレッシュ以降に確認されたもの)が表示されます。
9. BPEL サービス・コンポーネント・インスタンスのリンク(例: XMLGInbound)をクリックして、「インスタンス」ページを表示します。ここでは、「監査証跡」タブで BPEL アクティビティの実行の詳細を参照できます。

「フロー」タブをクリックして、BPEL プロセスのフロー・ダイアグラムを確認します。プロセス・ダイアグラムのアクティビティをクリックして、アクティビティの詳細と、プロセスを介したペイロードのフローを表示します。

Oracle E-Business Suite でのレコードの検証

BPEL プロセスが正常に開始されて完了したら、Oracle E-Business Suite で取引を検証できます。

Oracle Transaction Monitor で取引を検証するには:

取引モニターから検証できます。取引モニターは、XML ゲートウェイによって処理されて Oracle Transport Agent によって配信または受け入れられた、Oracle E-Business Suite との間のインバウンドおよびアウトバウンド取引のステータスをモニターするためのツールです。取引モニターには、これらの文書の完全な履歴と監査証跡が表示されます。

1. 「ワークフロー管理者 Web アプリケーション」職責を持つユーザーとして Oracle E-Business Suite にログインします。

取引モニターのリンクを選択して、受注を検索するための検索ウィンドウを開きます。

取引モニターからの検索

Transaction Monitor:Search

Search Criteria

Select search criteria and press Go to view the report.

☒ Inbound Messages
Processing Status

☐ Outbound Messages
Generation Status
Delivery Status
Retry Status

Transaction Type Transaction Subtype
Source TP Location Code Trading Partner Name
Document ID Site Name
Party Type
From Date
To Date

Copyright (c) 1998, 2015, Oracle and/or its affiliates. All rights reserved. [Privacy Statement](#)

- 「日付:自」フィールドと「日付:至」フィールドを消去し、「文書 ID」フィールドに PO-4466-5 と入力します。
- 「パーティ・タイプ」として「顧客」を選択します。「進む」をクリックして、検索を実行します。
「インバウンド検索結果」リージョンに XML インバウンド取引「PO-4466-5」が表示されます。
- 取引「PO-4466-5」のステータスが「成功」であることを確認します。

これを確認するには、「受注管理スーパーユーザー, Vision オペレーション (USA)」職責で Oracle E-Business Suite にログインします。「受注, 返品: 受注のインポート」→「訂正」を選択します。「受注の検索」ウィンドウが表示されます。「受注ソース」フィールドのドロップダウン・リストから「XML」を選択します。「訂正」ウィンドウに、受注ソースが XML であるすべての取引が表示されます。

「当初システム文書参照」フィールドに PO-4466-5 がリストされていることを確認します。

「訂正」ウィンドウでの取引の確認

Order S	Orig Sys Doc	Change	Request ID	Operation Code	Order Number	Sold To Org ID
20	PO-4466-5		42907902			1004
20	order_id_01		39716466			3347
20	order_id_01		39716468			3347

Payments Discounts Pricing Attributes Sales Credits

Actions Add Customers Errors Validate

Lines Import

Oracle Order Management に受注をインポートするには:

1. 「受注管理スーパーユーザー, Vision オペレーション (USA)」職責を持つユーザーとしてフォーム・ベースの Oracle E-Business Suite にログインします。
2. 「受注, 返品 : 受注のインポート」→「受注インポート要求」を選択します。「受注インポート要求」ウィンドウが「パラメータ」ダイアログとともに表示されます。
「受注参照」ドロップダウン・リストから「PO-4466-5」を選択します。

受注インポート要求

3. 「パラメータ」ダイアログで「OK」をクリックします。「受注インポート」という要求名がインポート要求ウィンドウに自動的に移入されます。
4. 「発行」をクリックして要求を発行します。コンカレント要求番号が表示されます。この要求番号を記録します。ただし、「決定」ダイアログで「いいえ」をクリックして、別の要求は発行しません。
5. アプリケーション・メニューから、「要求」→「表示」を選択して「要求の検索」ウィンドウを開きます。
6. 先ほど記録した要求番号を入力し、「検索」をクリックします。
「受注インポート」要求のステータスが表示されます。ステータスは成功として表示され、Order Management アプリケーションで受注が作成されます。

Oracle Order Management で取引を検証するには:

1. 「受注管理スーパーユーザー」職責を持つユーザーとしてフォーム・ベースの Oracle E-Business Suite にログインします。
2. 「受注返品」→「受注」を選択します。「受注」フォームが表示されます。
3. 「顧客発注」フィールドに発注番号を入力して(例: PO-4466-5)、受注を検索します。新しく作成された受注の詳細が表示されます。

受注

Sales Orders (Vision Operations) - 65182, Hilman and Associates

Order Information | Line Items

Default

Main | Others

Customer: Hilman and Associates
Customer Number: 1004
Customer PO: PO-4466-5
Customer Contact: Cowlshaw, Marcus Mr.
Ship To Location: Tulsa (OPS)
1230 East 60th Street South
Tulsa, OK, 74146, US
Bill To Location: Tulsa (OPS)
1230 East 60th Street South
Tulsa, OK, 74146, US

Order Number: 65182
Order Type: Mixed
Date Ordered: 04-SEP-2012 23:26:29
Price List: Corporate
Salesperson: Cleary, Mr. Timothy
Status: Entered
Currency: USD
Subtotal: 26,156.40
Tax: 0.00
Charges: 150.00
Total: 26,306.40

Actions | Related Items | Configurator | Availability | Book Order

サブスクリプション・モデルを介した XML ゲートウェイ・アウトバウンドの使用

この項では、次のトピックを取り上げます。

- 設計時に新しい BPEL プロセスによる SOA コンポジット・アプリケーションを作成する場合の XML ゲートウェイ・アウトバウンド・メッセージの使用、ページ 5-32
- 実行時における BPEL プロセスによる SOA コンポジット・アプリケーションのデプロイおよびテスト、ページ 5-54

設計時における XML ゲートウェイ・アウトバウンド・サービスの使用

アウトバウンド XML ゲートウェイ・マップ・インタフェースの場合、アウトバウンド・メッセージは最初に ECX_OUTBOUND キューにエンキューされるため、Oracle BPEL PM は、同じ関連識別子を持つメッセージを ECX_OUTBOUND キューでリスニングします。

Oracle E-Business Suite Integrated SOA Gateway は、ECX_OUTBOUND キュー内のメッセージをデキューして Oracle E-Business Suite からアウトバウンド・データを取得することにより、サブスクリプション・モデルを介してこれをサポートします。取得したデータは、メッセージを購読している取引先やコンシューマに渡すことができます。

BPEL プロセスによる SOA コンポジット・アプリケーションのシナリオ

XML ゲートウェイ・アウトバウンド・インタフェース「発注確認 XML 取引」を例として使用します。

この XML ゲートウェイ・アウトバウンド・インタフェースは、ECX_CBODO_OAG72_OUT アウトバウンド・マップを介して Web サービスとして公開されます。

発注が作成されて承認される際、発注の承認時に、発注確認フローを作成し、発注確認を XML ファイルとして送信するワークフローがトリガーされます。このワークフローは、他のシステムに配信するために、確認 BOD を発注確認として ECX_OUTBOUND キューに配信します。

このメッセージの相関 ID は BPEL に設定され、Oracle BPEL PM は、相関識別子= 'BPEL' のメッセージを ECX_OUTBOUND キューでリスニングします。発注確認としての確認 BOD は、ファイル・アダプタを使用して出力 XML ファイルとして書き込まれます。

デプロイメント後、BPEL プロセスが正常に実行されると、発注が承認されたときに、出力 XML ファイル内の同じ受注記帳参照 ID (顧客発注)が取得されます。

XML ゲートウェイ・アウトバウンド・メッセージングを使用して BPEL プロセスを作成するための前提条件

Oracle E-Business Suite で相関識別子を設定する必要があります。アウトバウンド・キューをリスニングしているエージェントが複数ある場合は、相関識別子を使用すると、特定のエージェント向けのメッセージにラベルを付けることができます。特定の相関をリスニングしているエージェントは、エージェントの相関識別子と一致するメッセージを捕捉します。

相関識別子を設定するには:

1. 「XML ゲートウェイ」職責を持つユーザーとして Oracle E-Business Suite にログインします。「ナビゲータ」ページが表示されます。
2. 「XML ゲートウェイ」リンクをクリックします。
3. 「XML ゲートウェイ」の下にある「**参照値の定義**」リンクをクリックします。
4. 「タイプ」フィールドで COMM_METHOD を検索して、システムに存在するかどうかを確認します。

XML Gateway の参照ウィンドウ

Code	Meaning	Description	Tag	From	To	Enabled
BPEL	BPEL	BPEL process		07-DEC-2006		☒
HTTP	HTTP	HTTP Delivery		28-SEP-2000		☒
HTTP-ATCH	Attachment Enabled H	Attachment Enabled HT		12-DEC-2002		☒
HTTP-OXTA	Oracle Transport Agen	OXTA over HTTP		29-MAY-2000		☒
HTTP-OXTAA1	Attachment Enabled C	Attachment Enabled OX		25-OCT-2002	12-DEC-2002	☐
HTTP-WM	WebMethods using HT	HTTP with WebMethods		05-OCT-2000		☒
HTTPS	HTTPS (Secure HTTP)	HTTPS Delivery		28-SEP-2000		☒
HTTPS-ATCH	Attachment Enabled H	Attachment Enabled HT		12-DEC-2002		☒
HTTPS-OMB	Oracle Message Broke	OMB using HTTPS		01-DEC-2000		☐
HTTPS-OXTA	Oracle Transport Agen	OXTA over HTTPS		29-MAY-2000		☒

5. 「コード」フィールドと「内容」フィールドに BPEL と入力して、新しいレコードを COMM_METHOD タイプに追加します。説明を入力し、レコードを保存します。

Oracle XML Gateway は、メッセージを ECX_OUTBOUND キューにエンキューするときに BPEL の相関を設定します。

相関識別子を正しく設定することに加えて、使用する取引先の「プロトコル・タイプ」フィールドが BPEL に設定されていることを確認する必要があります。目的の取引を指定して取引先を設定する方法の詳細は、「SOA コンポジット・アプリケーションの手動テスト」(ページ 5-58) を参照してください。

「取引先設定」ウィンドウ

Transaction Type	Transaction SubType	Standard Code	External Transaction Type	External Transaction SubType	Direction	Map	Connection/Hub	Protocol Type
ONT	POA	OAG	PO	ACKNOWLEDG	OUT	ONT_3A4A_O	DIRECT	SMTP
ONT	CPO	OAG	PO	CANCEL	IN	ONT_3A9R_O		
ONT	POI	OAG	PO	PROCESS	IN	ONT_3A4R_O		
ONT	SSO	OAG	SALESORDER	SHOW	OUT	PROCESS_P	DIRECT	SMTP
ECX	CBODO	OAG	BOD	CONFIRM	OUT	ECX_CBODO	DIRECT	BPEL

BPEL プロセスによる SOA コンポジット・アプリケーションの作成フロー

発注確認 XML 取引のシナリオに基づいて、この章では次の設計時タスクについて説明します。

1. 新しい BPEL プロセスによる SOA コンポジット・アプリケーションの作成、ページ 5-36

このステップを使用して、XMLGOutbound.bpel という、新しい BPEL プロセスによる SOA コンポジット・アプリケーションを作成します。

2. AQ アダプタのパートナー・リンクの作成、ページ 5-37

このステップを使用して、ECX_OUTBOUND キューからイベント詳細をデキューします。

3. 受入アクティビティの追加、ページ 5-46

受入アクティビティを使用して、発注確認の詳細を Assign アクティビティへの入力として取得します。

4. ファイル・アダプタのパートナー・リンクの追加、ページ 5-48

ここでは、発注確認の詳細を出力ファイルとして XML ファイルに書き込みます。

5. Invoke アクティビティの追加、ページ 5-51

ここでは、ファイル・アダプタのパートナー・リンクを起動することにより、発注確認情報を XML ファイルに書き込みます。

6. Assign アクティビティの追加、ページ 5-53

受入アクティビティから出力を取得し、Invoke アクティビティに入力を渡すには、Assign アクティビティを使用します。

BPEL プロセスのサービス・コンポーネントを使用した SOA コンポジット・アプリケーションの一般情報と作成方法の詳細は、*Oracle Fusion Middleware Oracle SOA Suite 開発者ガイド*を参照してください。

新しい BPEL プロセスによる SOA コンポジット・アプリケーションの作成

このステップを使用して、様々な BPEL プロセス・アクティビティによる新規 SOA コンポジット・アプリケーションを作成します。

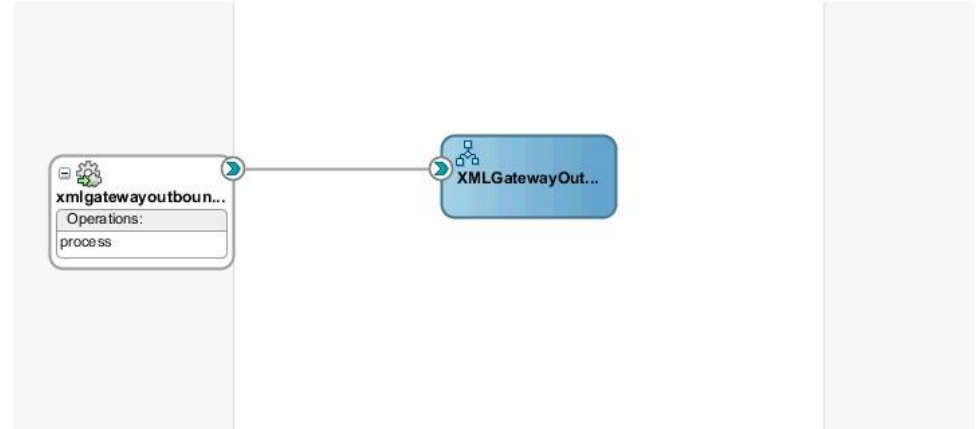
新しい BPEL プロセスによる SOA コンポジット・アプリケーションを作成するには:

1. Oracle JDeveloper を開きます。
2. アプリケーション・ナビゲータで「**新規アプリケーション**」をクリックします。「SOA アプリケーションの作成 - アプリケーションの名前付け」ページが表示されます。
3. 「アプリケーション名」フィールドにアプリケーションの適切な名前を入力して、「アプリケーション・テンプレート」リストから「**SOA アプリケーション**」を選択します。
「**次へ**」をクリックします。「SOA アプリケーションの作成 - プロジェクトの名前付け」ページが表示されます。
4. 「プロジェクト名」フィールドにプロジェクトの適切な名前を入力します。たとえば、XMLGatewayOutbound と入力します。
5. 「プロジェクト・テクノロジー」タブで、「Web サービス」を選択して、「使用可能」のテクノロジー・リストから「**選択済**」のテクノロジー・リストに「**SOA**」が選択されるようにします。
「**次へ**」をクリックします。「SOA アプリケーションの作成 - SOA 設定の構成」ページが表示されます。
6. 「コンポジット・テンプレート」リストから**コンポジット BPEL プロセス**を選択して、「**終了**」をクリックします。新規アプリケーションと SOA プロジェクトを作成しました。これにより、SOA コンポジットが自動的に作成されます。
「BPEL プロセスの作成」ページが表示されます。
7. デフォルトの「**BPEL 1.1 Specification**」の選択を変更せずに維持します。これにより、BPEL 1.1 Specification をサポートする BPEL プロジェクトが作成されます。
「名前」フィールドに BPEL プロセスの適切な名前を入力します。たとえば、XMLGOutbound とします。
「テンプレート」フィールドで「**同期 BPEL プロセス**」を選択します。
「トランザクション」ドロップダウン・リストから「**required**」を選択します。「**OK**」をクリックします。

受入アクティビティと返信アクティビティのある同期 BPEL プロセスが作成されます。また、指定した名前(例: XMLGOutbound.bpel および XMLGOutbound.wsdl)を使用した bpel および wsdl を含む必要なソース・ファイルと、composite.xml も生成されます。

7. 「SOA コンテンツ」→「ビジネス・ルール」に移動して、composite.xml をクリックして、コンポジット・ダイアグラムを表示します。

コンポジット・ダイアグラムの表示



XMLGOutbound コンポーネントをダブルクリックして BPEL プロセスを開きます。

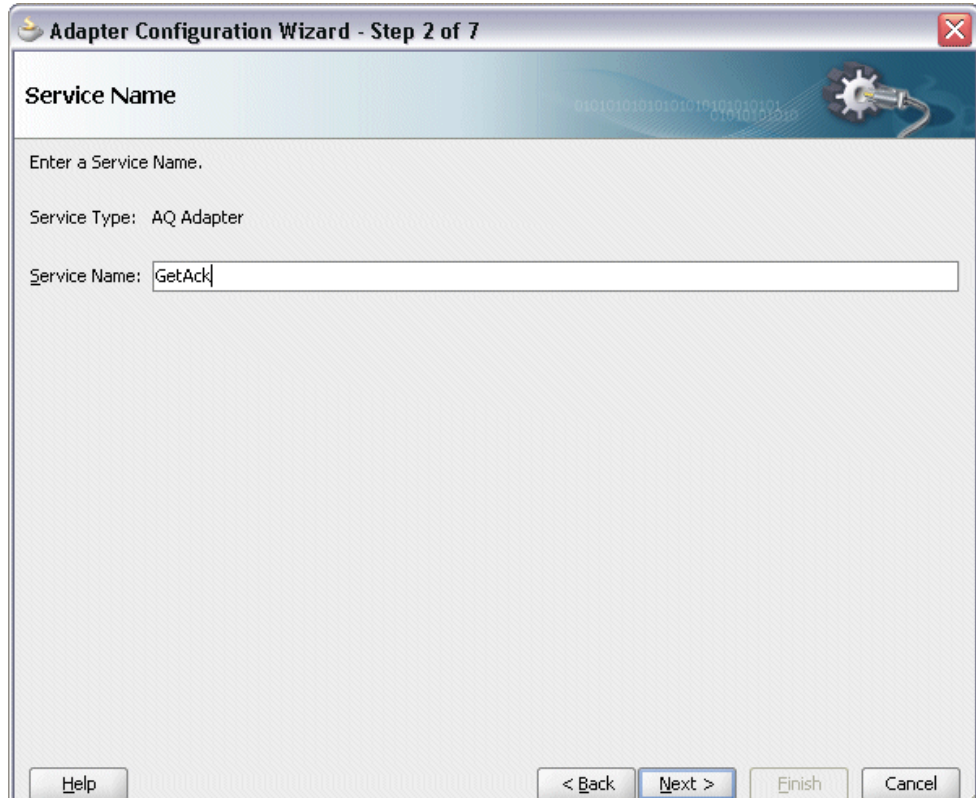
AQ アダプタのパートナ・リンクの作成

このステップを使用して、ECX_OUTBOUND キュー内の XML ゲートウェイ・アウトバウンド・メッセージ(例: ECX_CBODO_OAG72_OUT)をデキューするための AQ アダプタのパートナ・リンク(GetAck)を作成します。

AQ アダプタのパートナ・リンクを作成するには:

1. Oracle JDeveloper で、「BPEL サービス」リストにある「AQ アダプタ」サービスをプロセス・ダイアグラムの右側の「パートナ・リンク」スイムレーンにドラッグ・アンド・ドロップします。ファイル・アダプタ構成ウィザードへようこそページが表示されます。
2. 「次へ」をクリックします。「サービス名」ダイアログ・ボックスが表示されます。
3. 「サービス名」ダイアログ・ボックスでサービス名を入力します(例: GetAck)。

サービス名の入力



4. 「次へ」をクリックします。「サービス接続」ダイアログ・ボックスが表示されます。
5. 既存のデータベース接続を使用する場合は、「接続」リストからデータベース接続を選択し、新規データベース接続を定義する場合は、「新規」をクリックしてデータベース接続作成ウィザードを開きます。

注意: Oracle E-Business Suite が実行されているデータベースに接続する必要があります。

新規データベース接続を作成するには:

1. 「新規」をクリックして、データベース接続の作成ウィザードを開きます。「次へ」をクリックし、一意の接続名を入力して、データベース接続の接続タイプ(例:「Oracle (JDBC)」)を選択します。「次へ」をクリックします。
2. 「認証」ダイアログ・ボックスで、適切なユーザー名およびパスワード情報を入力してデータベース接続を認証します。「次へ」をクリックします。
3. 「接続」ダイアログ・ボックスで、次の情報を指定します。

- ドライバ: Thin
 - ホスト名: データベース接続のホスト名を入力します。たとえば、`myhost01.example.com` のようにします。
 - JDBC ポート: データベース接続の JDBC ポート番号(1521 など)を入力します。
 - SID: データベース接続の一意の SID (sid01 など)を指定します。
4. 「次へ」をクリックしてデータベース接続をテストします。
ステータス・メッセージ「成功しました。」は、接続が有効であることを示します。
 5. 「次へ」をクリックして、データベース接続のサマリーを示す「サービス接続」ダイアログ・ボックスに戻ります。
 6. 「サービス接続」ダイアログ・ボックスの「JNDI 名」フィールドには、指定したデータベース接続に対応する JNDI (Java Naming and Directory Interface)名が自動的に表示されます。かわりに、別の JNDI 名を入力することもできます。
 7. 「次へ」をクリックして「アダプタ・インタフェース」ダイアログ・ボックスを開きます。
「操作およびスキーマから定義(後で指定)」ラジオ・ボタンを選択して、「次へ」をクリックします。「操作」ダイアログ・ボックスが表示されます。
 8. 「操作タイプ」フィールドで「デキュー」ラジオ・ボタンを選択します。「操作名」フィールドにも「デキュー」が移入されます。

「操作」ダイアログ

Adapter Configuration Wizard - Step 5 of 7

Operation

The AQ Adapter supports three operations. There is a Dequeue operation that polls for incoming messages from a queue, an Enqueue operation that puts outgoing messages on a queue, and an Enqueue/Dequeue operation that puts outgoing messages on a queue and expects response messages on a queue. Specify the Operation type and Operation Name. Only one operation per Adapter Service may be defined using this wizard.

Operation Type: ☒ Dequeue
☐ Enqueue
☐ Enqueue/Dequeue

Operation Name:

Help < Back Next > Finish Cancel

9. 「次へ」をクリックして「キュー名」ダイアログ・ボックスを開きます。
「データベース・スキーマ」フィールドで「APPLSYS」を選択します。「キュー名」フィールドに ECX_OUTBOUND と入力します。

データベース・スキーマとキュー名の指定

Queue Name

Specify the database schema and the queue to be used for the service. Use the Browse button to find the queue in the specified schema.

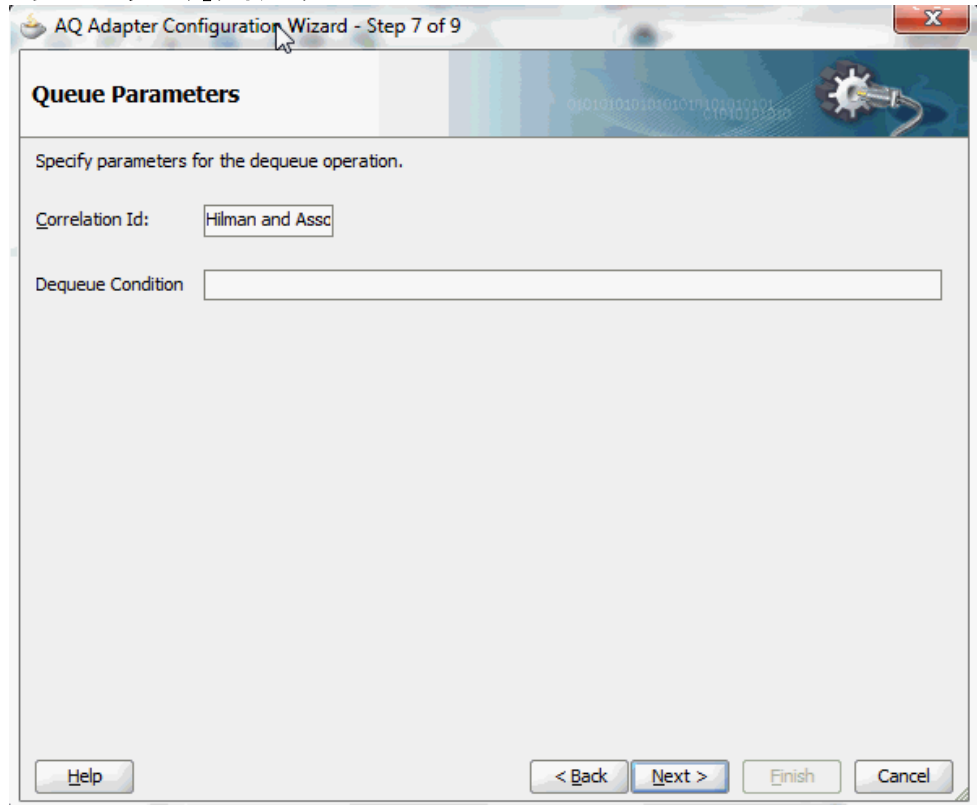
Queue Information

Database Schema:

Queue Name:

10. 「次へ」をクリックして「キュー・パラメータ」ダイアログ・ボックスを開きます。

「キュー・パラメータ」ダイアログ



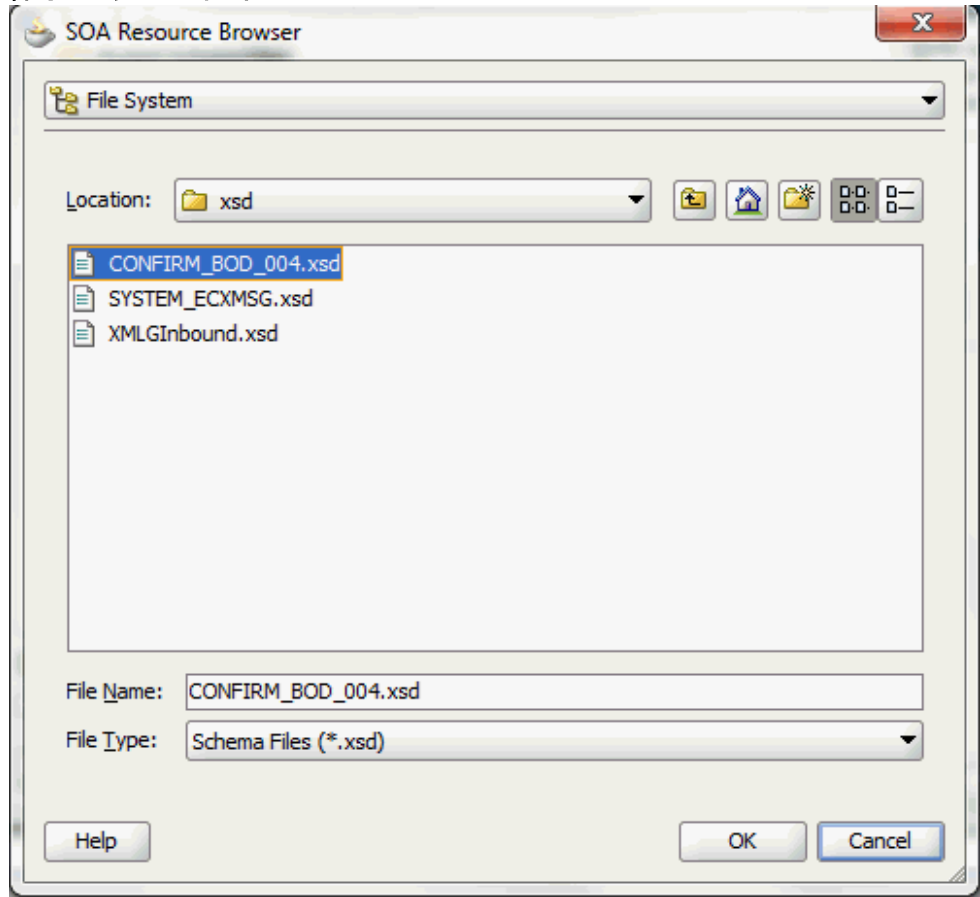
「相関 ID」フィールドに一意の顧客名を入力します(例: Hilman and Associates)。「次へ」をクリックします。「メッセージ」ダイアログ・ボックスが開きます。

11. 「スキーマ・ファイルを参照」をクリックして「タイプ・チューザ」ウィンドウを開きます。

「タイプ・チューザ」ウィンドウの右上隅にある「スキーマ・ファイルのインポート」をクリックします。これにより、「スキーマ・ファイルのインポート」ポップアップ・ウィンドウが開きます。

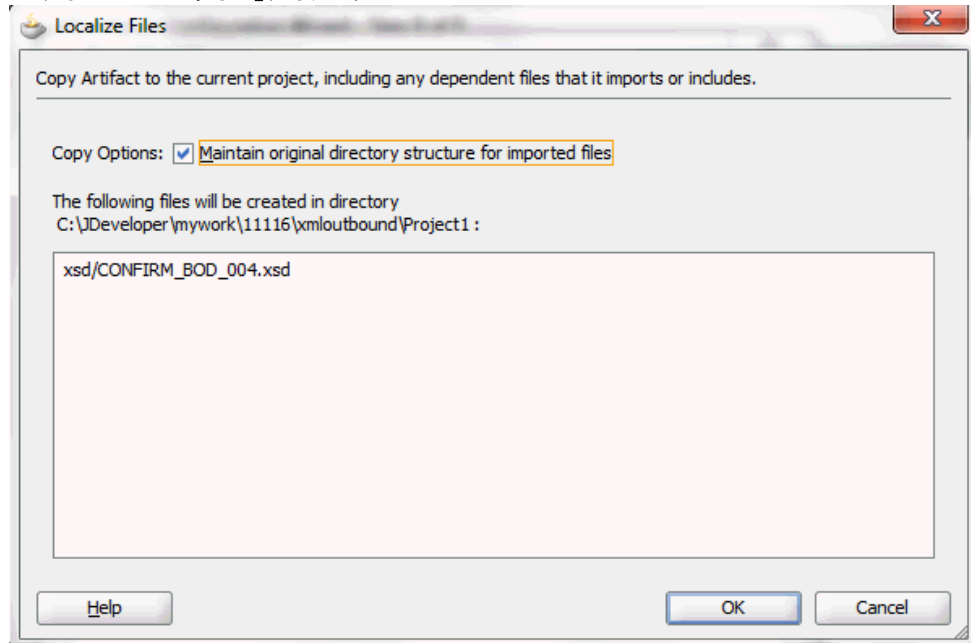
「リソースの参照」アイコンをクリックして、「SOA リソース・ブラウザ」ウィンドウを表示します。場所として xsd フォルダを選択し、フォルダから CONFIRM_BOD_004.xsd ファイルを選択します。

目的のスキーマのインポート



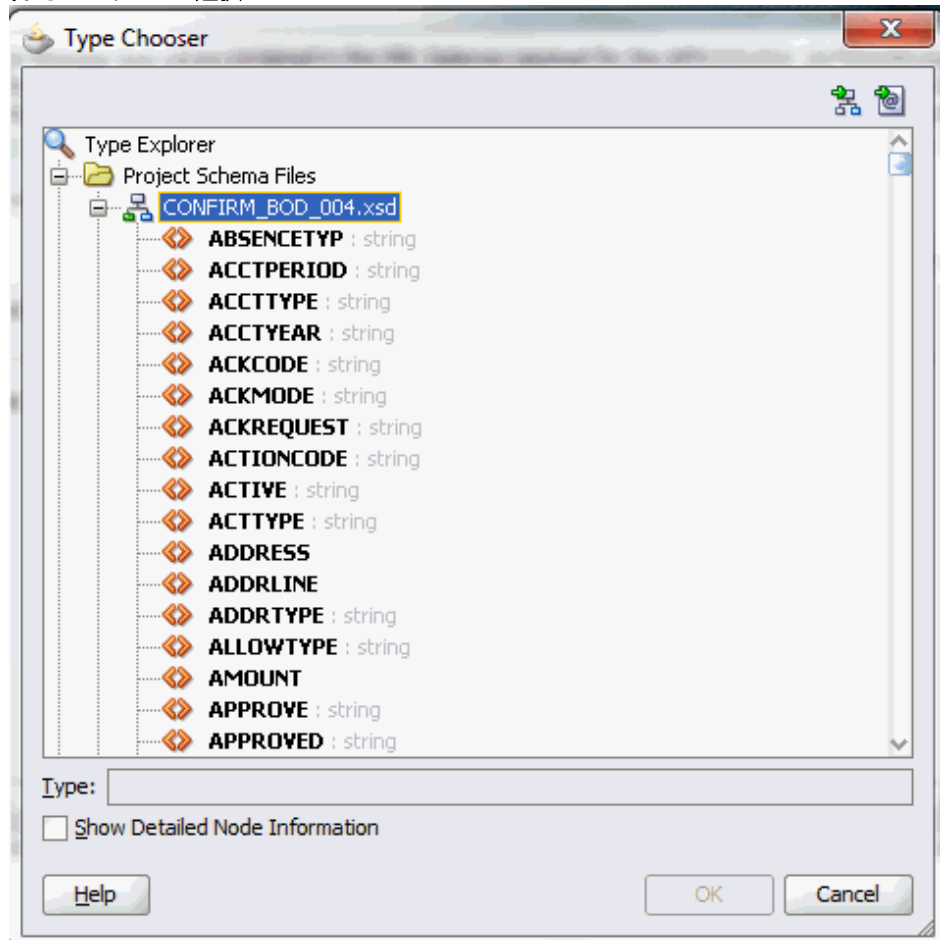
「OK」をクリックします。「ファイルのローカライズ」ダイアログ・ボックスが開きます。「OK」をクリックします。

「ファイルのローカライズ」ダイアログ



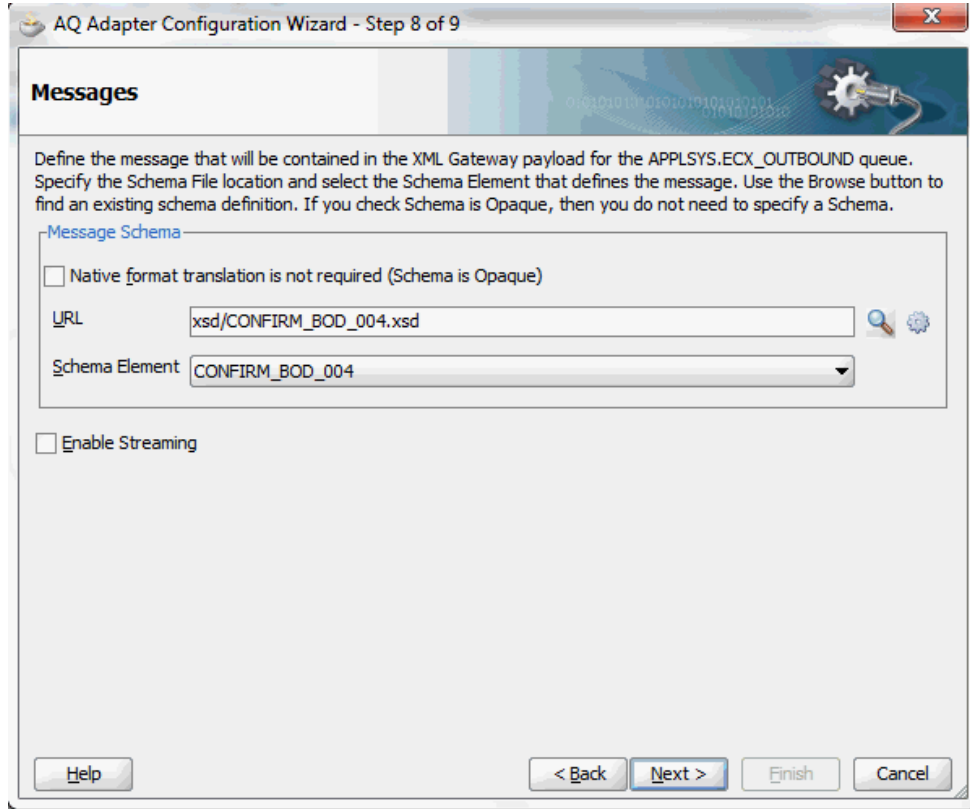
「タイプ・チューザ」ウィンドウに、選択した CONFIRM_BOD_004.xsd ファイルが表示されます。下にスクロールして、CONFIRM_BOD_004.xsd から CONFIRM_BOD_004 スキーマを選択します。「OK」をクリックします。

目的のスキーマの選択



「メッセージ」ダイアログ・ボックスに、選択したスキーマの場所とスキーマ要素が自動的に表示されます。

選択したメッセージ・スキーマの表示



12. 「次へ」をクリックして「終了」ダイアログ・ボックスに進み、GetAck サービスの AQ アダプタの定義が終了したことを確認します。

13. 「終了」をクリックします。GetAck サービスに対応する WSDL ファイルが生成されます。

「適用」→「OK」をクリックして、パートナ・リンクの構成を完了します。パートナ・リンクは、必要な WSDL 設定で作成され、BPEL プロジェクトでは、プロセス・ダイアグラムの枠線の領域の新しいアイコンによって表現されます。

受入アクティビティの追加

このステップでは、AQ アダプタについて構成したパートナ・リンク GetAck から XML データを受け入れるための受入アクティビティを構成します。

受入アクティビティから得た XML データは、次のステップで作成する Assign アクティビティに対する入力変数として使用します。

受入アクティビティを追加するには:

1. Oracle JDeveloper で、コンポーネント・パレットから「BPEL コンストラクト」を展開します。

受入アクティビティをプロセス・ダイアグラムの中央のスイムレーンにドラッグ・アンド・ドロップします。

2. 受入アクティビティを GetAck パートナ・リンクにリンクします。受入アクティビティでは、パートナ・リンクからイベント・データを取得します。「Receive の編集」ダイアログ・ボックスが表示されます。
3. 受入アクティビティの名前を入力します。
「変数」フィールドの横にある「入力変数の作成」アイコンをクリックして、新しい変数を作成します。「変数の作成」ダイアログ・ボックスが表示されます。
4. 「グローバル変数」を選択して、変数の名前を入力します。デフォルトの名前をそのまま使用してもかまいません。「OK」をクリックして、「Receive の編集」ダイアログ・ボックスに戻ります。

受入アクティビティの表示

5. 「適用」→「OK」をクリックして、受入アクティビティの構成を終了します。BPEL プロセス・ダイアグラムに受入アクティビティが表示されます。

ファイル・アダプタのパートナ・リンクの追加

このステップを使用して、発注確認を XML ファイルに書き込むことにより、パートナ・リンクを構成します。

ファイル・アダプタのパートナ・リンクを追加するには:

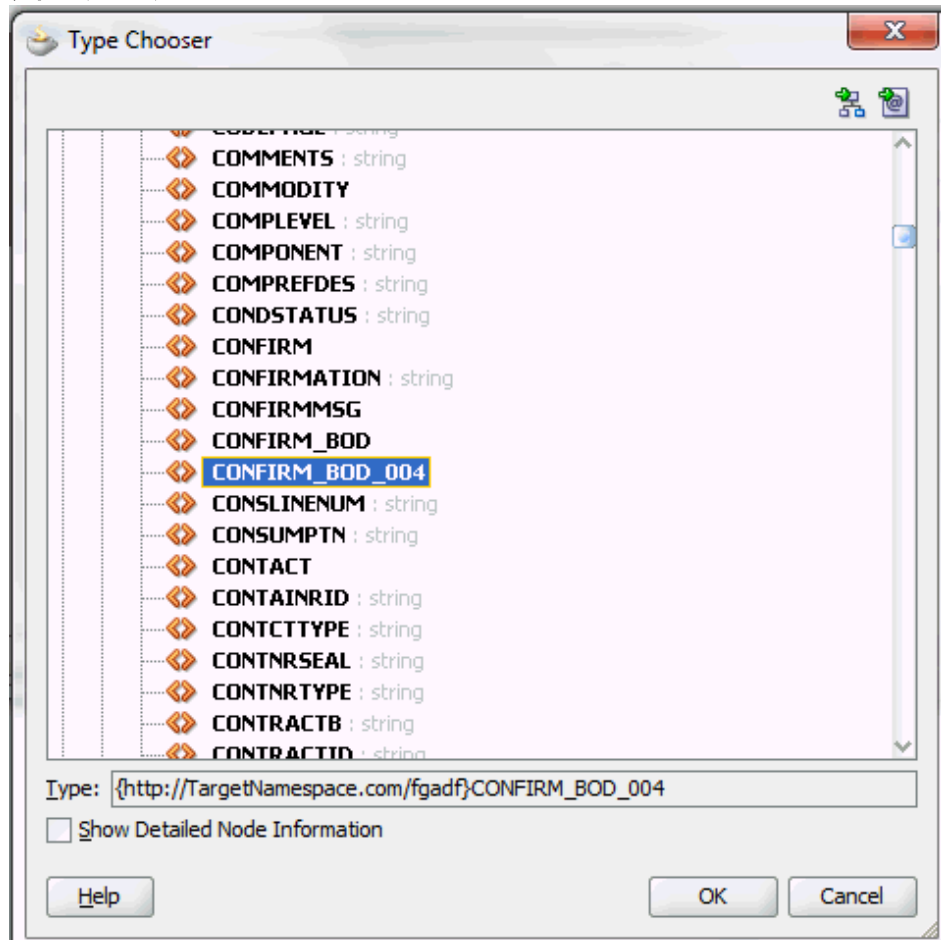
1. Oracle JDeveloper で、「**BPEL サービス**」リストから「**ファイル・アダプタ**」サービスを、プロセス・ダイアグラムの右側の「パートナ・リンク」スイムレーンにドラッグ・アンド・ドロップします。アダプタ構成ウィザードが表示されます。
2. 「**次へ**」をクリックします。「サービス名」ダイアログ・ボックスが表示されます。
3. ファイル・アダプタ・サービス名を入力します(例: WriteAck)。
4. 「**次へ**」をクリックすると、「アダプタ・インタフェース」ダイアログ・ボックスが表示されます。
「**操作およびスキーマから定義(後で指定)**」ラジオ・ボタンを選択して、「**次へ**」をクリックします。「**操作**」ダイアログ・ボックスが表示されます。
5. 「**Write File**」などの操作タイプを指定します。これにより、「**操作名**」フィールドが自動的に設定されます。
6. 「**次へ**」をクリックして「ファイル構成」ダイアログ・ボックスにアクセスします。

出力ファイルの構成

The screenshot shows the 'FILE Adapter Configuration Wizard - Step 5 of 7' window. The title bar includes a close button (X). The main window has a header 'File Configuration' and a sub-header 'Specify the parameters for the Write File operation.' Below this, there are two radio buttons: 'Physical Path' and 'Logical Name'. The 'Logical Name' option is selected. A text field labeled 'Directory for Outgoing Files (logical name):' contains the text 'outputDir'. Below this is another text field labeled 'File Naming Convention (po_%SEQ%.txt):' containing the text 'PO_%yyMMddHHmmss%.xml'. There is a checkbox labeled 'Append to existing file' which is unchecked. Below this is a section titled 'Write to output file when any of these conditions are met:'. It contains three rows of conditions: 'Number of Messages Equals:' with a value of '1', 'Elapsed Time Exceeds:' with a value of '1' and a unit dropdown set to 'minutes', and 'File Size Exceeds:' with a value of '1000' and a unit dropdown set to 'kilobytes'. At the bottom of the window are four buttons: 'Help', '< Back', 'Next >', and 'Cancel'.

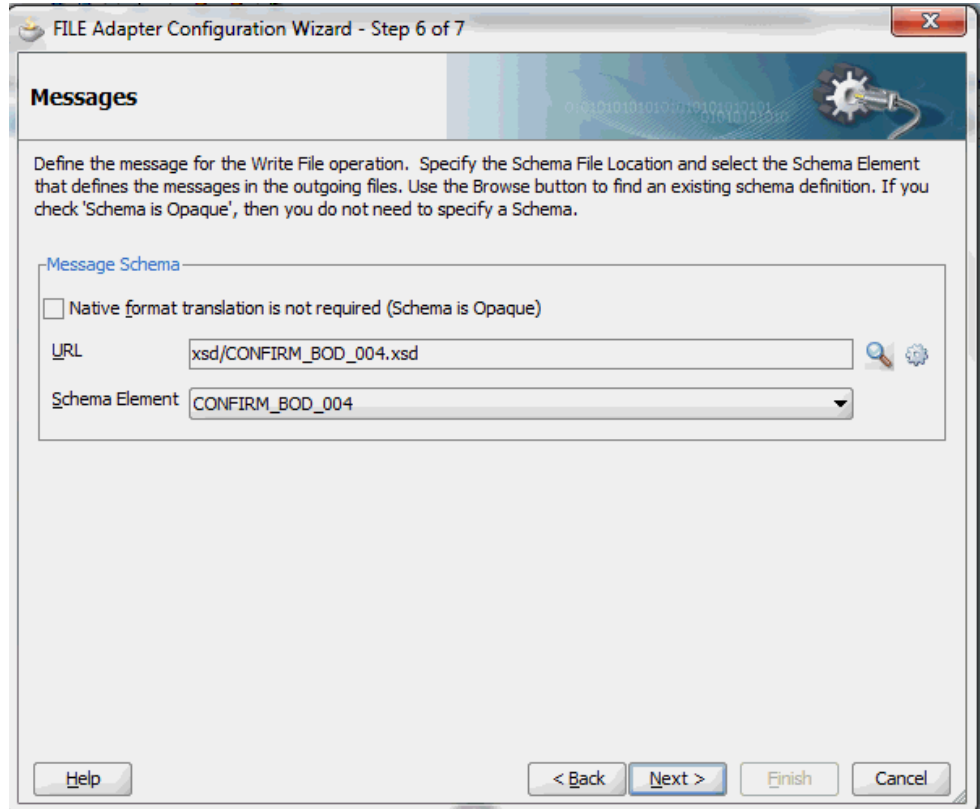
7. 「ディレクトリは次のように指定します」フィールドで、**論理名**を選択します。「発信ファイルのディレクトリ(論理名)」フィールドにディレクトリ・パスを入力し、出力ファイルのネーミング規則を指定します(例: PO_%yyMMddHHmmss%.xml)。
8. デフォルトの書込み条件(「メッセージ数の到達」= 1)を確認します。「次へ」をクリックします。「メッセージ」ダイアログ・ボックスが表示されます。
9. 「参照」チェック・ボックスを選択して、スキーマの場所とスキーマ要素を探します。
「タイプ・チューザ」ダイアログ・ボックスが表示されます。「プロジェクトのスキーマ・ファイル」→「CONFIRM_BOD_004.xsd」を展開し、「CONFIRM_BOD_004」を選択します。

タイプ・チューザ



「タイプ・チューザ」ウィンドウで「OK」をクリックして、選択したスキーマの場所とスキーマ要素を「メッセージ」ダイアログ・ボックスに移入します。

選択したメッセージ・スキーマの表示



10. 「次へ」→「終了」をクリックします。ウィザードによって、パートナー・リンクに対応する WSDL ファイルが生成されます。メインの「パートナー・リンクの作成」ダイアログ・ボックスが表示され、新しい WSDL ファイル WriteAck.wsdl が指定されます。

「適用」→「OK」をクリックして構成を完了し、ファイル・アダプタ・サービスに必要な WSDL 設定を指定してパートナー・リンクを作成します。

WriteAck パートナ・リンクが BPEL プロセス・ダイアグラムに表示されます。

Invoke アクティビティの追加

このステップでは、**受入**アクティビティから受け取った発注確認を WriteAck パートナ・リンクに XML ファイルで送信するための **Invoke** アクティビティを構成します。

Invoke アクティビティを追加するには:

1. Oracle JDeveloper で、コンポーネント・パレットから「**BPEL コンストラクト**」を展開します。**Invoke** アクティビティをプロセス・ダイアグラムの中央のスイムレーン(**受入**アクティビティの後)にドラッグ・アンド・ドロップします。

2. **Invoke** アクティビティを WriteAck サービスにリンクします。**Invoke** アクティビティでは、イベント・データをパートナー・リンクに送信します。「編集 Invoke」ダイアログ・ボックスが表示されます。
3. **Invoke** アクティビティの名前を入力し、「入力変数」フィールドの横にある「作成」アイコンをクリックして、新しい変数を作成します。「変数の作成」ダイアログ・ボックスが表示されます。
4. 「グローバル変数」を選択して、変数の名前を入力します。デフォルト名を受け入れることもできます。「OK」をクリックします。

Invoke アクティビティの表示

The screenshot shows the 'Edit Invoke' dialog box with the following details:

- General Tab:**
 - Name:** Invoke
 - Conversation ID:** (empty)
 - Detail Label:** (empty)
 - ☐ Invoke as Detail
 - Interaction Type:** Partner Link
 - Partner Role Web Service Interface:**
 - Partner Link:** WriteAck
 - Operation:** Write
 - Variables:**
 - Input:** Invoke_Write_InputVariable
 - Output:** (empty)

「編集 Invoke」ダイアログ・ボックスで「適用」→「OK」をクリックして、**Invoke** アクティビティの構成を終了します。

プロセス・ダイアグラムに **Invoke** アクティビティが表示されます。

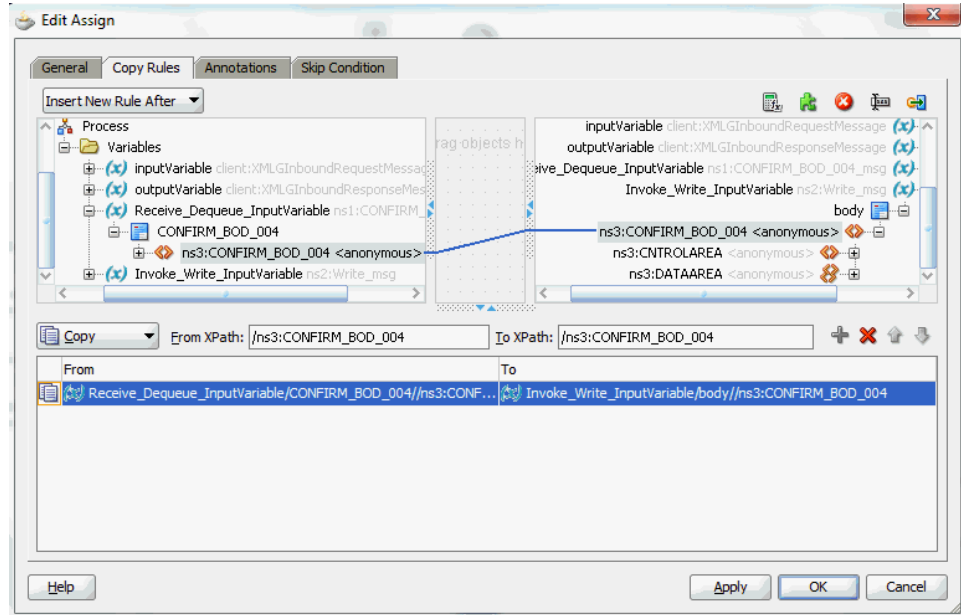
Assign アクティビティの追加

このステップを使用して、**受入**アクティビティからの発注確認の詳細を **Invoke** アクティビティに渡します。

Assign アクティビティを追加するには:

1. Oracle JDeveloper で、コンポーネント・パレットから「**BPEL コンストラクト**」を展開します。
Assign アクティビティをコンポーネント・パレットからプロセス・ダイアグラムの中央のスイムレーン(受入アクティビティと **Invoke** アクティビティの間)にドラッグ・アンド・ドロップします。
2. **Assign** アクティビティをダブルクリックして、「編集 Assign」ダイアログ・ボックスにアクセスします。
3. 「コピー・ルール」タブを選択して、ソース・ツリーとターゲット・ツリーを展開します。
 - 「元」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「**Receive_DEQUEUE_InputVariable**」に移動し、「**CONFIRM_BOD_004**」を選択します。選択したエントリが XPath フィールドに入力されます。
 - 「先」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「**Invoke_Write_InputVariable**」→「**body**」に移動し、「**CONFIRM_BOD_004**」を選択します。選択したエントリが XPath フィールドに入力されます。

パラメータの割当て



ソース・ノード(CONFIRM_BOD_004)をドラッグして、先ほど識別したターゲット・ノード(CONFIRM_BOD_004)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

4. 「編集 Assign」ダイアログ・ボックスで「適用」→「OK」をクリックして、Assign アクティビティの構成を完了します。

実行時における BPEL プロセスによる SOA コンポジット・アプリケーションのデプロイおよびテスト

XML ゲートウェイ・アウトバウンド・メッセージ・マップのための BPEL プロセスによる SOA コンポジット・アプリケーションを作成したら、Oracle WebLogic 管理対象サーバーにデプロイする必要があります。これは、Oracle JDeveloper を使用して達成できます。コンポジットがデプロイされたら、Oracle Enterprise Manager Fusion Middleware Control コンソールからテストできます。

前提条件

Oracle JDeveloper を使用して BPEL プロセスによる SOA コンポジットをデプロイする前に、設計時環境とランタイム・サーバーの間の接続を確立する必要があります。必要なサーバー接続を構成する方法の詳細は、「サーバー接続の構成」(ページ B-1)を参照してください。

注意: WebLogic サーバーのローカル・インスタンスが使用される場合、Oracle JDeveloper から「実行」→「サーバー・インスタンスの起動」を選択して、WebLogic サーバーを起動します。WebLogic 管理サーバーの DefaultServer インスタンスが

正常に起動されると、サーバーが実行モードで起動されたというメッセージと DefaultServer が起動されたというメッセージが「実行中:DefaultServer」ログと「メッセージ」ログに表示されます。

次の実行時タスクを実行します。

1. BPEL プロセスによる SOA コンポジットのデプロイ、ページ 5-55
2. SOA コンポジット・アプリケーションの手動テスト、ページ 5-58

BPEL プロセスによる SOA コンポジット・アプリケーションのデプロイ

これまでに作成した BPEL プロセス(XMLGOutbound.bpel)による SOA コンポジット・アプリケーションを実行する前に、デプロイする必要があります。

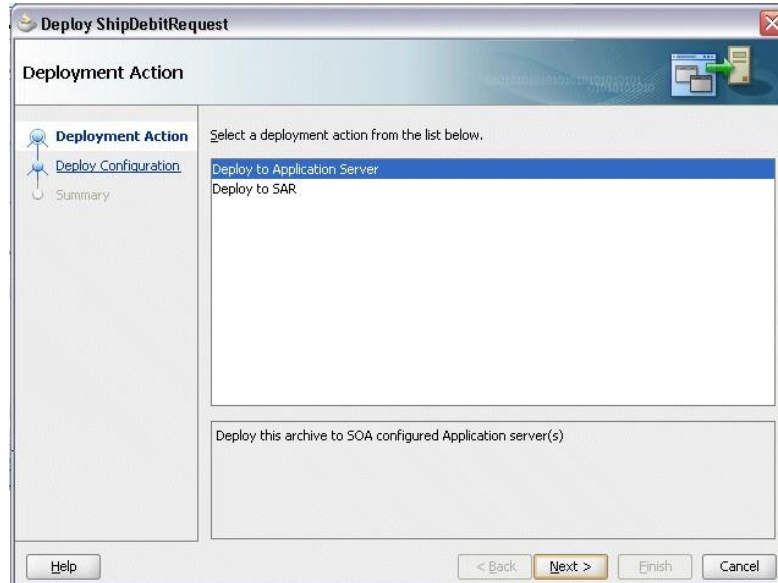
SOA コンポジット・アプリケーション BPEL プロセスをデプロイするには:

1. JDeveloper のアプリケーション・ナビゲータで、**XMLGOutbound** プロジェクトを選択します。
2. プロジェクトを右クリックして、メニューから「**配置**」→[project name]→[serverConnection]を選択します。

たとえば、接続が適切に設定されている場合は、「**デプロイ**」→「**XMLGInbound**」→「**SOAServer**」を選択してプロセスをデプロイできます。

注意: サーバー接続を初めて設定する場合は、「**デプロイメント・アクション**」ダイアログが表示されます。「**アプリケーション・サーバーにデプロイ**」を選択して、「**次へ**」をクリックします。

「デプロイメント・アクション」ダイアログ



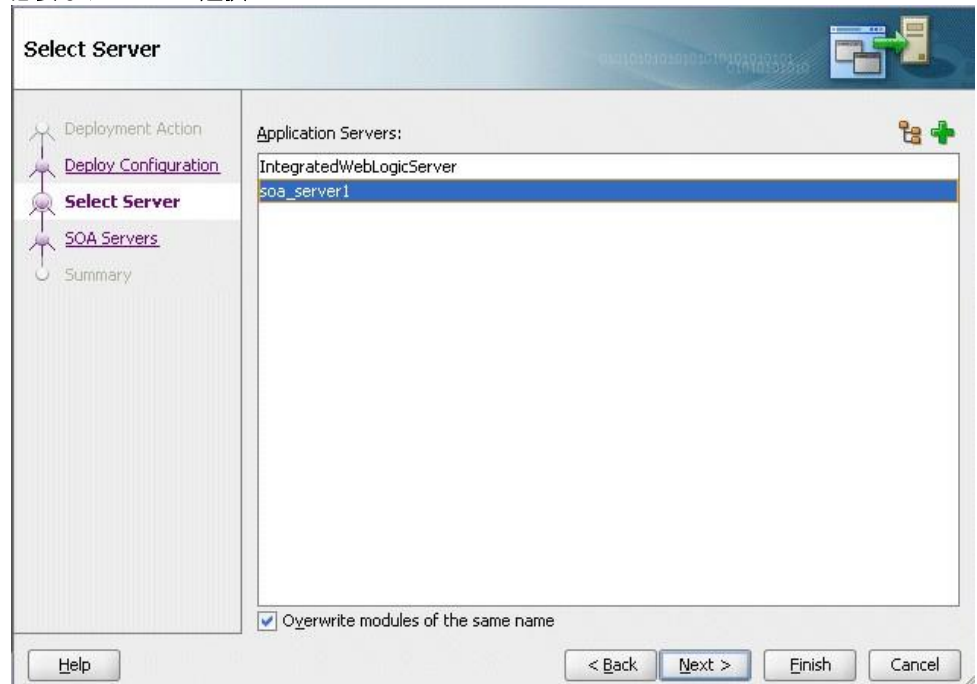
「構成のデプロイ」ダイアログで、「次へ」をクリックして新しいアプリケーション・サーバーを追加する前に、次の情報が選択されるようにします。

- 新規リビジョン ID: 1.0
- コンポジット・リビジョンをデフォルトとしてマークします。: このチェック・ボックスを選択します。
- 同じリビジョン ID で既存のコンポジットを上書きします。: このチェック・ボックスを選択します。

JDeveloper から新しい Oracle WebLogic Server 接続を作成するステップは、「サーバー接続の構成」(ページ B- 1)を参照してください。

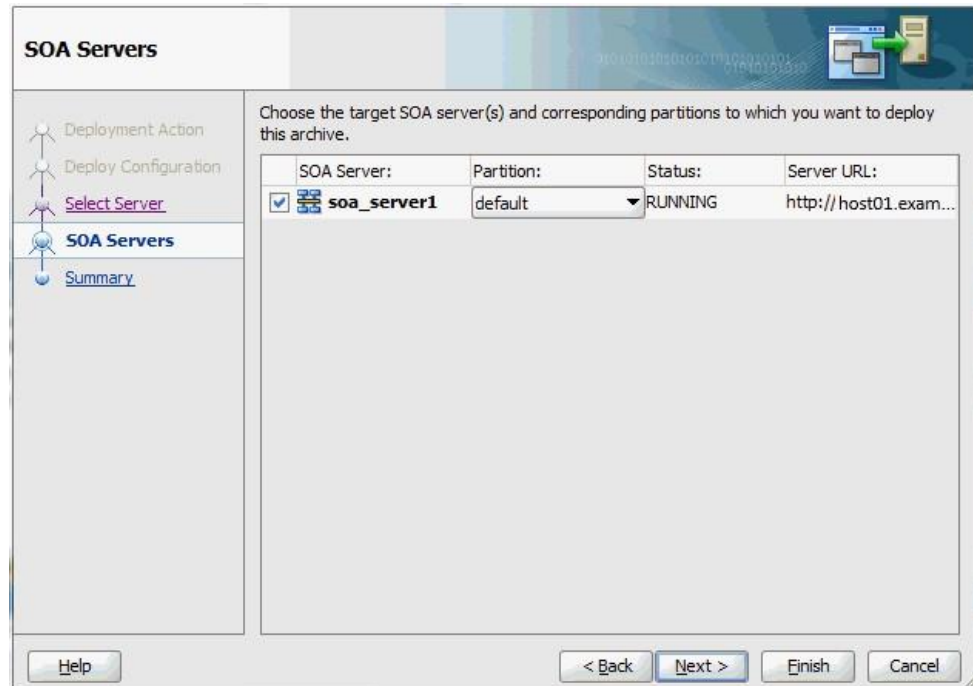
3. 「サーバーの選択」ダイアログで、以前にサーバー接続を確立した soa-server1 を選択します。「次へ」をクリックします。

必要なサーバーの選択



4. 「SOA サーバー」ダイアログで、デフォルトのターゲット SOA サーバー(soa-server1)の選択を受け入れます。

必要な SOA サーバーの選択



「次へ」→「終了」をクリックします。

5. Oracle JDeveloper セッションから初めてコンポジットをデプロイする場合は、「認証リクエスト」ウィンドウが表示されます。Oracle SOA Suite のインストール時に指定したユーザー名とパスワード情報を入力します。「OK」をクリックします。
6. デプロイ処理が開始します。SOA - ログ・ウィンドウで、デプロイ・プロセスをモニターして、コンパイルが成功したことを確認します。

デプロイメント - ログ・ウィンドウで、デプロイが成功したことを確認します。

BPEL プロセスによる SOA コンポジット・アプリケーションのテスト

SOA コンポジット・アプリケーションに含まれる BPEL プロセスが正常にデプロイされたら、Oracle Enterprise Manager Fusion Middleware Control コンソールからプロセスを管理およびモニターできます。Oracle E-Business Suite にログインして、手動で受注を作成して記帳するとともに、ワークフロー・バックグラウンド・プロセス・コンカレント要求を発行して発注確認を生成できます。

Oracle Enterprise Manager Fusion Middleware Control コンソールにログインして、XML ゲートウェイの ECX_OUTBOUND キューから発注確認を受け取った後に出力ディレクトリに書き込む BPEL プロセスを検証します。

BPEL プロセスによる SOA コンポジット・アプリケーションを手動でテストするには:

1. 「XML ゲートウェイ」職責を持つユーザーとして Oracle E-Business Suite にログインします。

ここでは、定義済の有効な顧客が発注に含まれるように XML ゲートウェイの取引先が正しく設定されていることを確認します。

2. ナビゲーション・メニューから「**取引先の定義**」を選択して、「取引先設定」ウィンドウにアクセスします。

3. 「取引先設定」フォームに次のようにヘッダー値を入力します。

- 取引先タイプ: 顧客
- 取引先名: Hilman and Associates など
- 取引先サイト: 取引先サイトの情報を入力します(例: 1230 East 60th Street South Tulsa, OK 74146)。
- 会社管理者 E メール: 有効な E メール・アドレスを入力します。

4. 次の取引先詳細を入力します。

- トランザクション・タイプ: ECX
- トランザクション・サブタイプ: CBODO
- 標準コード: OAG
- 外部トランザクション・タイプ: BOD
- 外部トランザクション サブタイプ: CONFIRM
- 方向: OUT
- マップ: ECX_CBODO_OAG72_OUT
- 接続/ ハブ: DIRECT
- プロトコル・タイプ: BPEL
- ユーザー名: operations
- パスワード: ユーザー「operations」に関連付けられているパスワードを 2 回入力します。
- プロトコル・アドレス: <http://us.example.com>
プロトコル・アドレスは、XML 文書を転記できる完全な URL です。
- ソース取引先ロケーション・コード: HIL-Tulsa

取引先設定

Transaction Type	Transaction SubType	Standard Code	External Transaction Type	External Transaction SubType	Direction	Map	Connection/Hub	Protocol Type
ONT	POA	OAG	PO	ACKNOWLEDGE	OUT	ONT_3A4A_O	DIRECT	SMTP
ONT	CPO	OAG	PO	CANCEL	IN	ONT_3A9R_O		
ONT	POI	OAG	PO	PROCESS	IN	ONT_3A4R_O		
ONT	SSO	OAG	SALESORDER	SHOW	OUT	PROCESS_P	DIRECT	SMTP
ECX	CBODO	OAG	BOD	CONFIRM	OUT	ECX_CBODO	DIRECT	BPEL

5. 取引先詳細を保存します。

発注確認を正常に生成するには、Order Management で次の設定タスクを実行し、受注を手動で記帳します。

1. 職責を「受注管理スーパーユーザー, Vision オペレーション (USA)」に切り替え、ナビゲーション・メニューから「顧客」→「標準」を選択して「顧客の入力」フォームを開きます。
2. 「名前」フィールドで Hilman and Associates を検索し、「進む」をクリックします。
3. 検索結果から、次の情報を持つ Business World を選択します。
 - 名前: Hilman and Associates
 - レジストリ ID: 1004
 - 所在地: 1230 East 60th Street South, Tulsa, OK 74146
4. 次の情報を持つアカウントを選択します。
 - アカウント番号: 1004

- アカウント摘要: Hilman and Associates
 - ステータス: 有効
5. 「詳細」アイコンをクリックして、「アカウントの更新: 1608」ページを開きます。
 6. 1230 East 60th Street South, Tulsa, OK 74146 という所在地を探し、「詳細」アイコンをクリックして、選択したアカウントの「サイト」ページを開きます。
 7. 「サイト詳細」タブの「アカウント・サイト詳細」リージョンで、「EDI 事業所」フィールドに HIL-Tulsa と入力します。
 8. 「ビジネス目的」タブで、次の値を指定して新しい行を作成します。
 - 目的: 販売先
 - 「主」チェック・ボックスを選択「適用」をクリックします。

次のステップを使用して、XML ゲートウェイ・インバウンド・サービスについて作成した受注の確認を生成します。

1. 同じ「受注管理スーパーユーザー, Vision オペレーション (USA)」職責を持つユーザーとして Oracle E-Business Suite にログインします。「受注返品」→「受注」を選択して、「受注」フォームを開きます。
2. 「顧客発注」フィールドに発注 ID「PO-4466-5」を入力して、XML ゲートウェイ・インバウンド・サービスで前に作成した受注を取得します。
3. 「受注の記帳」をクリックして受注を記帳します。
「ステータス」フィールドが「記帳済」に変わります。

Sales Orders (Vision Operations) - 65182, Hilman and Associates

Order Information Line Items

Default

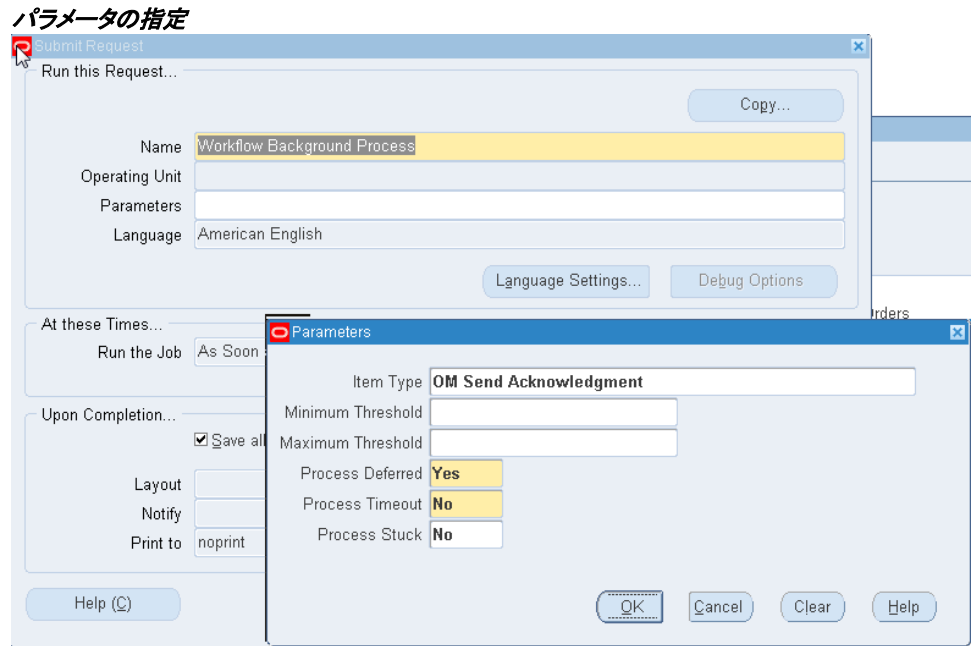
Main Others

Customer	Hilman and Associates	Order Number	65182
Customer Number	1004	Order Type	Mixed
Customer PO	PO-4466-5	Date Ordered	04-SEP-2012 23:26:29
Customer Contact	Cowlshaw, Marcus Mr.	Price List	Corporate
Ship To Location	Tulsa (OPS)	Salesperson	Cleary, Mr. Timothy
	1230 East 60th Street South	Status	Entered
		Currency	USD
	Tulsa, OK, 74146, US	Subtotal	26,156.40
Bill To Location	Tulsa (OPS)	Tax	0.00
	1230 East 60th Street South	Charges	150.00
		Total	26,306.40
	Tulsa, OK, 74146, US		

[] ☐

Actions Related Items Configurator Availability Book Order

4. 「システム管理者」職責に切り替え、「要求」→「実行」を選択します。
5. 「単一要求」を選択し、「OK」をクリックします。
6. 「要求の発行」フォームで次の情報を入力します。



- 名称: ワークフロー・バックグラウンド・プロセス
- 次のパラメータを入力します。
 - 項目タイプ: OM 通知の送信
 - 処理繰延: Yes
 - 処理タイムアウト: No
 - 処理スタック: No
- 「OK」をクリックします。

7. 「発行」をクリックして、確認の送信要求を発行します。

8. 要求 ID を入力して要求を表示し、ステータスが「成功」であることを確認します。

Oracle Enterprise Manager Fusion Middleware Control コンソールを使用した検証

Oracle Enterprise Manager Fusion Middleware Control コンソール

(<http://<servername>:<portname>/em>)にログインして、XMLGOutbound プロセスがデプロイされていることを確認します。このプロセスは、継続的に ECX_OUTBOUND キューで発注確認をポーリングしています。

「ファーム」ナビゲーション・ペインのツリーで「SOA」→「soa-infra」ノードを展開し、SOA インフラストラクチャのホーム・ページおよびメニューを介して、soa-infra 管理対象サーバーで実行されているデプロイ済の SOA コンポジット・アプリケーションにアクセスします。XMLGOutbound [1.0]のリンクをクリックします。

「テスト」をクリックします。インスタンスを開始するための「Web サービスのテスト」ページが表示されます。プロセスに必要な入力文字列を入力し、「Web サービスのテスト」をクリックしてプロセスを開始します。完了時に、「レスポンス」タブにテスト結果が表示されます。BPEL サービス・コンポーネント・インスタンスのリンク(例: XMLGOutbound)をクリックして、「インスタンス」ページを表示します。ここでは、「監査証跡」タブで BPEL アクティビティの実行の詳細を参照できます。

「フロー」タブをクリックして、BPEL プロセスのフロー・ダイアグラムを確認します。プロセス・ダイアグラムのアクティビティをクリックして、アクティビティの詳細と、プロセスを介したペイロードのフローを表示します。たとえば、**受入**アクティビティをクリックします。「XML 文書の表示」リンクをクリックして、受け入れられた XML ファイルを開きます。「顧客発注」などの参照 ID を書き留めます。

書込み操作について指定したディレクトリ(例: outputDir)に移動します。これは、発注確認を XML ファイル(例: PO_060719175318.xml)に書き込んだ後にファイル・アダプタがファイルを配置した論理的な場所(通常は c:\temp)です。

PO_060719175318.xml ファイルを開き、ORIGREF 要素を探します。その値は、確認が生成された PO-4466-5 (記帳済の受注)になっています。

サブスクリプション・モデルを介したビジネス・イベントの使用

概要

Oracle Workflow ビジネス・イベント・システム(BES)は、Oracle Advanced Queuing (AQ)インフラストラクチャを活用して、システム間でビジネス・イベントをやり取りするアプリケーション・サービスです。ビジネス・イベント・システムは、イベント・マネージャとワークフロー・プロセス・イベント・アクティビティで構成されます。

イベント・マネージャを使用すると、重要なイベントに対するサブスクリプションを登録できます。ワークフロー・プロセス内のビジネス・イベントを表すイベント・アクティビティを使用することで、ワークフロー・プロセス内の複雑なビジネス・フローやロジックをモデル化できます。

イベントは、ローカルに呼び出されることもあれば、AQ を介して外部システムまたはローカル・システムから受信することもあります。ローカル・イベントが発生すると、サブスクリプションが遅延されている場合を除き、イベントを呼び出したコードと同じトランザクションで購読側のコードが実行されます。

Oracle E-Business Suite Integrated SOA Gateway では、イベント・サブスクリプションを介してビジネス・イベントがサポートされます。統合管理者は、ビジネス・イベントのインタフェースの詳細ページからビジネス・イベントを購読できます。そのイベントのサブスクリプションを送信エージェントとしてエンキューできます。統合開発者は、Oracle JDeveloper で BPEL プロセスによる SOA コンポジット・アプリケーションを作成して、設計時に購読対象のイベントを組み込み、実行時に必要に応じてアプリケーション・データを更新できます。

サブスクリプション・モデルがビジネス・イベントに対してどのように機能するかについて理解を深めるために、この章では設計時と実行時の詳細なタスクを説明します。以降の項で説明する例では、BPEL プロセスによる SOA コンポジット・アプリケーションを作成するための設計時ツールとして Oracle JDeveloper 11g (11.1.1.6.0)を使用し、プロセスの配置には Oracle SOA Suite 11g (11.1.1.6.0)を使用します。

注意: Oracle JDeveloper を他の Oracle Fusion Middleware コンポーネント(Oracle SOA Suite など)とともに使用している場合、SOA テクノロジを有効にするには、SOA テクノロジ用の JDeveloper の拡張機能である Oracle SOA Suite コンポジット

ト・エディタを手動でダウンロードする必要があります。追加の Oracle Fusion Middleware 設計時コンポーネントのインストールの詳細は、*Oracle Fusion Middleware Oracle JDeveloper インストレーション・ガイド*を参照してください。

設計時に BPEL プロセスによる SOA コンポジット・アプリケーションを作成する場合のビジネス・イベントの使用

BPEL プロセスによる SOA コンポジット・アプリケーションのシナリオ

BPEL プロセスによる SOA コンポジット・アプリケーションの作成を説明するための例として、発注 XML のビジネス・イベントの呼出しを使用します。

発注が作成されて承認されると、発注承認済ビジネス・イベント

`oracle.apps.po.event.xmlpo` が呼び出されます。このイベントのサブスクリプションは、インタフェースの詳細ページで作成され(内部的には、WF_BPEL_QAGENT を送信エージェントとする選択済のイベントについて、イベント・サブスクリプションが自動的に作成されます)、WF_EVENT_T 構造でアドバンスト・キューWF_BPEL_Q にエンキューされるため、最初に WF_BPEL_Q キューからサブスクリプションをデキューしてイベント詳細を取得する BPEL プロセスを作成します。イベント詳細は、BPEL プロセス・アクティビティを介して渡され、出力ファイルとして XML ファイルに書き込まれます。

デプロイメント後、BPEL プロセスが正常に実行されると、発注が承認されたときに、同じ発注情報が出力ファイルから取得されます。

ビジネス・イベントを使用して BPEL プロセスによる SOA コンポジット・アプリケーションを作成するための前提条件

統合管理者は、まず、Oracle Integration Repository のユーザー・インタフェースからビジネス・イベントを購読する必要があります。内部的には、WF_BPEL_QAGENT を送信エージェントとするそのイベントについて、イベント・サブスクリプションが自動的に作成されます。

たとえば、ビジネス・イベント `oracle.apps.po.event.xmlpo` を購読する必要があります。イベント・サブスクリプションが正常に作成された場合、確認メッセージが表示されます。

ビジネス・イベントを購読するには、管理者は、まず、Oracle Integration Repository からイベントを探し、その後、インタフェースの詳細ページで「購読」をクリックしてサブスクリプションを作成します。

ビジネス・イベントを購読する方法の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「ビジネス・イベントに対する予約」を参照してください。

注意: 購読したビジネス・イベントを使用して BPEL プロセスを作成する場合、購読対象のビジネス・イベントが正常に WF_BPEL_Q キューにエンキューされるためには、次のことを確認する必要があります。

- コンシューマ名が一意です。
- ビジネス・イベントを呼び出す前に BPEL プロセスがデプロイされています。

サブスクリプションが作成されてエンキューされると、統合開発者は、設計時に Oracle JDeveloper で BPEL 言語を使用して、有用なビジネス・プロセスに購読対象のイベントを組み込むことができます。

BPEL プロセスによる SOA コンポジット・アプリケーションの作成フロー

発注 XML のビジネス・イベントの呼出しのシナリオに基づいて、この章では次の設計時タスクについて説明します。

1. 新しい BPEL プロセスによる SOA コンポジット・アプリケーションの作成、ページ 6-3
このステップを使用して、GetPurchaseOrder.bpel という、新しい BPEL プロセスによる SOA コンポジット・アプリケーションを作成します。
2. AQ アダプタのパートナー・リンクの作成、ページ 6-5
このステップを使用して、WF_BPEL_Q キューからイベント詳細をデキューします。
3. 受入アクティビティの追加、ページ 6-11
受入アクティビティを使用して、イベント詳細を Assign アクティビティへの入力として取得します。
4. ファイル・アダプタのパートナー・リンクの作成、ページ 6-13
ここでは、イベント詳細を出力ファイルとして XML ファイルに書き込みます。
5. Invoke アクティビティの追加、ページ 6-19
ここでは、ファイル・アダプタのパートナー・リンクを起動することにより、ビジネス・イベント情報を XML ファイルに書き込みます。
6. Assign アクティビティの追加、ページ 6-21
受入アクティビティから出力を取得し、Invoke アクティビティに入力を渡すには、Assign アクティビティを使用します。

BPEL プロセスのサービス・コンポーネントを使用した SOA コンポジット・アプリケーションの一般情報と作成方法の詳細は、*Oracle Fusion Middleware Oracle SOA Suite 開発者ガイド* を参照してください。

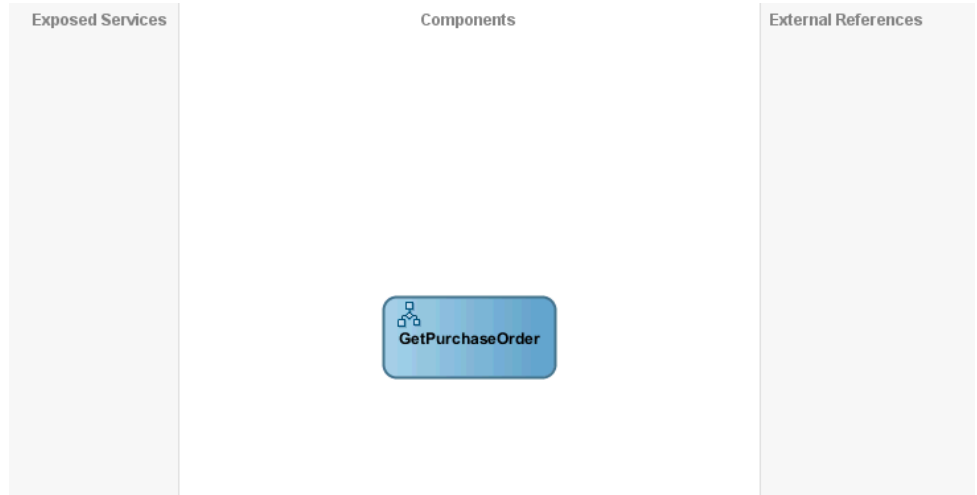
新しい BPEL プロセスによる SOA コンポジット・アプリケーションの作成

このステップを使用して、様々な BPEL プロセス・アクティビティを含む、新しい BPEL プロセスによる SOA コンポジット・アプリケーションを作成します。

新しい BPEL プロセスによる SOA コンポジット・アプリケーションを作成するには:

1. Oracle JDeveloper を開きます。
2. アプリケーション・ナビゲータで「**新規アプリケーション**」をクリックします。
「SOA アプリケーションの作成 - アプリケーションの名前付け」ページが表示されます。
3. 「アプリケーション名」フィールドにアプリケーションの適切な名前を入力して、「アプリケーション・テンプレート」リストから「**SOA アプリケーション**」を選択します。
「**次へ**」をクリックします。「SOA アプリケーションの作成 - プロジェクトの名前付け」ページが表示されます。
4. 「プロジェクト名」フィールドにプロジェクトの適切な名前を入力します(例: GetPOProject)。
5. 「プロジェクト・テクノロジー」タブで、「**Web サービス**」を選択して、「使用可能」のテクノロジー・リストから「**選択済**」のテクノロジー・リストに「**SOA**」が選択されるようにします。
「**次へ**」をクリックします。「SOA アプリケーションの作成 - SOA 設定の構成」ページが表示されます。
6. 「コンポジット・テンプレート」リストから**コンポジット BPEL プロセス**を選択して、「**終了**」をクリックします。新規アプリケーションと SOA プロジェクトを作成しました。これにより、SOA コンポジットが自動的に作成されます。
「BPEL プロセスの作成」ページが表示されます。
7. デフォルトの「**BPEL 1.1 Specification**」の選択を変更せずに維持します。これにより、BPEL 1.1 Specification をサポートする BPEL プロジェクトが作成されます。
「名前」フィールドに BPEL プロセスの適切な名前を入力します(例: GetPurchaseOrder)。
「テンプレート」フィールドで「**サービスを後で定義**」を選択します。「**OK**」をクリックします。
空の BPEL プロセスが作成されます。また、指定した名前(例: GetPurchaseOrder.bpel および GetPurchaseOrder.wsdl)を使用した bpel および wsdl を含む必要なソース・ファイルと、composite.xml も生成されます。
8. 「**終了**」をクリックします。
9. 「SOA コンテンツ」→「ビジネス・ルール」にナビゲートし、composite.xml をダブルクリックしてコンポジット・ダイアグラムを表示します。

コンポジット・ダイアグラムの表示



GetPurchaseOrder コンポーネントをダブルクリックして BPEL プロセスを開きます。

AQ アダプタのパートナ・リンクの作成

このステップを使用して、`oracle.apps.po.event.xmlpo` イベントのサブスクリプションをデキューするための AQ アダプタのパートナ・リンク(GetPurchaseOrder)を作成します。
イベント・サブスクリプションをデキューするための AQ アダプタのパートナ・リンクを作成するには:

1. Oracle JDeveloper で、「**BPEL サービス**」リストにある「**AQ アダプタ**」サービスをプロセス・ダイアグラムの右側の「パートナ・リンク」スイムレーンにドラッグ・アンド・ドロップします。ファイル・アダプタ構成ウィザードへようこそページが表示されます。
2. 「次へ」をクリックします。「サービス名」ダイアログ・ボックスが表示されます。
3. 「サービス名」ダイアログ・ボックスでサービス名を入力します(例: GetPO)。
4. 「次へ」をクリックします。「サービス接続」ダイアログ・ボックスが表示されます。
5. 既存のデータベース接続を使用する場合は、「**接続**」リストからデータベース接続を選択し、新規データベース接続を定義する場合は、「**新規**」をクリックしてデータベース接続作成ウィザードを開きます。

注意: Oracle E-Business Suite が実行されているデータベースに接続する必要があります。

新規データベース接続を作成するには:

1. 「**新規**」をクリックして、データベース接続の作成ウィザードを開きます。「次へ」をクリックし、一意の接続名を入力して、データベース接続の接続タイプを選択します。

- 「次へ」をクリックします。
2. 「認証」ダイアログ・ボックスで、適切なユーザー名およびパスワード情報を入力してデータベース接続を認証します。「次へ」をクリックします
 3. 「接続」ダイアログ・ボックスで、次の情報を指定します。
 - ドライバ: Thin
 - ホスト名: データベース接続のホスト名を入力します。たとえば、`myhost01.example.com` のようにします。
 - JDBC ポート: データベース接続の JDBC ポート番号(1521 など)を入力します。
 - SID: データベース接続の一意の SID (sid01 など)を指定します。
 4. 「次へ」をクリックしてデータベース接続をテストします。
ステータス・メッセージ「成功しました。」は、接続が有効であることを示します。
 5. 「次へ」をクリックして、データベース接続のサマリーを示す「サービス接続」ダイアログ・ボックスに戻ります。
 6. 「サービス接続」ダイアログ・ボックスの「**JNDI 名**」フィールドには、指定したデータベース接続に対応する JNDI (Java Naming and Directory Interface)名が自動的に表示されます。かわりに、別の JNDI 名を入力することもできます。
 7. 「次へ」をクリックして「アダプタ・インタフェース」ダイアログ・ボックスを開きます。
「操作およびスキーマから定義(後で指定)」ラジオ・ボタンを選択して、「次へ」をクリックします。「操作」ダイアログ・ボックスが表示されます。
 8. 「操作タイプ」フィールドで「デキュー」ラジオ・ボタンを選択します。「操作名」フィールドにも「デキュー」が移入されます。

操作の指定

Adapter Configuration Wizard - Step 5 of 7

Operation

The AQ Adapter supports three operations. There is a Dequeue operation that polls for incoming messages from a queue, an Enqueue operation that puts outgoing messages on a queue, and an Enqueue/Dequeue operation that puts outgoing messages on a queue and expects response messages on a queue. Specify the Operation type and Operation Name. Only one operation per Adapter Service may be defined using this wizard.

Operation Type: ☒ Dequeue ☐ Enqueue ☐ Enqueue/Dequeue

Operation Name:

Help < Back Next > Finish Cancel

9. 「次へ」をクリックして「キュー名」ダイアログ・ボックスを開きます。

「データベース・スキーマ」フィールドで「APPS」を選択します。「キュー名」フィールドにWF_BPEL_Qと入力します。

データベース・スキーマとキュー名の指定

AQ Adapter Configuration Wizard - Step 6 of 9

Queue Name

Specify the database schema and the queue to be used for the service. Use the Browse button to find the queue in the specified schema.

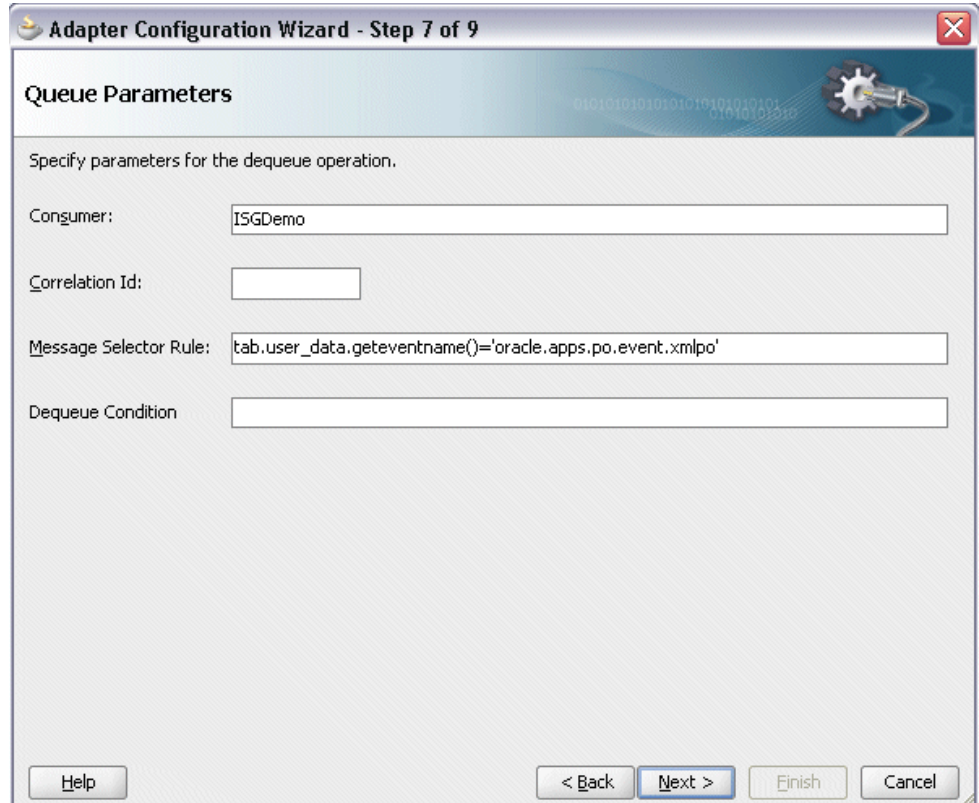
Queue Information

Database Schema:

Queue Name:

10. 「次へ」をクリックして「キュー・パラメータ」ダイアログ・ボックスを開きます。

キュー・パラメータの指定



次の情報を入力します。

- 一意のコンシューマ名を入力します。

重要: 購読対象のビジネス・イベントが正常に WF_BPEL_Q キューにエンキューされるためには、コンシューマ名が一意である必要があります。

- メッセージ・セクタ・ルールの情報を入力します(例: `tab.user_data.geteventname()='oracle.apps.po.event.xmlpo'`)。

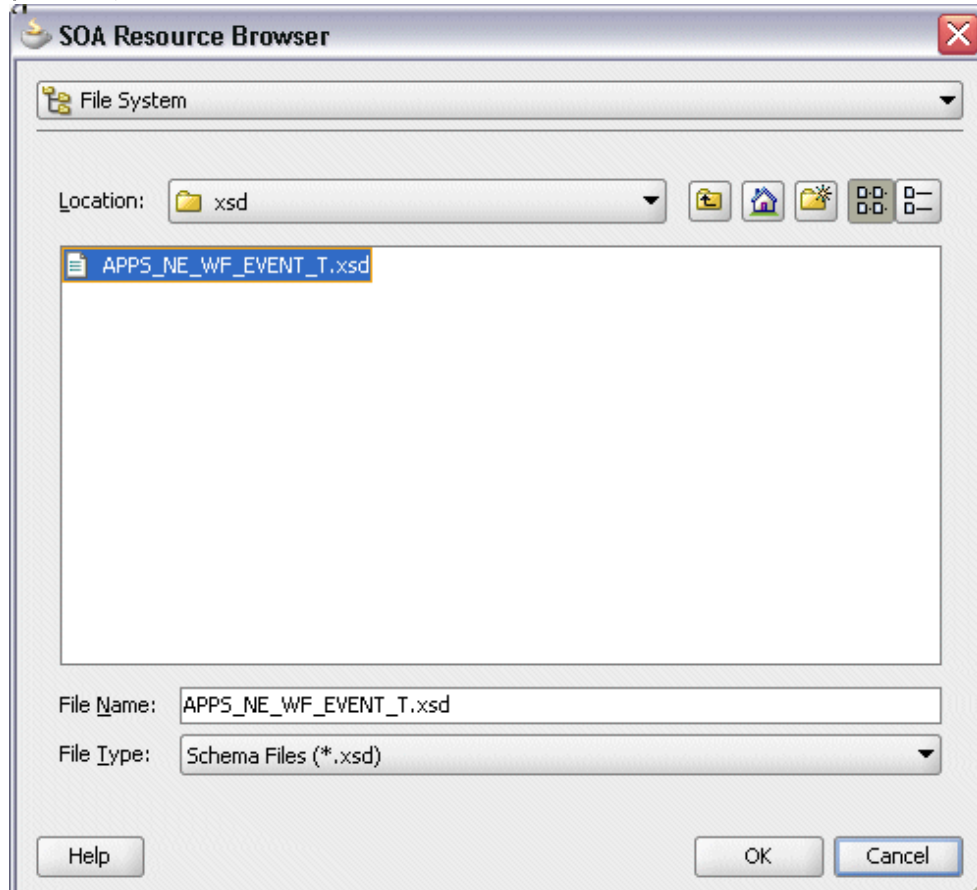
11. 「次へ」をクリックします。「メッセージ」ダイアログ・ボックスが開き、APPS.WF_BPEL_Q キューのビジネス・イベント・システム・ペイロードに含まれるメッセージを定義できます。

「スキーマ・ファイルを参照」をクリックして「タイプ・チューザ」ウィンドウを開きます。

「タイプ・チューザ」ウィンドウの右上隅にある「スキーマ・ファイルのインポート」をクリックします。これにより、「スキーマ・ファイルのインポート」ポップアップ・ウィンドウが開きます。

「リソースの参照」アイコンをクリックして、「SOA リソース・ブラウザ」ウィンドウを表示します。場所として xsd フォルダを選択し、フォルダから APPS_NE_WF_EVENT_T ファイルを選択します。

目的のスキーマのインポート

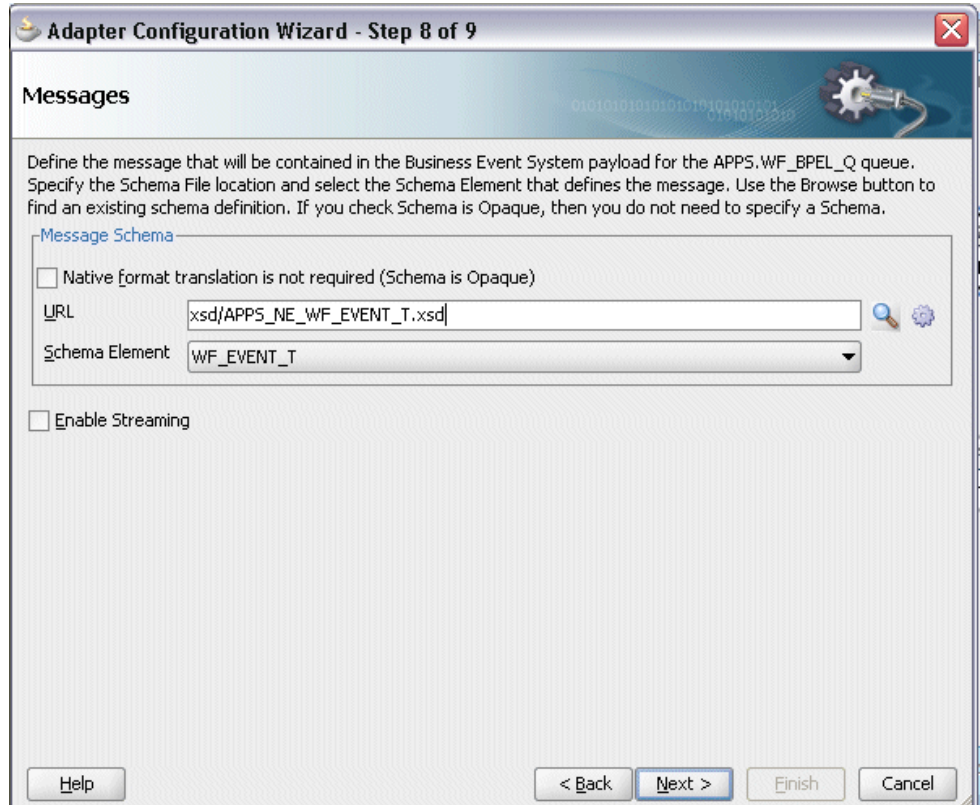


「OK」を 2 回クリックします。

「タイプ・チューザ」ウィンドウで、「プロジェクトのスキーマ・ファイル」を展開し、APPS_NE_WF_EVENT_T.xsd から WF_EVENT_T スキーマを選択します。「OK」をクリックします。

選択した APPS_NE_WF_EVENT_T.xsd が URL として表示され、WF_EVENT_T 要素がスキーマ要素として選択されます。

メッセージ・スキーマの表示



12. 「次へ」をクリックして「終了」ダイアログ・ボックスに進み、GetPO サービスの AQ アダプタの定義が終了したことを確認します。
13. 「終了」をクリックします。GetPO サービスに対応する WSDL ファイルが生成されます。
「適用」→「OK」をクリックして、パートナ・リンクの構成を完了します。パートナ・リンクは、必要な WSDL 設定で作成され、BPEL プロジェクトでは、プロセス・ダイアグラムの枠線の領域の新しいアイコンによって表現されます。

受入アクティビティの追加

このステップでは、ビジネス・イベントの AQ アダプタ・サービスについて構成したパートナ・リンク GetPO から XML データを受け入れるための受入アクティビティを構成します。

受入アクティビティから得た XML データは、次のステップで作成する Assign アクティビティに対する入力変数として使用します。

発注 XML データを取得するための受入アクティビティを追加するには:

1. Oracle JDeveloper で、コンポーネント・パレットから「BPEL コンストラクト」を展開します。

受入アクティビティをプロセス・ダイアグラムの中央のスイムレーンにドラッグ・アンド・ドロップします。

2. 受入アクティビティを GetPO パートナ・リンクにリンクします。受入アクティビティでは、パートナ・リンクからイベント・データを取得します。「Receive の編集」ダイアログ・ボックスが表示されます。
3. 受入アクティビティの名前を入力し(例: Receive_PO)、「変数」フィールドの横にある「作成」アイコンをクリックして新しい変数を作成します。「変数の作成」ダイアログ・ボックスが表示されます。
4. 「グローバル変数」を選択し、変数の名前を入力します。デフォルトの名前をそのまま使用してもかまいません。「OK」をクリックして、「Receive の編集」ダイアログ・ボックスに戻ります。
5. 「適用」→「OK」をクリックして、受入アクティビティの構成を終了します。

受入アクティビティの表示

The screenshot shows the 'Edit Receive' dialog box. The 'General' tab is selected. The 'Name' field contains 'Receive_PO'. The 'Conversation ID' field is empty. The 'Create Instance' checkbox is unchecked. The 'Interaction Type' is set to 'Partner Link'. The 'My Role Web Service Interface' section contains 'Partner Link' (GetPO) and 'Operation' (DEQUEUE). The 'Variable' section contains 'Variable' (Receive_PO_DEQUEUE_InputVariable). The 'OK' button is highlighted.

BPEL プロセス・ダイアグラムに受入アクティビティが表示されます。

ファイル・アダプタのパートナ・リンクの追加

このステップを使用して、イベント・データを XML ファイルに書き込むことにより、ビジネス・イベントを構成します。

ファイル・アダプタのパートナ・リンクを追加するには:

1. Oracle JDeveloper で、「BPEL サービス」リストから「ファイル・アダプタ」サービスを、プロセス・ダイアグラムの右側の「パートナ・リンク」スイムレーンにドラッグ・アンド・ドロップします。アダプタ構成ウィザードが表示されます。
2. 「次へ」をクリックします。「サービス名」ダイアログ・ボックスが表示されます。
3. ファイル・アダプタ・サービス名を入力します(例: WritePurchaseOrder)。
4. 「次へ」をクリックします。「アダプタ・インタフェース」ダイアログ・ボックスが表示されます。
5. 「操作およびスキーマから定義(後で指定)」ラジオ・ボタンを選択して、「次へ」をクリックします。「操作」ダイアログ・ボックスが表示されます。

操作の指定

Adapter Configuration Wizard - Step 4 of 7

Operation

The File Adapter supports four operations. There is a Read File operation that polls for incoming files in your local file system, a Write File operation that creates outgoing files, a Synchronous Read File operation that reads the current contents of a file, and a List Files operation that lists file names in specified locations. Specify the Operation type and Operation Name. Only one operation per Adapter Service may be defined using this wizard.

Operation Type: ☐ Read File ☒ **Write File** ☐ Synchronous Read File ☐ List Files

Operation Name:

6. 「**Write File**」などの操作タイプを指定します。これにより、「操作名」フィールドが自動的に設定されます。

「次へ」をクリックして「ファイル構成」ダイアログ・ボックスにアクセスします。

出力ファイルの構成

Adapter Configuration Wizard - Step 5 of 7

File Configuration

Specify the parameters for the Write File operation.

Directory specified as ☒ Physical Path ☐ Logical Name

Directory for Outgoing Files (physical path):

OutputDir Browse

File Naming Convention (po_%SEQ%.txt): PO_%SEQ%.xml

☐ Append to existing file

Write to output file when any of these conditions are met

☒ Number of Messages Equals: 1

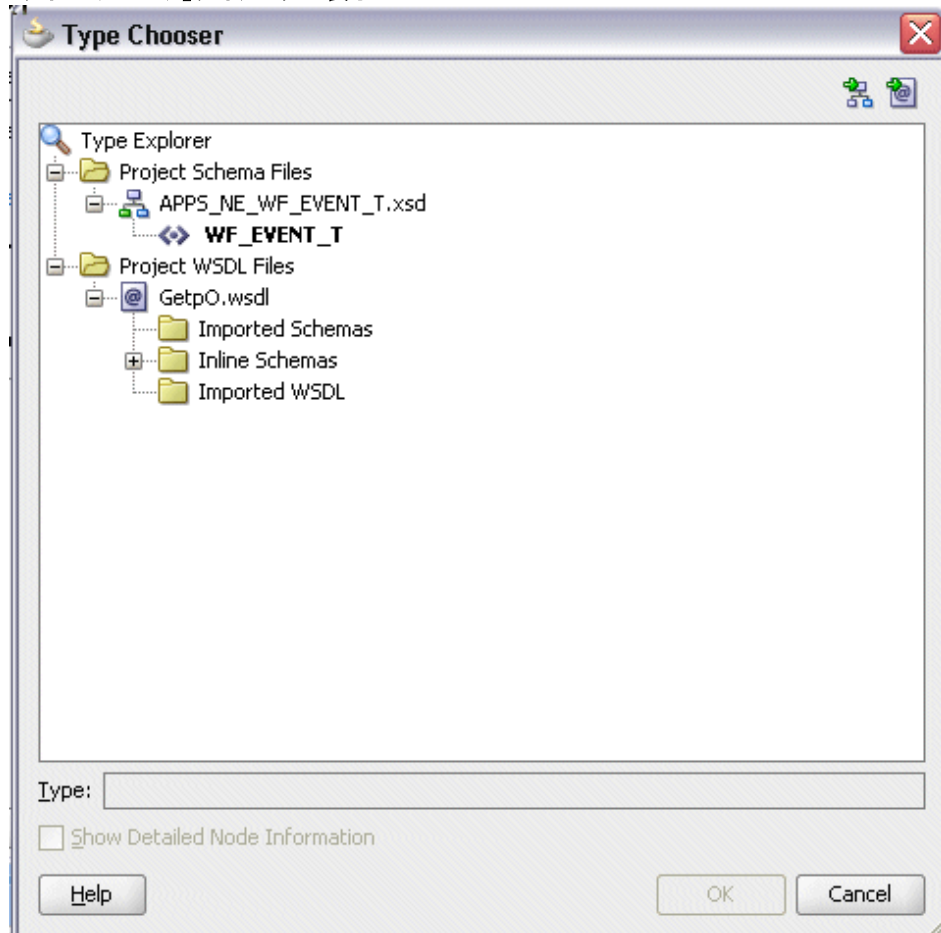
☐ Elapsed Time Exceeds: 1 minutes

☐ File Size Exceeds: 1000 kilobytes

Help < Back Next > Finish Cancel

7. 「ディレクトリは次のように指定します」フィールドで、「**物理パス**」を選択します。「発信ファイルのディレクトリ(物理パス)」フィールドにディレクトリ・パスを入力し、出力ファイルのネーミング規則を指定します(例: PO_%SEQ%.xml)。
8. デフォルトの書込み条件(「メッセージ数の到達」= 1)を確認します。「**次へ**」をクリックします。「メッセージ」ダイアログ・ボックスが表示されます。
9. 「URL」フィールドの右にある「**スキーマ・ファイルを参照**」を選択します。「タイプ・チューザ」ダイアログが表示されます。

「タイプ・チューザ」ダイアログの表示



「プロジェクトのスキーマ・ファイル」を展開し、APPS_NE_WF_EVENT_T.xsd から WF_EVENT_T スキーマを選択します。「OK」をクリックします。

選択した APPS_NE_WF_EVENT_T.xsd が URL として表示され、WF_EVENT_T 要素がスキーマ要素として選択されます。

メッセージ・スキーマの表示

Adapter Configuration Wizard - Step 6 of 7

Messages

Define the message for the Write File operation. Specify the Schema File Location and select the Schema Element that defines the messages in the outgoing files. Use the Browse button to find an existing schema definition. If you check 'Schema is Opaque', then you do not need to specify a Schema.

Message Schema

☐ Native format translation is not required (Schema is Opaque)

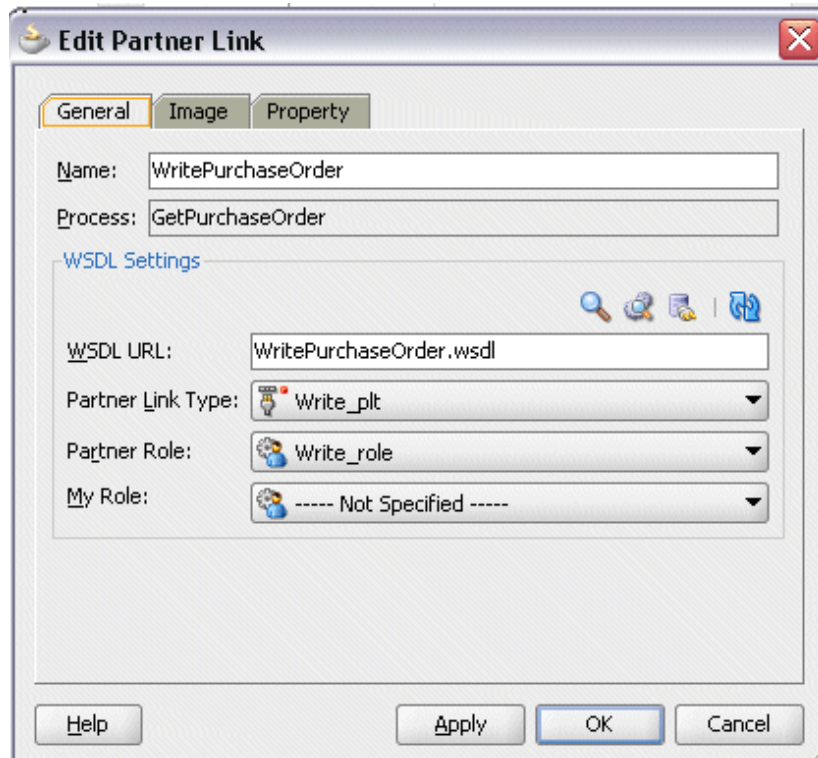
URL: xsd/APPS_NE_WF_EVENT_T.xsd

Schema Element: WF_EVENT_T

Help < Back Next > Finish Cancel

10. 「次へ」→「終了」をクリックします。ウィザードによって、パートナー・リンクに対応する WSDL ファイルが生成されます。メインの「パートナー・リンクの作成」ダイアログ・ボックスが表示され、新しい WSDL ファイル WritePurchaseOrder.wsdl が指定されます。

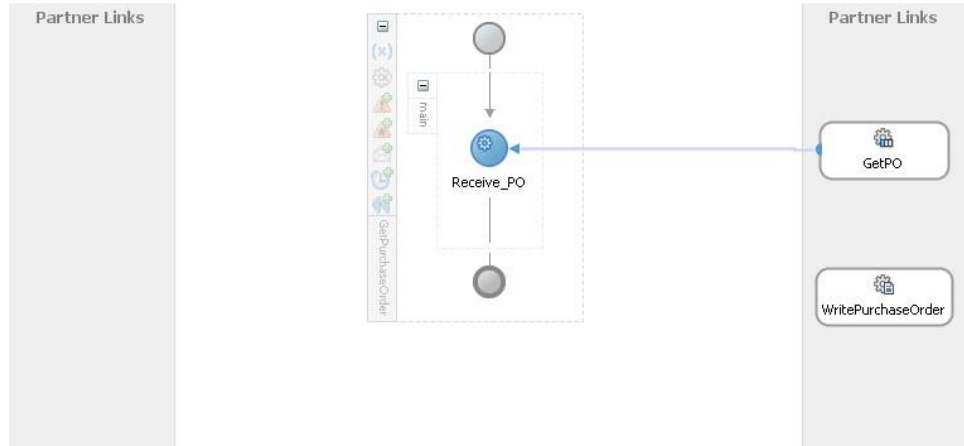
パートナ・リンクの構成の完了



「適用」→「OK」をクリックして構成を完了し、ファイル・アダプタ・サービスに必要な WSDL 設定を指定してパートナ・リンクを作成します。

WritePurchaseOrder パートナ・リンクが BPEL プロセス・ダイアグラムに表示されます。

ファイル・アダプタのパートナー・リンクの表示



Invoke アクティビティの追加

このステップでは、受入アクティビティから受け取った発注承認済イベントの詳細を WritePurchaseOrder パートナ・リンクに XML ファイルで書き込むための Invoke アクティビティを構成します。

Invoke アクティビティを追加するには:

1. Oracle JDeveloper で、コンポーネント・パレットから「**BPEL コンストラクト**」を展開します。**Invoke** アクティビティをコンポーネント・パレットからプロセス・ダイアグラムの中央のスイムレーン(受入アクティビティの後)にドラッグ・アンド・ドロップします。
2. Invoke アクティビティを WritePurchaseOrder サービスにリンクします。Invoke アクティビティでは、イベント・データをパートナ・リンクに送信します。「編集 Invoke」ダイアログ・ボックスが表示されます。

Invoke アクティビティの編集

Edit Invoke

Annotations Assertions Skip Condition Headers

General Correlations Properties

Name:

Conversation ID:

Detail Label:

☐ Invoke as Detail

Interaction Type: Partner Link ▼

Partner Role Web Service Interface

Partner Link:

Operation: Write ▼

Variables

Input:

Output:

Help Apply OK Cancel

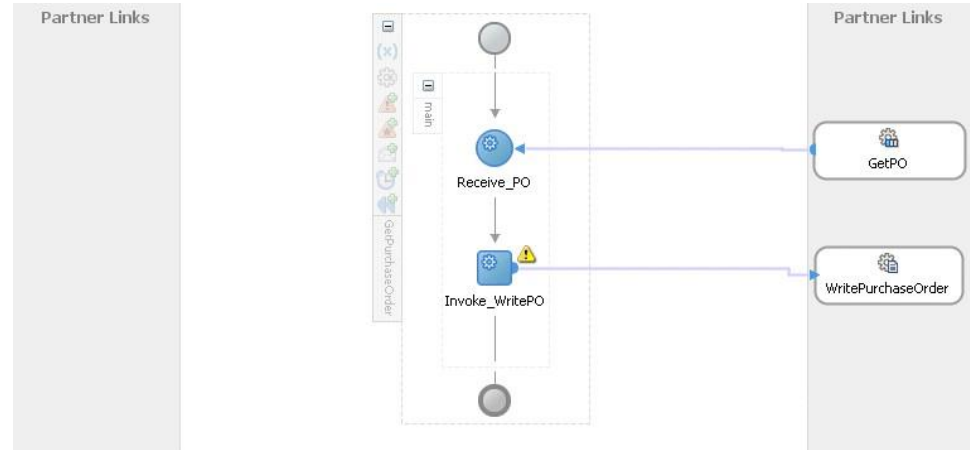
3. Invoke アクティビティの名前を入力し、「入力変数」フィールドの横にある「作成」アイコンをクリックして、新しい変数を作成します。「変数の作成」ダイアログ・ボックスが表示されます。

4. 「グローバル変数」を選択し、変数の名前を入力します。デフォルト名を受け入れることもできます。「OK」をクリックして「変数の作成」ダイアログ・ボックスを閉じます。

「編集 Invoke」ダイアログ・ボックスで「適用」→「OK」をクリックして、Invoke アクティビティの構成を終了します。

プロセス・ダイアグラムに Invoke アクティビティが表示されます。

Invoke アクティビティを含むプロセス・ダイアグラム



Assign アクティビティの追加

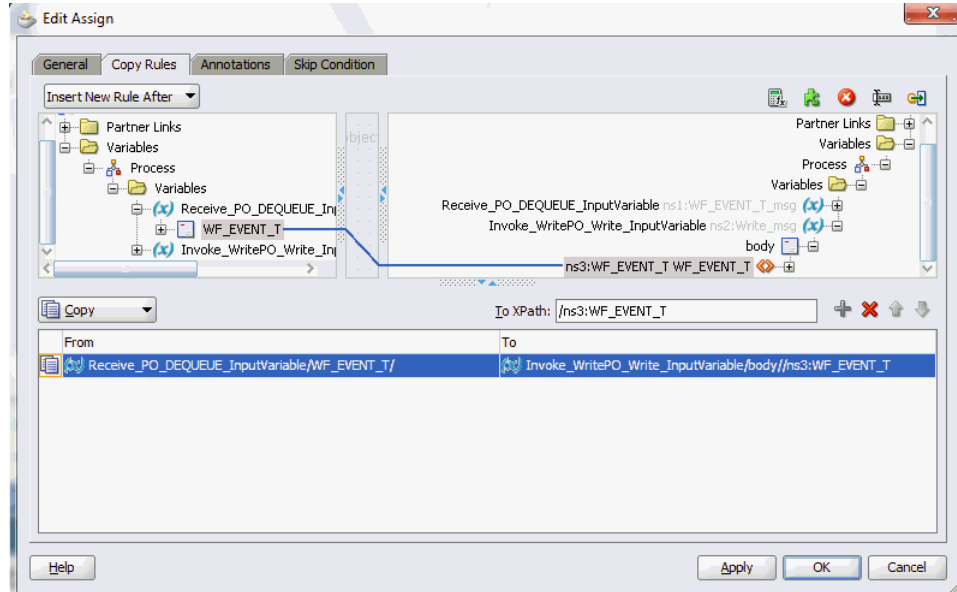
このステップを使用して、発注承認済イベントの詳細を受入アクティビティから Invoke アクティビティに渡します。

Assign アクティビティを追加するには:

1. Oracle JDeveloper で、コンポーネント・パレットから「**BPEL コンストラクト**」を展開します。
Assign アクティビティをコンポーネント・パレットからプロセス・ダイアグラムの中央のスイムレーン(受入アクティビティと **Invoke** アクティビティの間)にドラッグ・アンド・ドロップします。
2. **Assign** アクティビティをダブルクリックして、「編集 Assign」ダイアログ・ボックスにアクセスします。
3. 「コピー・ルール」タブを選択して、ソース・ツリーとターゲット・ツリーを展開します。
 - 「元」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「**Receive_PO_Dequeue_InputVariable**」に移動し、「**WF_EVENT_T**」要素を選択します。
 - 「先」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「**Invoke_WritePO_Write_InputVariable**」に移動し、「**body**」を選択します。

ソース・ノード(WF_EVENT_T)をドラッグして、先ほど識別したターゲット・ノード(body)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

パラメータの割当て



4. 「編集 Assign」ダイアログ・ボックスで「適用」→「OK」をクリックして、Assign アクティビティの構成を完了します。

Composite.xml をクリックして、Oracle JDeveloper のコンポジット・ダイアグラムを表示します。

実行時における BPEL プロセスによる SOA コンポジット・アプリケーションのデプロイおよびテスト

BPEL プロセス内に購読対象のイベントを含む SOA コンポジット・アプリケーションを作成したら、Oracle WebLogic 管理対象サーバーにデプロイする必要があります。これは、Oracle JDeveloper を使用して達成できます。コンポジットがデプロイされたら、Oracle Enterprise Manager Fusion Middleware Control コンソールからテストできます。

前提条件

Oracle JDeveloper を使用して BPEL プロセスによる SOA コンポジットをデプロイする前に、設計時環境とランタイム・サーバーの間の接続を確立する必要があります。必要なサーバー接続を構成する方法の詳細は、「サーバー接続の構成」(ページ B-1)を参照してください。

注意: WebLogic サーバーのローカル・インスタンスが使用される場合、Oracle JDeveloper から「実行」→「サーバー・インスタンスの起動」を選択して、WebLogic サーバーを起動します。

WebLogic 管理サーバーの DefaultServer インスタンスが正常に起動されたら、
「実行中:DefaultServer」と「メッセージ」のログに<Server started in Running mode>
と DefaultServer started のメッセージが表示されるはずです。

次の実行時タスクを実行します。

1. BPEL プロセスによる SOA コンポジット・アプリケーションのデプロイ、ページ 6-23
2. BPEL プロセスによる SOA コンポジット・アプリケーションの手動での開始、ページ 6-26

BPEL プロセスによる SOA コンポジット・アプリケーションのデプロイ

BPEL プロセスを手動でテストする前に、まずデプロイする必要があります。

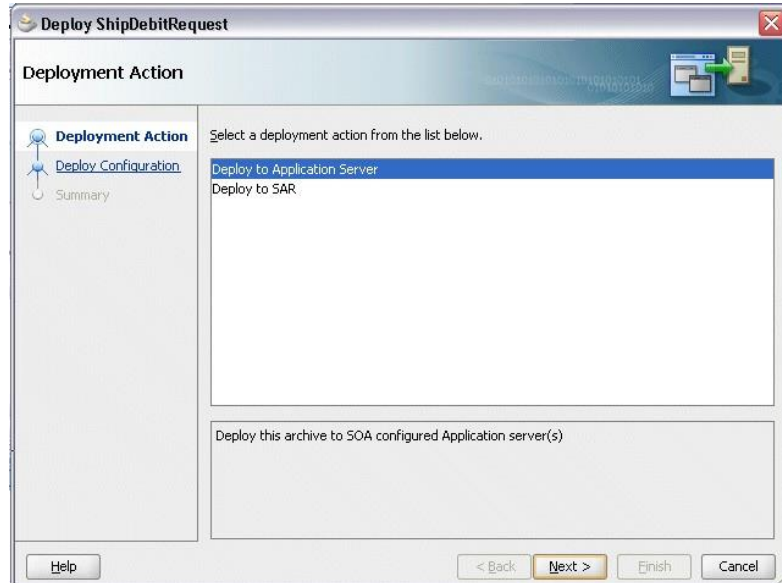
BPEL プロセスによる SOA コンポジット・アプリケーションをデプロイするには:

1. JDeveloper のアプリケーション・ナビゲータで、**GetPOProject** プロジェクトを選択します。
2. プロジェクトを右クリックして、メニューから「デプロイ」→「[project name]」→
「[serverConnection]」を選択します。

たとえば、接続を適切に設定していれば、「デプロイ」→「GetPOProject」→
「SOAServer」を選択してプロセスをデプロイできます。

注意: サーバー接続を初めて設定する場合は、「デプロイメント・アクション」ダイア
ログが表示されます。「アプリケーション・サーバーにデプロイ」を選択して、「次
へ」をクリックします。

「デプロイメント・アクション」ダイアログ



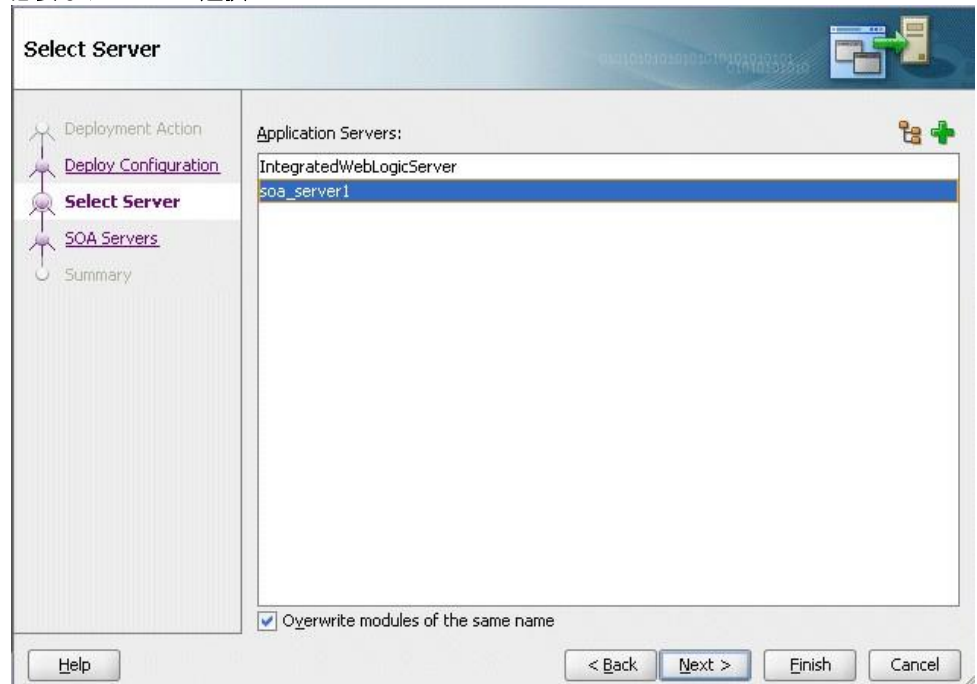
「構成のデプロイ」ダイアログで、「次へ」をクリックして新しいアプリケーション・サーバーを追加する前に、次の情報が選択されるようにします。

- 新規リビジョン ID: 1.0
- コンポジット・リビジョンをデフォルトとしてマークします。: このチェック・ボックスを選択します。
- 同じリビジョン ID で既存のコンポジットを上書きします。: このチェック・ボックスを選択します。

JDeveloper から新しい Oracle WebLogic Server 接続を作成するステップは、「サーバー接続の構成」(ページ B- 1)を参照してください。

3. 「サーバーの選択」ダイアログで、以前にサーバー接続を確立した soa-server1 を選択します。

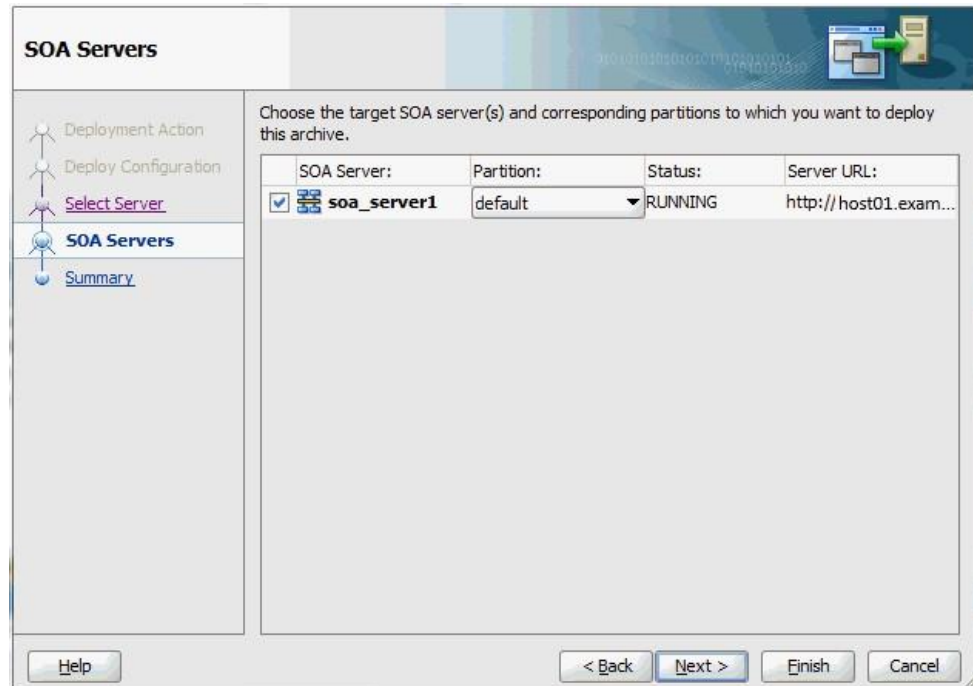
必要なサーバーの選択



「次へ」をクリックします。

4. 「SOA サーバー」ダイアログで、デフォルトのターゲット SOA サーバー(soa-server1)の選択を受け入れます。

必要な SOA サーバーの選択



「次へ」→「終了」をクリックします。

5. Oracle JDeveloper セッションから初めてコンポジットをデプロイする場合は、「認証リクエスト」ウィンドウが表示されます。Oracle SOA Suite のインストール時に指定したユーザー名とパスワード情報を入力します。「OK」をクリックします。
6. デプロイ処理が開始します。SOA - ログ・ウィンドウで、デプロイ・プロセスをモニターして、コンパイルが成功したことを確認します。

デプロイメント - ログ・ウィンドウで、デプロイが成功したことを確認します。

BPEL プロセスによる SOA コンポジット・アプリケーションのテスト

SOA コンポジット・アプリケーションに含まれる BPEL プロセスが正常にデプロイされたら、Oracle Enterprise Manager Fusion Middleware Control コンソールからプロセスを管理およびモニターできます。プロセスを手動で開始して、プロセスと統合インタフェースをテストすることもできます。

Oracle E-Business Suite にログインして発注承認および発注確認プロセスを手動で開始し、関連するイベントが呼び出され、さらに、更新された発注詳細が XML ファイルに書き込まれることを確認します。

SOA コンポジット・アプリケーションに含まれている BPEL プロセスを手動でテストするには:

1. Oracle Enterprise Manager Fusion Middleware Control コンソール
(<http://<hostname>:<port>/em>)にナビゲートします。ログイン・ページが表示さ

れます。

2. Oracle SOA Suite のインストール中に指定したユーザー名およびパスワード情報を入力します。「**ログイン**」をクリックして、ファームにログインします。デプロイしたコンポジット (GetPurchaseOrder) がアプリケーション・ナビゲーション・ツリーに表示されます。

複数のターゲット Oracle Enterprise Manager Fusion Middleware Control コンソール・ファームがある場合、適切なターゲット・インスタンス・ファームを選択する必要がある場合があります。

Oracle SOA Suite の詳細は、*Oracle Fusion Middleware Oracle SOA Suite 管理者ガイド* を参照してください。

3. 「ファーム」ナビゲーション・ペインのツリーで「SOA」→「soa-infra」ノードを展開し、SOA インフラストラクチャのホーム・ページおよびメニューを介して、soa-infra 管理対象サーバーで実行されているデプロイ済の SOA コンポジット・アプリケーションにアクセスします。

GetPurchaseOrder がデプロイされていることを確認します。

4. 「XML ゲートウェイ」職責を持つユーザーとして Oracle E-Business Suite にログインします。

ここでは、定義済の有効な仕入先が発注に含まれるように XML ゲートウェイの取引先が正しく設定されていることを確認します。

5. ナビゲーション・メニューから「取引先の定義」を選択して、「取引先設定」ウィンドウにアクセスします。

6. 「取引先設定」フォームに次のようにヘッダー値を入力します。

- 取引先タイプ: 仕入先
- 取引先名: Advanced Network Devices
- 取引先サイト: 取引先サイトの情報を入力します(例: 2000 Century Way, Santa Clara, CA 95613-4565)。
- 会社管理者 E メール: 有効な E メール・アドレスを入力します。

7. 次の取引先詳細を入力します。

- トランザクション・タイプ: PO
- トランザクション・サブタイプ: PRO
- 標準コード: OAG
- 外部トランザクション・タイプ: PO
- 外部トランザクション サブタイプ: Process

- 方向: OUT
- マップ: itg_process_po_007_out
- 接続/ ハブ: DIRECT
- プロトコル・タイプ: SOAP

取引先情報の入力

Transaction Type	Transaction SubType	Standard Code	External Transaction Type	External Transaction SubType	Direction	Map	Connection/Hub	Protocol Type
PO	PRO	OAG	PO	PROCESS	OUT	itg_process_p	DIRECT	SOAP

- 取引先詳細を保存します。職責を「購買, Vision オペレーション (USA)」に切り替え、ナビゲーション・メニューから「発注」を選択します。
- 「発注」ウィンドウで、前に定義した取引先を表すヘッダー値を指定して発注を作成します。
 - 仕入先: 仕入先情報を入力します(例: Advanced Network Devices)。
 - サイト: サイト情報を選択します(例: SANTA CLARA-ERS)。
- 「明細」タブで、次の値を指定してデータ行を入力します。
 - タイプ: 商品
 - 品目: CM13139

14. BPEL サービス・コンポーネント・インスタンスのリンク(例: GetPurchaseOrder)をクリックして、「インスタンス」ページを表示します。ここでは、「監査証跡」タブで BPEL アクティビティの実行の詳細を参照できます。「フロー」タブをクリックして、BPEL プロセスのフロー・ダイアグラムを確認します。
15. BPEL プロセス・ダイアグラムで受入アクティビティをダブルクリックし、「XML 文書の表示」リンクをクリックして XML ファイルを開きます。発注(番号 5789)を受信していることを確認してください。

受入イベント名の調査

```
<?xml version="1.0" encoding="UTF-8" ?>
- <WF_EVENT_T xmlns="http://xmlns.oracle.com/xdb/APPS/GetPurchaseOrder/GetPurchaseOrder">
  <PRIORITY xmlns="">50</PRIORITY>
  <SEND_DATE xmlns="">2008-06-07T05:36:41.000-07:00</SEND_DATE>
  <RECEIVE_DATE xmlns="">2008-06-07T05:38:30.000-07:00</RECEIVE_DATE>
  <CORRELATION_ID xmlns="" />
- <PARAMETER_LIST xmlns="">
  - <PARAMETER_LIST_ITEM>
    <NAME>APPLICATION_ID</NAME>
    <VALUE>201</VALUE>
  </PARAMETER_LIST_ITEM>
  - <PARAMETER_LIST_ITEM>
    <NAME>DOCUMENT_DIRECTION</NAME>
    <VALUE>OUT</VALUE>
  </PARAMETER_LIST_ITEM>
  - <PARAMETER_LIST_ITEM>
    <NAME>DOCUMENT_NO</NAME>
    <VALUE>5789</VALUE>
  </PARAMETER_LIST_ITEM>
  - <PARAMETER_LIST_ITEM>
    <NAME>ECX_DEBUG_LEVEL</NAME>
    <VALUE>0</VALUE>
  </PARAMETER_LIST_ITEM>
  - <PARAMETER_LIST_ITEM>
    <NAME>ECX_DOCUMENT_ID</NAME>
    <VALUE>5789:0:204</VALUE>
  </PARAMETER_LIST_ITEM>
  - <PARAMETER_LIST_ITEM>
    <NAME>ECX_PARAMETER1</NAME>
    <VALUE />
  </PARAMETER_LIST_ITEM>
  - <PARAMETER_LIST_ITEM>
    <NAME>ECX_PARAMETER2</NAME>
    <VALUE>0</VALUE>
  </PARAMETER_LIST_ITEM>
</PARAMETER_LIST>
</WF_EVENT_T>
```

16. 呼び出されたイベントと文書番号について、Assign アクティビティと Invoke アクティビティも調べます。
17. 書込み操作について指定したディレクトリ(例: outputDir)に移動します(通常は c:\temp です)。出力ファイル(例: PO_1.xml)を開き、発注番号が承認済の発注のもと同じであることを確認します。

出力の発注番号の確認

```
<PRIORITY xmlns="" 50</PRIORITY>
<SEND_DATE xmlns="" 2008-06-07T05:35:41.000-07:00</SEND_DATE>
<RECEIVE_DATE xmlns="" 2008-06-07T05:38:30.000-07:00</RECEIVE_DATE>
<CORRELATION_ID xmlns="" />
<PARAMETER_LIST xmlns="">
  <PARAMETER_LIST_ITEM>
    <NAME>APPLICATION_ID</NAME>
    <VALUE>201</VALUE>
  </PARAMETER_LIST_ITEM>
  <PARAMETER_LIST_ITEM>
    <NAME>DOCUMENT_DIRECTION</NAME>
    <VALUE>OUT</VALUE>
  </PARAMETER_LIST_ITEM>
  <PARAMETER_LIST_ITEM>
    <NAME>DOCUMENT_NO</NAME>
    <VALUE>8788</VALUE>
  </PARAMETER_LIST_ITEM>
  <PARAMETER_LIST_ITEM>
    <NAME>ECK_DEBUG_LEVEL</NAME>
    <VALUE>0</VALUE>
  </PARAMETER_LIST_ITEM>
  <PARAMETER_LIST_ITEM>
    <NAME>ECK_DOCUMENT_ID</NAME>
    <VALUE>8789:0:204</VALUE>
  </PARAMETER_LIST_ITEM>
  <PARAMETER_LIST_ITEM>
    <NAME>ECK_PARAMETER1</NAME>
    <VALUE>/>
  </PARAMETER_LIST_ITEM>
  <PARAMETER_LIST_ITEM>
    <NAME>ECK_PARAMETER2</NAME>
    <VALUE>0</VALUE>
  </PARAMETER_LIST_ITEM>
  <PARAMETER_LIST_ITEM>
    <NAME>ECK_PARAMETERS</NAME>
    <VALUE>1318:50578:201</VALUE>
  </PARAMETER_LIST_ITEM>
  <PARAMETER_LIST_ITEM>
    <NAME>ECK_PARAMETER4</NAME>
    <VALUE>97094</VALUE>
  </PARAMETER_LIST_ITEM>
  <PARAMETER_LIST_ITEM>
    <NAME>ECK_PARAMETERS5</NAME>
    <VALUE>204</VALUE>
  </PARAMETER_LIST_ITEM>
  <PARAMETER_LIST_ITEM>
    <NAME>ECK_PARTY_ID</NAME>
```


コンカレント・プログラムの使用

概要

コンカレント・プログラムは、関連するパラメータを含む実行ファイルのインスタンスです。コンカレント・プログラムは、コンカレント・プログラム実行ファイルを使用して、正しい実行ファイルを見つけます。実行ファイルは、オペレーティング・システム・ファイル、またはアプリケーション・ロジック(PL/SQL、Java など)を含むデータベース・ストアード・プロシージャです。複数のコンカレント・プログラムが、同じ実行ファイルを使用して特定のタスクを実行する場合があります、それぞれに異なるパラメータ・デフォルトがあります。

コンカレント・プログラムは、SOAP ベースおよび REST ベースのサービスとして公開できます。この章では、サービス起動を説明するための例として、コンカレント・プログラム SOAP サービスを使用します。デプロイ済 SOAP サービスは、サービス・エンドポイントを持つ SOA コンポジット・アプリケーション内の有用な BPEL プロセスにオーケストレーションできます。実行時に、soa-infra アプリケーションが実行されている WebLogic 管理対象サーバーの SOA コンポジットを顧客に公開し、Web サービス・クライアントまたはオーケストレーション・ツール (Oracle JDeveloper、Apache Axis、.NET Web サービス・クライアント、Oracle BPEL Process Manager、Oracle Enterprise Service Bus (ESB)など)によって起動できます。

この章では、BPEL プロセスによる SOA コンポジット・アプリケーションを作成して Web サービスを起動し、これを使用して Oracle E-Business Suite を更新する方法の詳細について説明します。この後の各項で説明する例では、Oracle JDeveloper 11g (11.1.1.6.0)を設計時ツールとして使用して SOA コンポジット・アプリケーションを作成し、Oracle SOA Suite 11g (11.1.1.6.0)を使用してこのプロセスをデプロイします。

注意: Oracle JDeveloper とその他の Oracle Fusion Middleware コンポーネント (Oracle SOA Suite など)を使用している場合、SOA テクノロジを有効にするには、SOA テクノロジ用の JDeveloper の拡張機能である Oracle SOA Suite コンポジット・エディタを手動でダウンロードする必要があります。追加の Oracle Fusion Middleware 設計時コンポーネントのインストールの詳細は、*Oracle Fusion Middleware Oracle JDeveloper インストレーション・ガイド*を参照してください。

設計時におけるコンカレント・プログラム WSDL の使用

BPEL プロセスによる SOA コンポジット・アプリケーションのシナリオ

この例では、出発出荷通知アウトバウンド WSHDSNO コンカレント・プログラムを使用して、BPEL プロセスの作成について説明します。

出荷通知生成要求を、SOA コンポジット・アプリケーションに含まれる BPEL プロセスへの入力として受け取ると、ヘッダーおよび明細品目を含む受注情報がファイル・アダプタによって読み取られます。その後、受注データが渡されて、出発出荷通知(DSNO)が作成されます。出荷通知作成文書番号が、要求アプリケーションに渡されます。

デプロイ後に BPEL プロセスが正常に実行されると、プロセスを検証して、生成された出荷通知に受注に記載されている正しい取引先情報があるかどうかを確認できます。

コンカレント・プログラム Web サービスを使用して BPEL プロセスによる SOA コンポジット・アプリケーションを作成するシナリオの前提条件

コンカレント・プログラムの設計時タスクを実行する前に、次のタスクを実行しておく必要があります。

- 統管理者または統合開発者は、事前に Web サービスを生成しておく必要があります。管理者が、Oracle SOA Suite WebLogic 管理対象サーバーに、生成されたサービスをデプロイします。
- 統合開発者は、Web サービスとして公開されているコンカレント・プログラムのデプロイ済 WSDL URL を見つけて記録しておく必要があります。
- SOAP 要求に対してコンカレント・プログラムを実行するために、SOAHeader 要素を設定する必要があります

コンカレント・プログラム Web サービス・コンポジットのデプロイ

選択したインタフェース定義の Web サービスが正常に生成されると、管理者は、Oracle Integration Repository から Oracle SOA Suite WebLogic 管理対象サーバーにサービスをデプロイできます。

たとえば、統合開発者がデプロイ済 WSDL を使用して BPEL プロセスによる SOA コンポジット・アプリケーションを作成する前に、まず次のステップを実行する必要があります。

1. Web サービスを生成するには、統管理者または統合開発者がまず Oracle Integration Repository からインタフェース定義(出発出荷通知アウトバウンド WSHDSNO コンカレント・プログラムなど)を見つけ、インタフェースの詳細ページの「SOAP Web サービス」タブで「生成」をクリックします。

Web サービスの生成方法の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「Web サービスの生成」を参照してください。

2. 生成された Web サービスを配置するには、管理者はいずれかの認証タイプを選択してから、インタフェースの詳細ページの「SOAP Web サービス」タブで「**配置**」をクリックします。Oracle SOA Suite に配置されたサービスは、有効なサービスであり、新しい SOAP 要求を受け入れる準備が整っています。

サービスが正常に配置されたら、「SOAP サービス・ステータス」フィールドが「有効」の状態となり、「配置済」とともに、選択した認証タイプが表示されます。認証タイプによる Web サービスの保護の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「Web サービス・セキュリティの管理」を参照してください。

Web サービスの配置方法の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「Web サービスの配置および配置解除」を参照してください。

WSDL URL の検索および記録

管理者が実行する必要があるタスクとは別に、統合開発者は、Oracle JDeveloper で有用な BPEL プロセスにオーケストレーションする必要があるインタフェース(WSHDSNO コンカレント・プログラムなど)のデプロイ済 Web サービスの WSDL URL を見つけて記録する必要があります。

この WSDL 情報は、設計時の BPEL プロセスの作成中にパートナ・リンクに直接使用されます。

デプロイ済 WSDL URL の確認と記録

Integration Repository

Integration Repository >

Concurrent Program : Departure Shipment Notice Outbound

Browse Search Printable Page

Log Configuration Disabled

Internal Name WSHDSNO
Type Concurrent Program
Product Shipping Execution Common
Status Active
Business Entity [Trip](#)

Scope Public
Interface Source Oracle

Full Description

This concurrent program generates a Departure Ship Notice Outbound(DSNO). DSNO is a notice used by the supplier to provide advance notification of a shipment to the buyer or trading partner that the material has been shipped. The Outbound Ship Notice/Manifest transaction serves business needs of both the sender and the receiver of the transaction.

Web Service

Web Service Status Deployed | Active | [View WSDL](#)
Interaction Pattern Synchronous

* Authentication Type

☒ Username Token
☐ SAML Token (Sender Vouches)

Source Information

Parameters

Name	Type	Required	Displayed	Description
Trip	WSH_SRS_TRIPS	Yes	Yes	Trip
Trip Stop	WSH_SRS_TRIP_REL_STOPS	Yes	Yes	Trip Stop

Methods

Name	Internal Name	Status	Description
Process	Process	Active	Departure Ship Notice Outbound

インタフェースを検索してインタフェースの詳細をレビューする方法の詳細は、「統合インタフェースの検索および表示」(ページ 2-1)を参照してください。

SOAP 要求の SOAHeader の変数の設定

サービスの実行中に、アプリケーション・コンテキストを設定するために要求される値を SOAHeader 要素で渡すには、特定の変数を設定する必要があります。これらのコンカレント・プログラム・インタフェース・タイプの SOAHeader 要素は、*Responsibility*、*RespApplication*、*SecurityGroup*、*NLSLanguage* および *Org_Id* です。

注意: ユーザー名とパスワード情報は、Web サービス・セキュリティ・ポリシー (oracle/wss_username_token_service_policy など)によって定義されます。SOAP 要求とともにセキュリティ・ヘッダーを渡す方法の詳細な手順は、「Web サービス・ポリシーの構成」(ページ 7-20)を参照してください。

これらの要素の予期される値を次の表に示します。

コンカレント・プログラム・インタフェース・タイプのヘッダー変数と予期される値

要素名	予期される値
Responsibility	responsibility_key (SYSTEM_ADMINISTRATOR など)
RespApplication	アプリケーション短縮名(FND など)
SecurityGroup	セキュリティ・グループ・キー(STANDARD など)
NLSLanguage	NLS 言語(AMERICAN など)
Org_Id	組織 ID (202 など)

注意: NLS 言語と組織 ID は、オプションで渡される値です。

- NLS 言語の要素が指定された場合、渡された言語で SOAP 要求を使用できます。すべての対応する SOAP 応答とエラー・メッセージも、同じ言語で戻すことができます。言語が指定されていない場合、ユーザーのデフォルトの言語が使用されます。
- サービスの実行が特定組織に依存する場合は、その SOAP 要求の Org_Id 要素を渡す必要があります。

BPEL PM で Invoke アクティビティの前に Assign アクティビティを構成することによって、コンテキスト情報を指定できます。

BPEL プロセスによる SOA コンポジット・アプリケーションの作成フロー

この章では、シナリオに基づいて次の設計時タスクを説明します。

1. 新しい BPEL プロセスによる SOA コンポジット・アプリケーションの作成、ページ 7-6

このステップを使用して、ShipNotice.bpel という、新しい BPEL プロセスによる SOA コンポジット・アプリケーションを作成します。これにより、サード・パーティ・アプリケーションからの入力を受け入れる受入と、要求アプリケーションに BPEL プロセスの出力を返信する返信の 2 つのダミー・アクティビティが自動的に作成されます。

2. パートナ・リンクの作成、ページ 7-8

このステップを使用して、出発出荷通知アウトバウンド Shipment_Notice コンカレント・サービスのパートナ・リンクを作成します。

3. ファイル・アダプタのパートナ・リンクの追加、ページ 7-9

これは、取引先から受け取った受注詳細を同期的に取り扱います。

4. Invoke アクティビティの追加、ページ 7-14

このステップを使用して、次の目的の 2 つの Invoke アクティビティを作成します。

1. ファイル・アダプタ - 同期読取り操作を指し示して、Assign アクティビティから受注詳細を読み取ります。
2. Shipment_Notice Web サービスを指し示して、ヘッダーおよび明細詳細を含む出荷通知を作成します。

5. Assign アクティビティの追加、ページ 7-16

このステップを使用して、次のことを目的とした 3 つの Assign アクティビティを作成します。

1. DSNO コンカレント・プログラム・サービスを起動するための SOAHeader 変数を渡します。
2. 同期読取り - ファイル・アダプタ・サービスの出力からの受注詳細を DSNO 作成の入力に渡します。
3. 出力する SOAP レスポンスを設定します。

BPEL プロセスのサービス・コンポーネントを使用した SOA コンポジット・アプリケーションの一般情報と作成方法の詳細は、*Oracle Fusion Middleware Oracle SOA Suite 開発者ガイド*を参照してください。

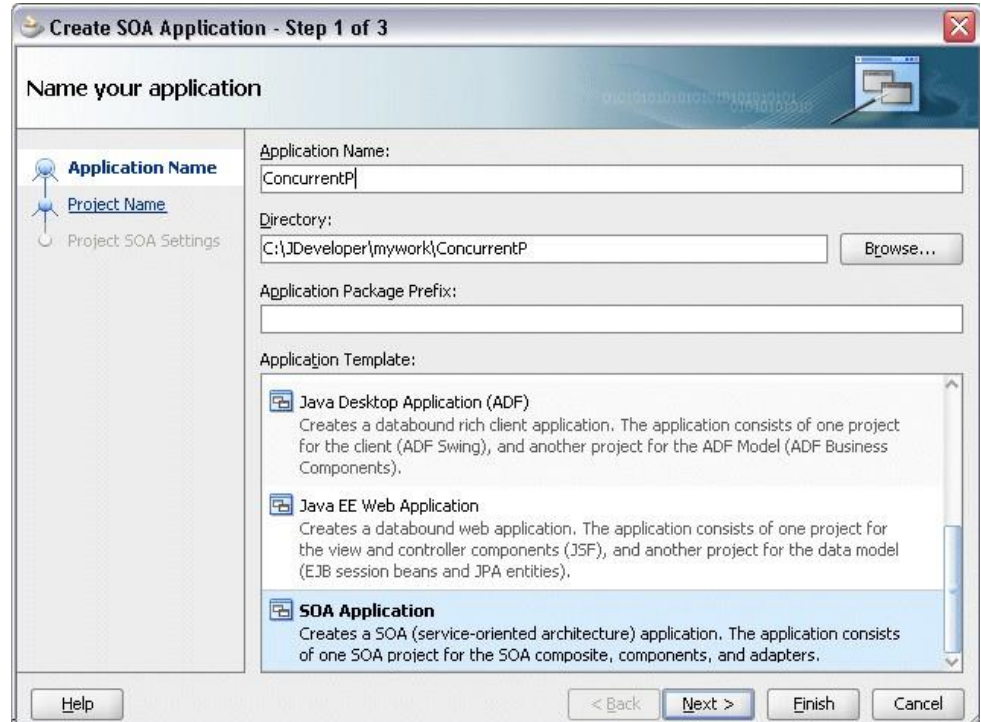
新しい BPEL プロセスによる SOA コンポジット・アプリケーションの作成

このステップを使用して、様々な BPEL プロセス・アクティビティによる新規 SOA コンポジット・アプリケーションを作成します。

新しい BPEL プロセスによる SOA コンポジット・アプリケーションを作成するには:

1. Oracle JDeveloper を開きます。
2. アプリケーション・ナビゲータで「**新規アプリケーション**」をクリックします。
「SOA アプリケーションの作成 - アプリケーションの名前付け」ページが表示されます。
3. 「アプリケーション名」フィールドにアプリケーションの適切な名前を入力して、「アプリケーション・テンプレート」リストから「**SOA アプリケーション**」を選択します。
「次へ」をクリックします。「SOA アプリケーションの作成 - プロジェクトの名前付け」ページが表示されます。

「SOA アプリケーションの作成」の「アプリケーションの名前付け」ページ



4. 「プロジェクト名」フィールドにプロジェクトの適切な名前を入力します(ShipNotice など)。

5. 「プロジェクト・テクノロジー」タブで、「Web サービス」を選択して、「使用可能」のテクノロジー・リストから「選択済」のテクノロジー・リストに「SOA」が選択されるようにします。

「次へ」をクリックします。「SOA アプリケーションの作成 - SOA 設定の構成」ページが表示されます。

6. 「コンポジット・テンプレート」リストから**コンポジット BPEL プロセス**を選択して、「終了」をクリックします。新規アプリケーションと SOA プロジェクトを作成しました。これにより、SOA コンポジットが自動的に作成されます。

「BPEL プロセスの作成」ページが表示されます。

7. デフォルトの「**BPEL 1.1 Specification**」の選択を変更せずに維持します。これにより、BPEL 1.1 Specification をサポートする BPEL プロジェクトが作成されます。

「名前」フィールドに BPEL プロセスの適切な名前を入力します(ShipNotice など)。

「テンプレート」フィールドで「**同期 BPEL プロセス**」を選択します。

「トランザクション」ドロップダウン・リストから「**required**」を選択します。「OK」をクリックします。

受入アクティビティと返信アクティビティのある同期 BPEL プロセスが作成されます。指定した名前(ShipNotice.bpel や ShipNotice.wsdl など)を使用する bpel や wsdl などの必要なソース・ファイルおよび composite.xml も生成されます。

8. 「SOA コンテンツ」→「ビジネス・ルール」にナビゲートし、composite.xml をダブルクリックしてコンポジット・ダイアグラムを表示します。

ShipNotice コンポーネントをダブルクリックして、BPEL プロセスを開きます。

Web サービスのパートナ・リンクの作成

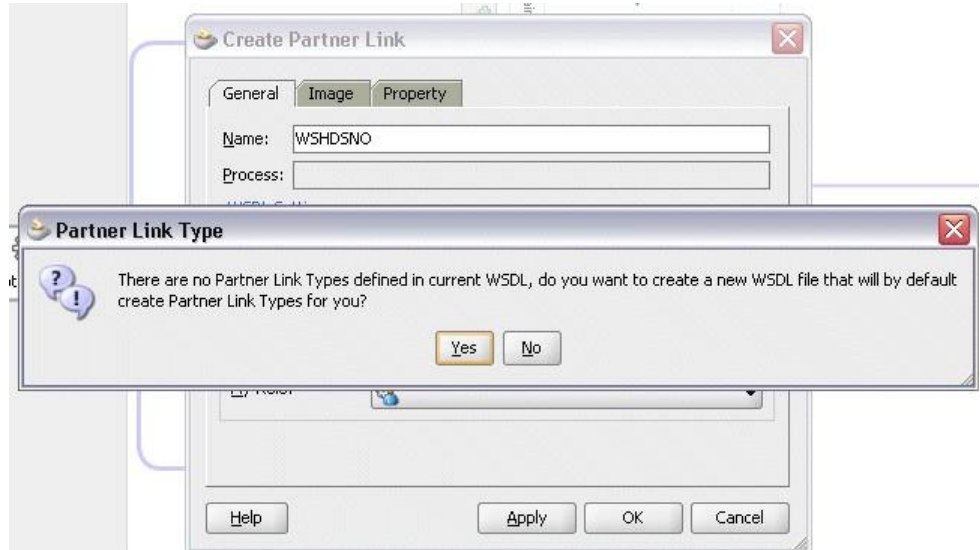
このステップを使用して、WSHDSNO コンカレント・プログラムを介して公開される Web サービスの Shipment_Notice というパートナ・リンクを作成します。

Shipment_Notice Web サービスのパートナ・リンクを作成するには:

1. Oracle JDeveloper で、「パートナ・リンク」領域にマウスを置いて右クリックして、プルダウン・メニューから「パートナ・リンクの作成」を選択します。または、「BPEL コンストラクト」リストから「パートナ・リンク」を、プロセス・ダイアグラムの右側の「パートナ・リンク」スイムレーンにドラッグ・アンド・ドロップできます。

「パートナ・リンクの作成」ウィンドウが表示されます。
2. 「WSDL ファイル」フィールドに、以前に記録した出発出荷通知アウトバウンド WSHDSNO サービスに対応する WSDL URL をコピーします。[Tab]キーを押します。
3. デフォルトでパートナ・リンク・タイプを作成する新しい WSDL ファイルをシステムで作成するかどうかを尋ねる「パートナ・リンク・タイプ」メッセージ・ダイアログ・ボックスが表示されます。

「パートナ・リンク・タイプ」メッセージ・ダイアログの表示



パートナ名の値が自動的に移入されるように、「はい」をクリックします。WSHDSNO などの名前を手動で入力できます。

パートナ・リンクは、必要な WSDL 設定で作成され、BPEL プロジェクトでは、プロセス・ダイアグラムの枠線の領域の新しいアイコンによって表現されます。

4. オプションで、アイコンをダブルクリックして、「パートナ・リンクの編集」ウィンドウを開き、デフォルトのパートナ・リンク名を変更できます。たとえば、WSHDSNO から Shipment_Notice に変更します。

ドロップダウン・リストから「パートナ・リンク・タイプ」フィールドと「パートナ・ロール」フィールドを選択します。「適用」をクリックします。

5. 「OK」をクリックして、パートナ・リンクの構成を完了します。

ファイル・アダプタのパートナ・リンクの追加

このステップを使用して、受注を同期的に読み取って受注詳細を取得することにより、BPEL プロセスを構成します。

受注詳細を読み取るファイル・アダプタのパートナ・リンクを追加するには:

1. Oracle JDeveloper で、「BPEL サービス」リストから「ファイル・アダプタ」サービスを、プロセス・ダイアグラムの右側の「パートナ・リンク」スイムレーンにドラッグ・アンド・ドロップします。ファイル・アダプタ構成ウィザードへようこそページが表示されます。
2. 「次へ」をクリックします。「サービス名」ダイアログ・ボックスが表示されます。
3. ファイル・アダプタ・サービス名(ReadOrder など)を入力します。

4. 「次へ」をクリックします。「アダプタ・インタフェース」ダイアログ・ボックスが表示されます。
5. 「操作およびスキーマから定義(後で指定)」ラジオ・ボタンを選択して、「次へ」をクリックします。「操作」ダイアログ・ボックスが表示されます。

操作の指定

Operation

The File Adapter supports four operations. There is a Read File operation that polls for incoming files in your local file system, a Write File operation that creates outgoing files, a Synchronous Read File operation that reads the current contents of a file, and a List Files operation that lists file names in specified locations. Specify the Operation type and Operation Name. Only one operation per Adapter Service may be defined using this wizard.

Operation Type: ☐ Read File ☐ Write File ☒ Synchronous Read File ☐ List Files

Operation Name:

Help < Back Next > Finish Cancel

6. 「同期 Read File」などの操作タイプを指定します。これにより、「操作名」フィールドが自動的に設定されます。

「次へ」をクリックして、「ファイル・ディレクトリ」ダイアログ・ボックスにアクセスします。

入力ファイル・ディレクトリの指定

 Physical Path ☐ Logical Name'. There is a text input field for 'Directory for Incoming Files (physical path):' containing '/usr/tmp' and a 'Browse' button. Below that is a checkbox 'Archive processed files' which is unchecked, followed by a text input field for 'Archive Directory for Processed Files (physical path):' and a 'Browse' button. At the bottom left is a checkbox 'Delete files after successful retrieval' which is checked. At the bottom right are four buttons: 'Help', '< Back', 'Next >', and 'Cancel'."/>

Adapter Configuration Wizard - Step 5 of 8

File Directories

Enter directory information for the incoming file of the Synchronous Read File operation.

Directory names are specified as: ☒ Physical Path ☐ Logical Name

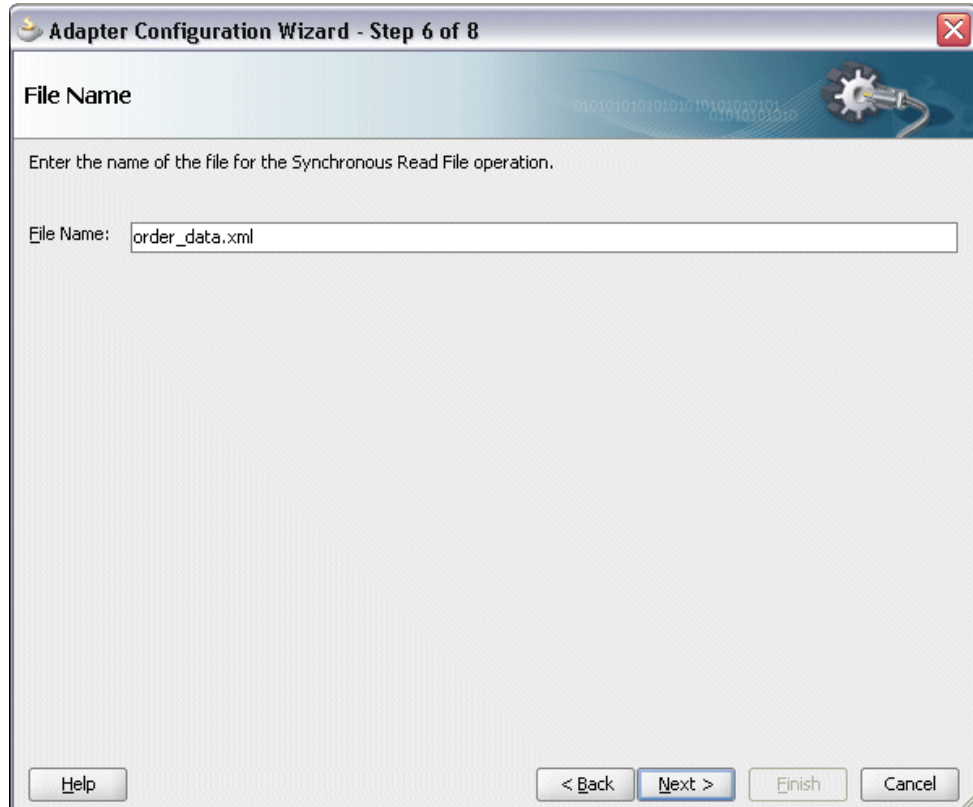
Directory for Incoming Files (physical path):

☐ Archive processed files
Archive Directory for Processed Files (physical path):

☒ Delete files after successful retrieval

7. 「物理パス」ラジオ・ボタンを選択して、入力ペイロードのファイル・ディレクトリの情報を
入力します。たとえば、ディレクトリ名として /usr/tmp/ と入力します。
「次へ」をクリックして、「ファイル名」ダイアログ・ボックスを開きます。
8. 同期 Read File 操作のファイル名 (order_data.xml など) を入力します。

入力ファイル名の指定



「次へ」をクリックします。「メッセージ」ダイアログ・ボックスが表示されます。

9. 「URL」フィールドの右にある「スキーマ・ファイルを参照」を選択します。「タイプ・チューザ」ウィンドウが表示されます。

1. 「タイプ・チューザ」ウィンドウの右上隅にある「スキーマ・ファイルのインポート」ボタンをクリックします。

2. サービスのスキーマの場所を入力します。例: `http://<soa_suite_hostname>:<port>/soa-infra/services/default/<jndi_name>_CONCURRENTPROGRAM_WSHDSN O/WSHDSNO_Service?XSD=APPS_ISG_CP_REQUEST_CP_SUBMIT.xsd.`

サービスのスキーマの場所は、サービスの WSDL URL

(`http://<soa_suite_hostname>:<port>/soa-infra/services/default/<jndi_name>_CONCURRENTPROGRAM_WSHDSN O/WSHDSNO_Service?wsdl` など)から見つけることができます。

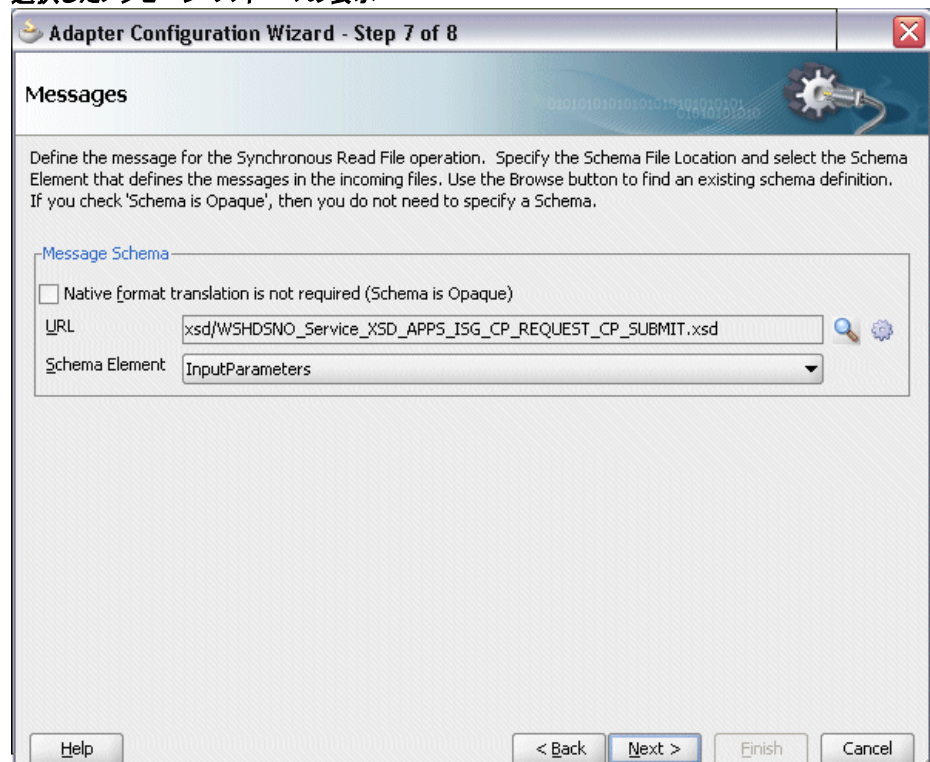
3. 「プロジェクトにコピー」チェック・ボックスを選択して、「OK」をクリックします。

「ファイルのローカライズ」ウィンドウが表示されます。「インポートされたファイルの元のディレクトリ構造を保持」チェック・ボックスが選択されていることを確認し、「OK」をクリックします。

「タイプ・チューザ」ウィンドウに、「インポートしたスキーマ」フォルダが自動的に追加されます。

4. 「インポートしたスキーマ」フォルダを展開し、
WSHDSNO_Service_XSD_APPS_ISG_CP_REQUEST_CP_SUBMIT.xsd で
InputParameters メッセージを選択します。「OK」をクリックします。
5. 選択した.xsd が URL として表示されて、InputParameters がスキーマ要素として選
択されます。

選択したメッセージ・スキーマの表示



10. 「次へ」→「終了」をクリックします。ウィザードによって、パートナ・リンクに対応する
WSDL ファイルが生成されます。メインの「パートナ・リンクの作成」ダイアログ・ボックスが
表示されるので、新しい WSDL ファイル ReadOrder.wsdl を指定します。

「適用」→「OK」をクリックして構成を完了し、ファイル・アダプタ・サービスに必要な
WSDL 設定を指定してパートナ・リンクを作成します。

BPEL プロセス・ダイアグラムに ReadOrder パートナ・リンクが表示されます。

Invoke アクティビティの追加

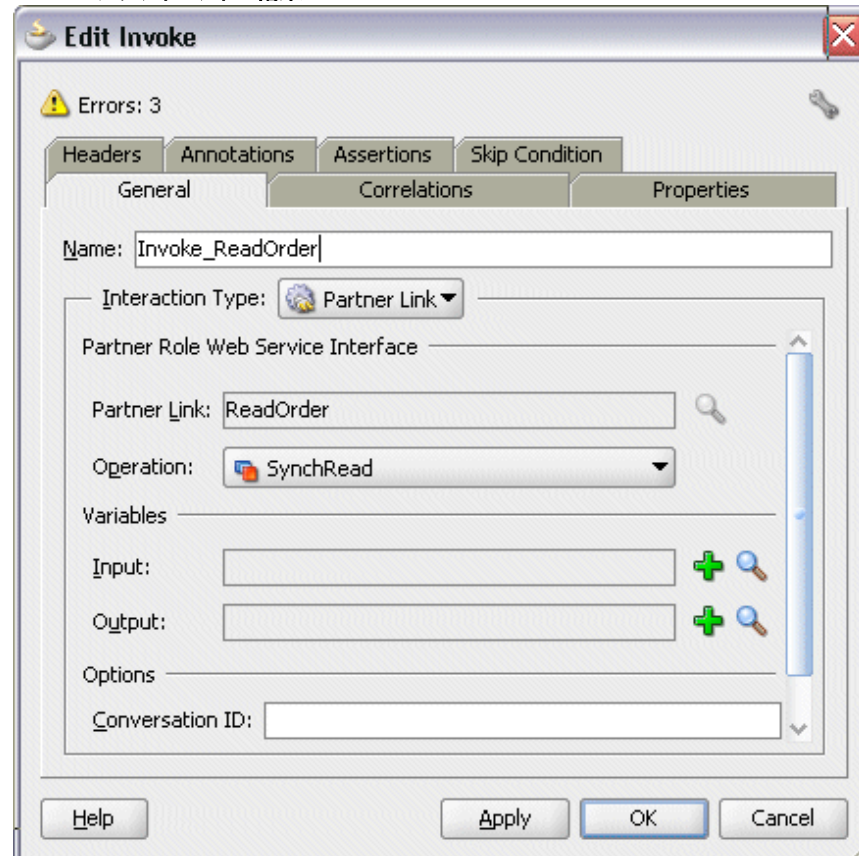
このステップでは、次の目的の 2 つの Invoke アクティビティを構成します。

- ファイル・アダプタの ReadOrder パートナ・リンクを介して、1 つ目の Assign アクティビティから渡された受注詳細を読み取ります。
- Assign アクティビティから受け取った受注ヘッダーおよび明細詳細を送信し、Shipment_Notice パートナ・リンクを介してアウトバウンド出荷通知(DSNO)を生成します。

ReadOrder パートナ・リンクの Invoke アクティビティを追加するには:

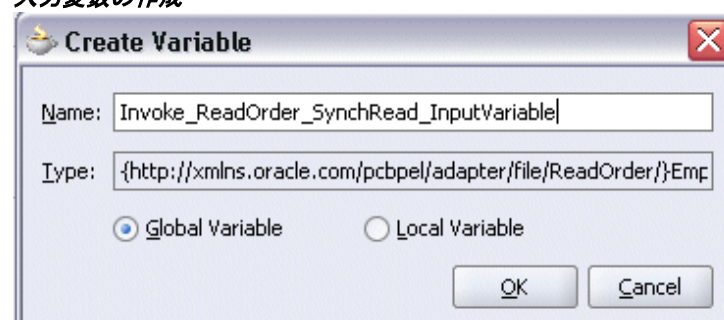
1. Oracle JDeveloper で、コンポーネント・パレットから「**BPEL コンストラクト**」を展開します。
Invoke アクティビティを、コンポーネント・パレットからプロセス・ダイアグラムの中央のスィムレーンの、**receiveInput** アクティビティと **replyOutput** アクティビティの間にドラッグ・アンド・ドロップします。
2. Invoke アクティビティを ReadOrder サービスにリンクします。Invoke アクティビティでは、受注データをパートナ・リンクに送信します。「編集 Invoke」ダイアログ・ボックスが表示されます。

Invoke アクティビティの編集



3. Invoke アクティビティの名前を入力し、「入力変数」フィールドの横にある「作成」アイコンをクリックして、新しい変数を作成します。「変数の作成」ダイアログ・ボックスが表示されます。

入力変数の作成

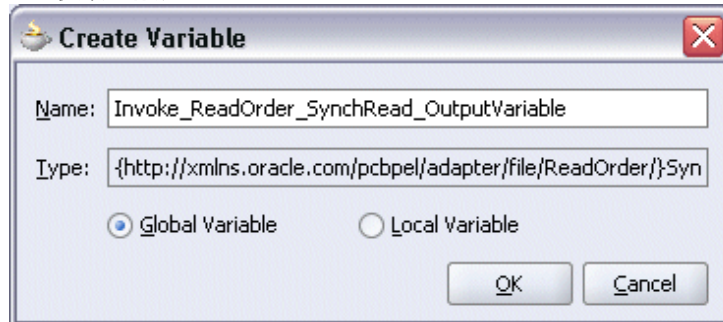


4. 変数の名前を入力します。デフォルト名を受け入れることもできます。
「グローバル変数」ラジオ・ボタンが選択されていることを確認し、「OK」をクリックします。

5. 「出力変数」フィールドの横にある「入力変数の作成」アイコンをクリックします。

出力変数の名前を入力します。デフォルト名を受け入れることもできます。「グローバル変数」ラジオ・ボタンが選択されていることを確認し、「OK」をクリックします。

出力変数の作成



6. 「編集 Invoke」ダイアログ・ボックスで「適用」→「OK」をクリックして、Invoke アクティビティの構成を終了します。

プロセス・ダイアグラムに Invoke アクティビティが表示されます。

Shipment_Notice パートナ・リンクの 2 つ目の **Invoke** アクティビティを追加するには:

1. Oracle JDeveloper で、コンポーネント・パレットから「BPEL コンストラクト」を展開します。2 つ目の **Invoke** アクティビティを、コンポーネント・パレットからプロセス・ダイアグラムの中央のスイムレーンの、**Invoke** アクティビティと **replyOutput** アクティビティの後にドラッグ・アンド・ドロップします。
2. Invoke アクティビティを **Shipment_Notice** サービスにリンクします。Invoke アクティビティでは、イベント・データをパートナ・リンクに送信します。「編集 Invoke」ダイアログ・ボックスが表示されます。
3. Invoke アクティビティの名前(**Invoke_ShipmentNotice** など)を入力します。1 つ目の Invoke アクティビティの作成手順の説明に従って、入力グローバル変数および出力グローバル変数を選択します。

「編集 Invoke」ダイアログ・ボックスで「適用」→「OK」をクリックして、Invoke アクティビティの構成を終了します。

プロセス・ダイアグラムに 2 つ目の Invoke アクティビティが表示されます。

Assign アクティビティの追加

このステップでは、次の 3 つの Assign アクティビティを構成します。

1. DSNO コンカレント・プログラム・サービスの起動における、SOAHeader のアプリケーション

ン・コンテキストの受渡し。

2. **Invoke_ShipmentNoticeInvoke** アクティビティを介した、同期読取り - ファイル・アダプタ・サービスの出力から **DSNO** 作成の入力への受注詳細の受渡し。
3. 出力する **SOAP** 応答の設定。

SOAHeader パラメータの割当て:

DSNO コンカレント・プログラム・サービスの起動で使用される **SOAHeader** 変数を渡す 1 つ目の **Assign** アクティビティを追加するには:

1. **Oracle JDeveloper** で、コンポーネント・パレットから「**BPEL コンストラクト**」を展開します。**Assign** アクティビティを、プロセス・ダイアグラムの中央のスイムレーンの、先ほど作成したばかりの 2 つの **Invoke** アクティビティの間にドラッグ・アンド・ドロップします。
2. **Assign** アクティビティをダブルクリックして、「編集 **Assign**」ダイアログ・ボックスにアクセスします。
3. 「編集 **Assign**」ダイアログ・ボックスで、割当て名として **SOAHeader** と入力します。「**OK**」をクリックします。

4. 「コピー・ルール」タブを選択し、ターゲット・ツリーを展開します。

- ・ 「式」アイコンをクリックして、「式ビルダー」ダイアログを起動します。

「式」ボックスに **ORDER_MGMT_SUPER_USER** と入力します。「**OK**」をクリックします。「割当ての編集」ダイアログの中央の、「元」と「先」のナビゲーション・ツリー・ノードの間に、「式」アイコンと式の値(**ORDER_MGMT_SUPER_USER**)が表示されます。

- ・ 「先」ナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「**Invoke_Shipment_Notice_WSHDSNO_InputVariable**」→「**header**」→「**ns2:SOAHeader**」にナビゲートして、「**ns2:Responsibility**」を選択します。

「式」アイコンをドラッグして、識別したばかりのターゲット・ノード(**ns2:Responsibility**)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。

「編集 **Assign**」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

5. 「式」アイコンをクリックして「式ビルダー」ダイアログを起動することによって、2 つ目のパラメータ・ペアを入力します。
 - ・ 「式」ボックスに **ONT** と入力します。「**OK**」をクリックします。「割当ての編集」ダイアログの中央の、「元」と「先」のナビゲーション・ツリー・ノードの間に、「式」アイコンと式の値(**ONT**)が表示されます。
 - ・ 「先」ナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「**Invoke_Shipment_Notice_WSHDSNO_InputVariable**」→「**header**」→「**ns2:**

SOAHeader」にナビゲートして、「**ns2:RespApplication**」を選択します。

「式」アイコンをドラッグして、識別したばかりのターゲット・ノード(**ns2:RespApplication**)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

6. 「式」アイコンをクリックして「式ビルダー」ダイアログを起動して、3 つ目のパラメータのペ
アを入力します。

- 「式」ボックスに **STANDARD** と入力します。「**OK**」をクリックします。「割当ての編集」ダイアログの中央の、「元」と「先」のナビゲーション・ツリー・ノードの間に、式の値(**STANDARD**)を持つ「式」アイコンが表示されます。
- 「先」ナビゲーション・ツリーで、「**変数**」→「**プロセス**」→「**変数**」→「**Invoke_Shipment_Notice_WSHDSNO_InputVariable**」→「**header**」→「**ns2:SOAHeader**」にナビゲートして、「**ns2:SecurityGroup**」を選択します。

「式」アイコンをドラッグして、識別したばかりのターゲット・ノード(**ns2:SecurityGroup**)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

7. 「式」アイコンをクリックして「式ビルダー」ダイアログを起動して、4 つ目のパラメータのペ
アを入力します。

- 「式」ボックスに **AMERICAN** と入力します。「**OK**」をクリックします。「割当ての編集」ダイアログの中央の、「元」と「先」のナビゲーション・ツリー・ノードの間に、式の値(**AMERICAN**)を持つ「式」アイコンが表示されます。
- 「先」のナビゲーション・ツリーで「**変数**」タイプを選択します。「**変数**」→「**プロセス**」→「**変数**」→「**Invoke_Shipment_Notice_WSHDSNO_InputVariable**」→「**header**」→「**ns2:SOAHeader**」にナビゲートして、「**ns2:NLSLanguage**」を選択します。

「式」アイコンをドラッグして、識別したばかりのターゲット・ノード(**ns2:NLSLanguage**)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

8. 「式」アイコンをクリックして「式ビルダー」ダイアログを起動して、5 つ目のパラメータのペ
アを入力します。

- 「式」ボックスに **202** と入力します。「**OK**」をクリックします。「割当ての編集」ダイアログの中央の、「元」と「先」のナビゲーション・ツリー・ノードの間に、式の値(**202**)を持つ「式」アイコンが表示されます。
- 「先」のナビゲーション・ツリーで「**変数**」タイプを選択します。「**変数**」→「**プロセス**」→「**変数**」→「**Invoke_Shipment_Notice_WSHDSNO_InputVariable**」→「**header**」→

「ns2:SOAHeader」にナビゲートして、「ns2:Org_Id」を選択します。

「式」アイコンをドラッグして、識別したばかりのターゲット・ノード(ns2:NLSLanguage)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

9. 「編集 Assign」ダイアログ・ボックスが表示されます。

「適用」→「OK」をクリックして、Assign アクティビティの構成を完了します。

受注詳細を **Invoke_ShipmentNoticeInvoke** アクティビティに設定する 2 つ目の Assign アクティビティを追加するには:

1. Oracle JDeveloper で、コンポーネント・パレットから「**BPEL コンストラクト**」を展開します。**Assign** アクティビティを、プロセス・ダイアグラムの中央のスイムレーンの、**Assign** アクティビティと **Invoke** アクティビティの間にドラッグ・アンド・ドロップします。
2. **Assign** アクティビティをダブルクリックして、「編集 Assign」ダイアログ・ボックスにアクセスします。
3. 「一般」タブをクリックして、Assign アクティビティの名前(SetOrderDetails など)を入力します。
4. 「コピー・ルール」タブを選択して、ソース・ツリーとターゲット・ツリーを展開します。
 - 「元」ナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「InvokeReadOrder_SynchRead_OutputVariable」→「body」→「InputParameters」にナビゲートして、「ns1:InputParameters」を選択します。
 - 「先」ナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「InvokeShipmentNotice_WSHDSNO_InputVariable」→「body」にナビゲートして、「ns1: InputParameters」を選択します。

ソース・ノード(ns1:InputParameters)をドラッグして、識別したばかりのターゲット・ノード(ns1: InputParameters)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

5. 「編集 Assign」ダイアログ・ボックスが表示されます。

「適用」→「OK」をクリックして、Assign アクティビティの構成を完了します。

出力する SOAP レスポンスを設定する 3 つ目の Assign アクティビティを追加するには:

1. 3 つ目の Assign アクティビティを追加するには、**Assign** アクティビティを、コンポーネント・パレットの「**BPEL コンストラクト**」セクションからプロセス・ダイアグラムの中央のスイムレーンの、**Invoke** アクティビティと**返信**アクティビティの間にドラッグ・アンド・ドロップします。
2. 1 つ目の Assign アクティビティの作成で説明したステップ 2 とステップ 3 を繰り返して、SetCPdetails という 3 つ目の Assign アクティビティを追加します。
3. 「コピー・ルール」タブを選択し、ターゲット・ツリーを展開します。

- 「元」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「InvokeShipmentNotice_WSHDSNO_OutputVariable」にナビゲートして、「body」を選択します。
- 「先」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「OutputVariable」にナビゲートして、「payload」を選択します。

ソース・ノード(本文)をドラッグして、識別したばかりのターゲット・ノード(ペイロード)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

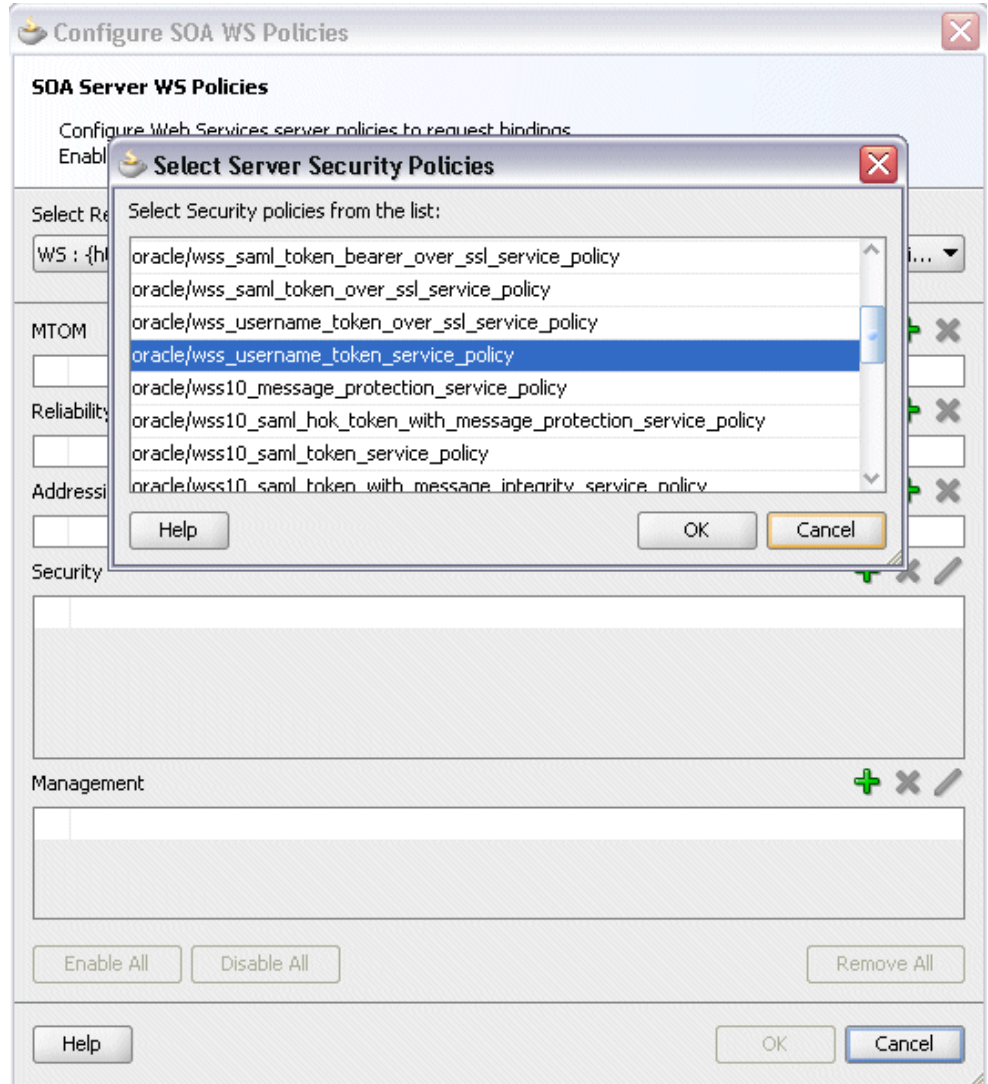
4. 「適用」→「OK」をクリックして、Assign アクティビティの構成を完了します。

Web サービス・ポリシーの構成

次のステップを使用して、設計時のセキュリティ・ポリシーを追加します。

1. 「SOA コンテンツ」→「ビジネス・ルール」→composite.xml に移動します。
Shipment_Notice サービスを右クリックして、ドロップダウン・リストから「WS ポリシーの構成」を選択します。
2. 「SOA WS ポリシーの構成」ダイアログが表示されます。
「セキュリティ」セクションで「セキュリティ・ポリシーの追加」アイコン(+)をクリックします。
サーバー・セキュリティ・ポリシーの選択ダイアログが表示されます。

必要なセキュリティ・ポリシーの選択



「oracle/wss_username_token_service_policy」を選択して、「OK」をクリックします。添付されたセキュリティ・ポリシーが「セキュリティ」セクションに表示されます。

composite.xml の Shipment_Notice サービスにロック・アイコンが表示されます (これは、セキュリティ・ポリシーが正常に添付されたことを示します)。

3. ナビゲーション・メニューから「表示」→「プロパティ・インスペクタ」を選択して、Shipment_Notice サービス・コンポーネントの「プロパティ・インスペクタ」ウィンドウを表示します。

「プロパティ」セクションで、バインディング・プロパティの「セキュリティ・ポリシーの追加」アイコン(+)をクリックします。「プロパティの作成」ダイアログが表示されます。

「名前」フィールドに oracle.webservices.auth.username と入力し、値として operations を入力します。「OK」をクリックします。

4. 「プロパティ」セクションのバインディング・プロパティで「セキュリティ・ポリシーの追加」アイコン(+)を再度クリックする、同じアプローチを使用します。「名前」フィールドに `oracle.webservices.auth.password` と入力します。「値」フィールドに、operations ユーザーに紐づくパスワードを入力します。

「OK」をクリックします。

「プロパティ」セクションに、選択したプロパティの名前と値の両方が表示されます。

`composite.xml` の「ソース」タブをクリックして、
`oracle.webservices.auth.username` プロパティと
`oracle.webservices.auth.password` プロパティの名前および関連する値が、
`Shipment_Notice` 参照に追加されていることを確認します。

プロパティ情報の確認

```
36 <reference name="Shipment_Notice" ui:wsdlLocation="WSHDSNO.wsdl">
37   <interface.wsdl interface="http://xmlns.oracle.com/apps/wsh/soapprovider/concurrentprogram/wshdsno/#wsdl.interf
38   <binding.ws port="http://xmlns.oracle.com/apps/wsh/soapprovider/concurrentprogram/wshdsno/#wsdl.endpoint(WSHDSNO
39     location="WSHDSNO.wsdl" soapVersion="1.1">
40     <wsp:PolicyReference URI="oracle/wss_username_token_client_policy"
41       orawsp:category="security" orawsp:status="enabled"/>
42     <property name="oracle.webservices.auth.username" type="xs:string"
43       many="false" override="may">operations</property>
44     <property name="oracle.webservices.auth.password" type="xs:string"
45       many="false" override="may">password</property>
46   </binding.ws>
47 </reference>
```

実行時における BPEL プロセスによる SOA コンポジットのデプロイおよびテスト

SOA コンポジットに含まれる BPEL クライアントから同期出発出荷通知アウトバウンド WSHDSNO サービスを起動するには、SOA コンポジットを Oracle WebLogic 管理対象サーバーにデプロイする必要があります。これは、Oracle JDeveloper を使用して達成できます。コンポジットがデプロイされたら、Oracle Enterprise Manager Fusion Middleware Control コンソールからテストできます。

前提条件

Oracle JDeveloper を使用して BPEL プロセスによる SOA コンポジットをデプロイする前に、設計時環境とランタイム・サーバーの間の接続を確立する必要があります。必要なサーバー接続を構成する方法の詳細は、「サーバー接続の構成」(ページ B-1)を参照してください。

注意: WebLogic サーバーのローカル・インスタンスが使用される場合、Oracle JDeveloper から「実行」→「サーバー・インスタンスの起動」を選択して、WebLogic サーバーを起動します。WebLogic 管理サーバーの DefaultServer インスタンスが正常に起動されたら、「実行中:DefaultServer」と「メッセージ」のログに<Server started in Running mode>と DefaultServer started のメッセージが表示されるはずです。

次の実行時タスクを実行します。

1. BPEL プロセスによる SOA コンポジット・アプリケーションのデプロイ、ページ 7-23
2. BPEL プロセスによる SOA コンポジット・アプリケーションのテスト、ページ 7-26

BPEL プロセスによる SOA コンポジット・アプリケーションのデプロイ

これまでに作成した BPEL プロセス(ShipNotice. bpel)による SOA コンポジット・アプリケーションを実行する前に、デプロイする必要があります。

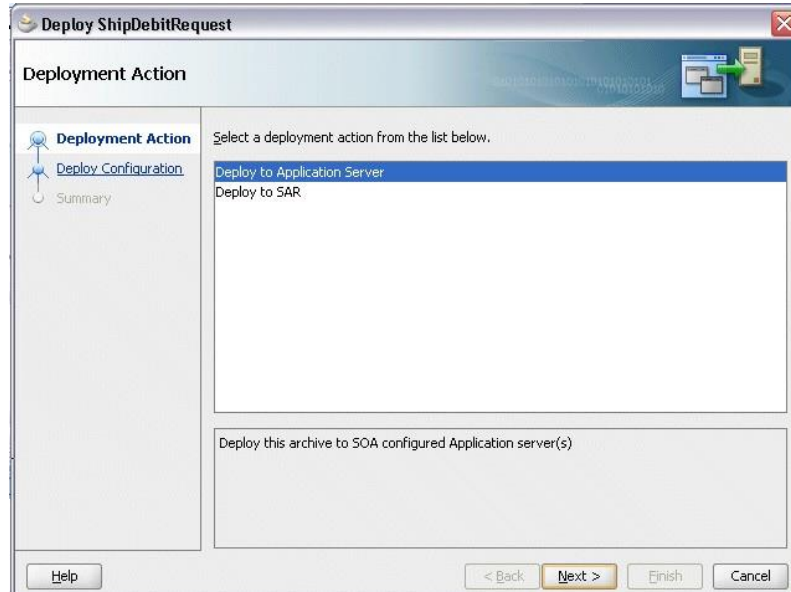
SOA コンポジット・アプリケーション BPEL プロセスをデプロイするには:

1. JDeveloper のアプリケーション・ナビゲータで、**ShipNotice** プロジェクトを選択します。
2. プロジェクトを右クリックして、メニューから「デプロイ」→「[project name]」→「[serverConnection]」を選択します。

たとえば、接続が適切に設定されている場合は、「デプロイ」→「**ShipNotice**」→「**SOAServer**」を選択してプロセスをデプロイできます。

注意: サーバー接続を初めて設定する場合は、「デプロイメント・アクション」ダイアログが表示されます。「アプリケーション・サーバーにデプロイ」を選択して、「次へ」をクリックします。

「デプロイメント・アクション」ダイアログ



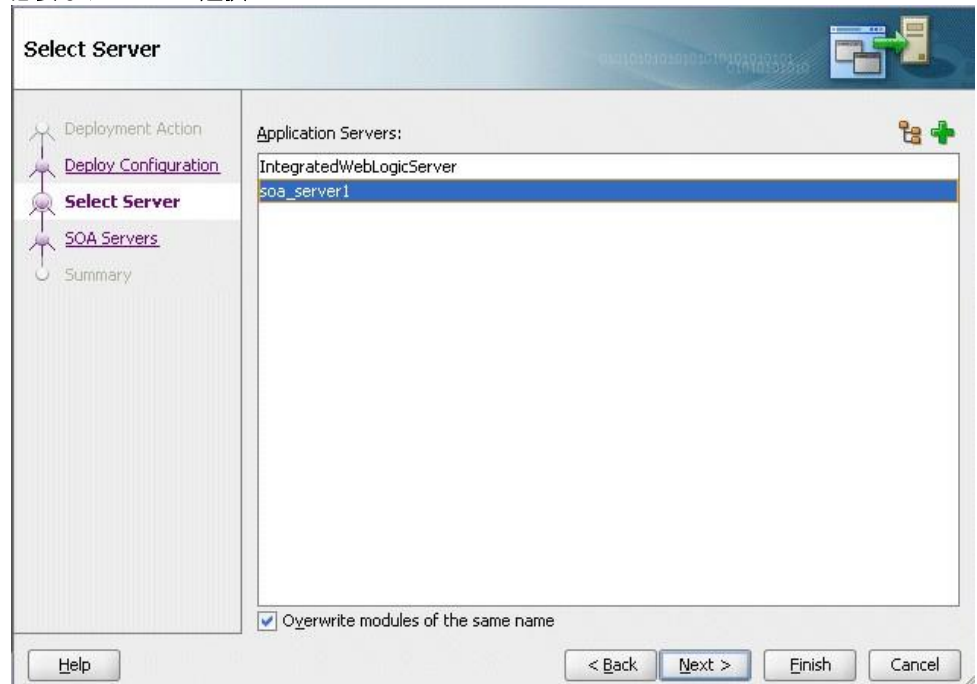
「構成のデプロイ」ダイアログで、「次へ」をクリックして新しいアプリケーション・サーバーを追加する前に、次の情報が選択されるようにします。

- 新規リビジョン ID: 1.0
- コンポジット・リビジョンをデフォルトとしてマークします。: このチェック・ボックスを選択します。
- 同じリビジョン ID で既存のコンポジットを上書きします。: このチェック・ボックスを選択します。

JDeveloper から新しい Oracle WebLogic Server 接続を作成するステップは、「サーバー接続の構成」(ページ B- 1)を参照してください。

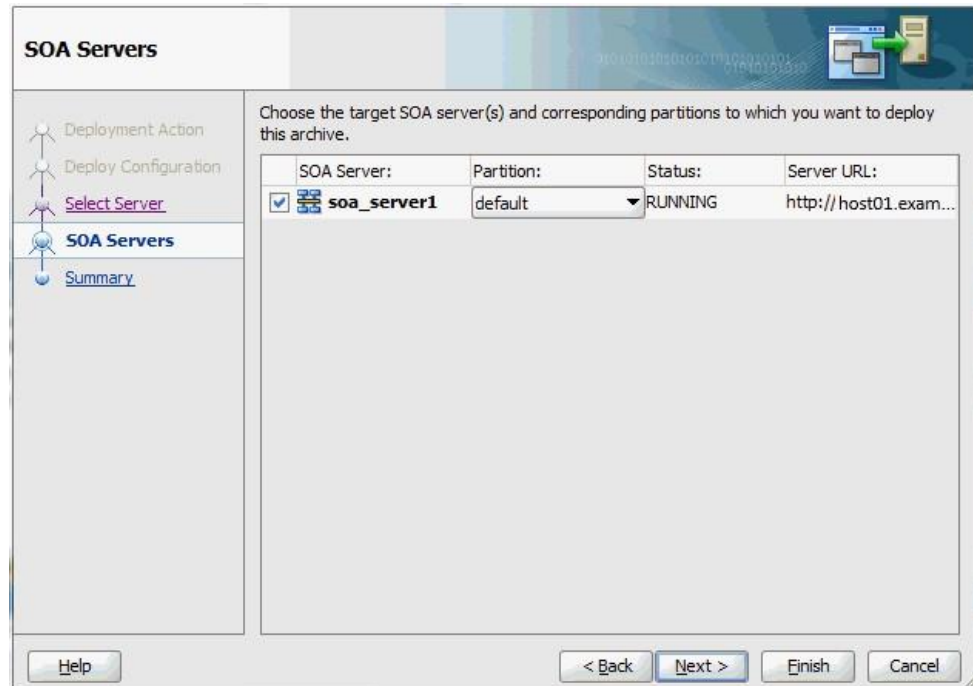
2. 「サーバーの選択」ダイアログで、以前にサーバー接続を確立した soa-server1 を選択します。「次へ」をクリックします。

必要なサーバーの選択



3. 「SOA サーバー」ダイアログで、デフォルトのターゲット SOA サーバー(soa-server1)の選択を受け入れます。

必要な SOA サーバーの選択



「次へ」→「終了」をクリックします。

5. Oracle JDeveloper セッションから初めてコンポジットをデプロイする場合は、「認証リクエスト」ウィンドウが表示されます。Oracle SOA Suite のインストール時に指定したユーザー名とパスワード情報を入力します。「OK」をクリックします。
6. デプロイ処理が開始します。SOA - ログ・ウィンドウで、デプロイ・プロセスをモニターして、コンパイルが成功したことを確認します。

デプロイメント - ログ・ウィンドウで、デプロイが成功したことを確認します。

BPEL プロセスによる SOA コンポジット・アプリケーションのテスト

SOA コンポジット・アプリケーションに含まれる BPEL プロセスが正常にデプロイされたら、Oracle Enterprise Manager Fusion Middleware Control コンソールからプロセスを管理およびモニターできます。

Oracle E-Business Suite にログインしてプロセスを手動で開始し、出発出荷通知アウトバウンド(DSNO)が XML ファイルに生成されることを確認することもできます。

Oracle SOA Suite の詳細は、*Oracle Fusion Middleware Oracle SOA Suite 管理者ガイド*を参照してください。

BPEL プロセスによる SOA コンポジット・アプリケーションをテストするには:

1. Oracle Enterprise Manager Fusion Middleware Control コンソール
(<http://<hostname>:<port>/em>)にナビゲートします。ログイン・ページが表示されます。
2. Oracle SOA Suite のインストール時に指定したユーザー名とパスワードを入力します。
「ログイン」をクリックして、ファームにログインします。デプロイしたコンポジット (ShipNotice) がアプリケーション・ナビゲーション・ツリーに表示されます。

複数のターゲット Oracle Enterprise Manager Fusion Middleware Control コンソール・ファームがある場合、適切なターゲット・インスタンス・ファームを選択する必要がある場合があります。
3. 「ファーム」のナビゲーション・ペインから、ツリーの「SOA」→「soa-infra」ノードを展開して、「SOA インフラストラクチャ」のホーム・ページおよびメニューを使用して、soa-infra 管理対象サーバーで実行されているデプロイ済の SOA コンポジット・アプリケーションにアクセスします。

ShipNotice [1.0]のリンクをクリックします。
4. 「ポリシー」タブをクリックして、設計時に Shipment_Notice サービス・バインディングに添付した oracle/wss_username_token_service_policy ポリシーがここに表示されていることを確認します。
5. ShipNotice [1.0]のホーム・ページで、「テスト」をクリックします。
6. インスタンスを開始するための「Web サービスのテスト」ページが表示されます。「引数の入力」セクションには、使用する XML ペイロード・データの情報を指定できます。

注意: 設計時に WS-Security 資格証明が入力されない場合、「リクエスト」タブの上部にある「セキュリティ」セクションの「WSS ユーザー名トークン」オプションを選択して、実行時に資格証明を入力できます。「ユーザー名」フィールドに operations、「パスワード」フィールドに、operations ユーザーに紐づくパスワードを入力します。

「Web サービスのテスト」をクリックして、プロセスを開始します。

完了時に、「レスポンス」タブにテスト結果が表示されます。

7. BPEL サービス・コンポーネント・インスタンスのリンク(ShipNotice など)をクリックして、「インスタンス」ページを表示し、その「監査証跡」タブで BPEL アクティビティの実行の詳細を確認できます。

「フロー」タブをクリックして、BPEL プロセスのフロー・ダイアグラムを確認します。プロセス・ダイアグラムのアクティビティをクリックして、アクティビティの詳細と、プロセスを介したペイロードのフローを表示します。
8. プロセス・フロー・チャートの「Invoke_ShipmentNotice」アイコンをダブルクリックし、「XML 文書の表示」リンクをクリックして XML ファイルを開きます。このファイルには、トランザクション用に返されるリクエスト ID が記録されます。

Oracle E-Business Suite でのレコードの検証

Oracle E-Business Suite のレコードを検証する場合は、その前にまずコンカレント要求が正常に完了していることを確認する必要があります。

1. 「システム管理者」職責を持つユーザーとして Oracle E-Business Suite にログインします。「要求」→「表示」を選択して、「要求の検索」ウィンドウを開きます。
2. 監査証跡から取得した要求 ID を入力してコンカレント要求を検索し、「検索」をクリックします。
3. 要求の詳細ページが表示されます。要求のフェーズとステータスをチェックして、要求のフェーズが「完了」であるかを確認します。

コンカレント要求が正常に完了したら、Oracle E-Business Suite で検証できます。

DSNO (出荷出荷通知アウトバウンド)はアウトバウンド XML メッセージであるため、該当する受信者に出荷通知が送信されるように、関連する XML ゲートウェイの設定タスクを適切に構成する必要があります。

詳細は、*Oracle XML Gateway ユーザーズ・ガイド*を参照してください。

DSNO XML ファイル内の出荷先所在地、発注および要求出荷日の情報が受注情報と一致するかどうかを検証できます。

オープン・インタフェース REST サービスの使用

概要

コンカレント・プログラム REST サービスと同様に、オープン・インタフェース表およびビューを REST サービスとして配置できるようになりました。これにより、配置したオープン・インタフェース REST サービスを活用して、Oracle E-Business Suite のベース表に対してデータを直接更新またはインポートすることや、ベース表のデータを選択することが可能になります。

注意: オープン・インタフェース表およびビューは、REST サービスとしてのみ提供できます。カスタム・オープン・インタフェースは、このリリースではサポートされていません。

オープン・インタフェース表またはビューを探すには、「検索」ページでインタフェース・タイプとして「オープン・インタフェース」または「インタフェースの表示」を選択し、必要に応じて追加の検索基準を入力して、目的のインタフェースを取得します。その後、統合管理者は、サポートされている HTTP メソッドとともに REST サービスとしてそれを配置できます。

サービス起動の例

オープン・インタフェース REST サービスを使用して Oracle E-Business Suite のベース表を変更する方法について理解を深めるために、この章には、オープン・インタフェース表 REST サービスの使用を通じてデータを挿入、更新、フェッチおよび削除する HTTP メソッドの使用例が含まれています。

オープン・インタフェース表 REST サービスの使用

REST サービスの起動シナリオ

オープン・インタフェース表 REST サービスをテストおよび使用する方法を説明するために、この例ではオープン・インタフェース表 AR Autoinvoice (RAXMTR)を使用します。

この例では、Oracle Receivables に対して請求書データを挿入、取得および削除するいくつかの HTTP 要求を受け取ります。これらの要求を実行するために、AR Autoinvoice インタフェースに含まれている RA_INTERFACE_LINES_ALL REST 操作が起動されます。

サービスが正常に起動されるたびに、クライアントは、Oracle Receivables に対して請求書明細データが正常に挿入、更新、取得または削除されたことを示す REST 応答メッセージを受け取ります。

オープン・インタフェース REST サービスの起動

REST サービスの起動シナリオに基づいて、この章では次のトピックについて説明します。

1. REST サービスの配置、ページ 8-2
2. セキュリティ付与の作成、ページ 8-3
3. 配置済 WADL からのリソース情報の記録、ページ 8-4
4. コマンドラインを使用した REST サービスの起動、ページ 8-4

REST サービスの配置

次のステップを使用して、オープン・インタフェース表 AR Autoinvoice (RAXMTR)を配置します。

1. 「統合管理者」ロールを持つユーザーとして、Oracle E-Business Suite にログインします。

職責「統合 SOA ゲートウェイ」を使って、ナビゲーション・メニューから「統合リポジトリ」リンクを選択します。
2. 「統合リポジトリ」タブで、「**検索**」をクリックしてメインの「検索」ページにアクセスします。
3. 検索基準として、次のキー検索値を入力します。
 - 内部名: RAXMTR
 - インタフェース・タイプ: Open Interface
4. 「**進む**」をクリックして、検索を実行します。

AR Autoinvoice のインタフェース名リンクをクリックして、インタフェースの詳細ページを開きます。
5. 「REST Web サービス」タブで、次の情報を入力します。

REST サービスとしてのオープン・インタフェース表の配置

Integration Repository Administration

Integration Repository >

Open Interface : AR Autoinvoice Browse Search Printable Page

Internal Name RAXMTR Status Active
 Type Concurrent Program and Open Interface Scope Public
 Product Receivables
 Business Entities [Credit Memo](#), [Debit Memo](#), [Receivables Invoice](#)
 Online Help [Importing Transaction Information Using AutoInvoice](#), [Oracle Receivables Help](#)

Overview REST Web Service Grants

* Service Alias autoinvoice x
 REST Service Status Not Deployed

Service Operations

Name	Direction	Get	Post	Put	Delete	Grant
AR Autoinvoice		☒	☒	☒	☒	
RA INTERFACE LINES ALL	Inbound	☒	☒	☒	☒	
RA INTERFACE SALES CREDITS ALL	Inbound	☒	☒	☒	☒	
RA INTERFACE DISTRIBUTIONS ALL	Inbound	☒	☒	☒	☒	
RA INTERFACE ERRORS ALL	Inbound	☒	☒	☒	☒	
SUBMIT_CP_RAXMTR		☐	☒	☐	☐	

☒ TIP To apply any changes in Operation, Undeploy the service.

REST Service Security

REST Web Service is secured by HTTP Basic Authentication at HTTP Transport level. Send either of the following in "Authorization" header as per HTTP Basic scheme:
 - Username:Password
 - Security Token.

Tip: [Login](#) to obtain Security Token for given user
 Use [Service](#) credentials.

Deploy

- サービス別名: autoinvoice

この別名が、WADL のサービス・エンドポイントと、選択されたメソッドまたはオペレーションのスキーマとして利用されます。

- 必要なサービス操作の選択

最初の行「AR Autoinvoice」で、4 つの HTTP メソッドのチェック・ボックス(「GET」、「POST」、「PUT」および「DELETE」)を選択します。表内にリストされている、メソッド名として示されている残りのオープン・インタフェース表名は、すべて自動的に選択されます。

6. 「配置」をクリックして、Oracle E-Business Suite WebLogic 環境にサービスを配置します。

REST サービスが正常に配置されたら、「WADL の表示」リンクとともに、「REST サービス・ステータス」フィールドに「配置済」が表示されます。「WADL の表示」リンクをクリックして、配置されたサービス WADL の記述を表示します。

REST サービスの配置の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「REST Web サービスの配置」を参照してください。

セキュリティ付与の作成

AR Autoinvoice を REST サービスとして配置したら、統合管理者はセキュリティ付与を作成して、サービスまたはメソッドのアクセス権限を特定のユーザー、ユーザー・グループまたは

すべてのユーザーに与えることができます。

次のステップを使用して、セキュリティ付与を作成します。

1. 「統合管理者」ロールを持つユーザーとして、Oracle E-Business Suite にログインします。職責「統合 SOA ゲートウェイ」を使って、ナビゲーション・メニューから「統合リポジット」リンクを選択します。
2. 検索を実行して、管理者が先ほど配置した AR Autoinvoice サービスを探します。
3. 選択した AR Autoinvoice のインタフェースの詳細ページで、「付与」タブをクリックします。
4. RA_INTERFACE_LINES_ALL メソッドのチェック・ボックスを選択し、「付与の作成」をクリックします。
5. 「付与の作成」ページで、「被付与者タイプ」として「全ユーザー」を選択します。

注意: この例では、RA_INTERFACE_LINES_ALL サービス操作はすべてのユーザーに許可されます。実際の実装では、厳格なセキュリティ・ルールを定義する必要があります。ユーザーに対する付与、またはより適切な方法として、ロールおよび職責によって定義されたユーザーのグループに対する付与を作成します。

「付与の作成」をクリックします。これにより、すべての Oracle E-Business Suite ユーザーに、選択したメソッドのアクセス権限が付与されます。

配置済 WADL からのリソース情報の記録

配置した AR Autoinvoice サービスからサービス・リソース情報を取得するには、統合開発者が「REST Web サービス」タブで「**WADL の表示**」リンクをクリックします。WADL の記述が別のウィンドウに表示されます。

WADL の記述から REST サービス・エンドポイントをコピーまたは記録します。これは、後でサービスを起動する際に使用されます。

`http://<hostname>:<port>/webservices/rest/autoinvoice/RA_INTERFACE_LINES_ALL/`

autoinvoice (このサンプル配置における別名)は、自分の配置で使用した別名に置き換えてください。

コマンドラインを使用した REST サービスの起動

この例では、cURL というサード・パーティのコマンドライン・ツールを使用し、URL 構文を使用してデータを転送します。このツールは、ほとんどの Linux 環境でデフォルトで使用可能です。

注意: 任意の REST クライアントを使用して、REST サービスの起動をテストできます。

この項には、次の起動シナリオが含まれています。

1. HTTP POST メソッドを使用した Oracle Receivables へのデータの挿入、ページ 8-5
2. HTTP PUT メソッドを使用した Oracle Receivables でのデータの更新、ページ 8-7
3. HTTP GET メソッドを使用した Oracle Receivables からのデータのフェッチ、ページ 8-8
4. HTTP DELETE メソッドを使用した Oracle Receivables でのデータの削除、ページ 8-9

HTTP POST メソッドを使用した Oracle Receivables へのデータの挿入

入力ペイロードを含むファイルの作成

Oracle Receivables に請求書データを挿入する HTTP 要求を受け取る場合、REST サービスを起動する前に、必要なデータを入力ペイロードとして含む `lines.xml` という次の xml ファイルを作成する必要があります。

```

<?xml version="1.0" encoding="UTF-8"?>
<RA_INTERFACE_LINES_ALL_Input>
  <RESTHeader>
    <Responsibility>RECEIVABLES_VISION_OPERATIONS</Responsibility>
    <RespApplication>AR</RespApplication>
    <SecurityGroupE>STANDARD</SecurityGroup>
    <NLSLanguage>AMERICAN</NLSLanguage>
    <Org_Id>204</Org_Id>
  </RESTHeader>
  <Select>INTERFACE_LINE_ID, BATCH_SOURCE_NAME</Select>
  <InputParameters>
    <RA_INTERFACE_LINES_ALL_REC>
      <INTERFACE_LINE_ATTRIBUTE9>1</INTERFACE_LINE_ATTRIBUTE9>
      <INTERFACE_LINE_ATTRIBUTE11>1</INTERFACE_LINE_ATTRIBUTE11>
      <INTERFACE_LINE_ATTRIBUTE10>1</INTERFACE_LINE_ATTRIBUTE10>
      <ORG_ID>204</ORG_ID>
      <COMMENTS>Test REST1</COMMENTS>
      <QUANTITY>1</QUANTITY>
      <TRX_NUMBER>trxpj1</TRX_NUMBER>
      <CONVERSION_RATE>1</CONVERSION_RATE>
      <CONVERSION_DATE>11-APR-2018</CONVERSION_DATE>
      <CONVERSION_TYPE>Use</CONVERSION_TYPE>
      <ORIG_SYSTEM_SHIP_ADDRESS_ID>1030</ORIG_SYSTEM_SHIP_ADDRESS_ID>
      <ORIG_SYSTEM_SHIP_CUSTOMER_ID>1004</ORIG_SYSTEM_SHIP_CUSTOMER_ID>
      <ORIG_SYSTEM_BILL_ADDRESS_ID>1030</ORIG_SYSTEM_BILL_ADDRESS_ID>
      <ORIG_SYSTEM_BILL_CUSTOMER_ID>1004</ORIG_SYSTEM_BILL_CUSTOMER_ID>
      <TERM_ID>4</TERM_ID>
      <TERM_NAME>30 Net</TERM_NAME>
      <CUST_TRX_TYPE_ID>1</CUST_TRX_TYPE_ID>
      <CUST_TRX_TYPE_NAME>Invoice</CUST_TRX_TYPE_NAME>
      <AMOUNT>1000.00</AMOUNT>
      <CURRENCY_CODE>USD</CURRENCY_CODE>
      <DESCRIPTION>Project Invoices</DESCRIPTION>
      <LINE_TYPE>LINE</LINE_TYPE>
      <SET_OF_BOOKS_ID>1</SET_OF_BOOKS_ID>
      <BATCH_SOURCE_NAME>PROJECTS INVOICES</BATCH_SOURCE_NAME>
      <INTERFACE_LINE_ATTRIBUTE7>Line</INTERFACE_LINE_ATTRIBUTE7>
      <INTERFACE_LINE_ATTRIBUTE6>1</INTERFACE_LINE_ATTRIBUTE6>
      <INTERFACE_LINE_ATTRIBUTE5>Sole, Mr.
Samuel</INTERFACE_LINE_ATTRIBUTE5>
      <INTERFACE_LINE_ATTRIBUTE4>Vision
Operations</INTERFACE_LINE_ATTRIBUTE4>
      <INTERFACE_LINE_ATTRIBUTE3>Services 01</INTERFACE_LINE_ATTRIBUTE3>
      <INTERFACE_LINE_ATTRIBUTE2>3</INTERFACE_LINE_ATTRIBUTE2>
      <INTERFACE_LINE_ATTRIBUTE1>ATZ
Services</INTERFACE_LINE_ATTRIBUTE1>
      <INTERFACE_LINE_CONTEXT>PROJECTS INVOICES</INTERFACE_LINE_CONTEXT>
    </RA_INTERFACE_LINES_ALL_REC>
  </InputParameters>
</RA_INTERFACE_LINES_ALL_Input>

```

コマンドライン・ツールを使用した REST サービスの起動

xml ファイルを作成したら、起動をテストするディレクトリに置きます。次のコマンドを実行してサービスを起動し、要求から送信された入力ペイロードに基づいて、それぞれの請求書明細データを Oracle Receivables に転送します。

```

curl -H 'Content-Type: application/xml' -u <username>:<password> -d
@lines.xml
https://<ebshost>:
<port>/webservices/rest/<alias>/RA_INTERFACE_LINES_ALL/

```

- <username>は、オープン・インタフェース表 AR Autoinvoice へのアクセス権を持つ Oracle E-Business Suite ユーザーの名前に置き換えてください。

- <alias>は、この例では autoinvoice に置き換えてください。

応答メッセージの表示

REST サービスが正常に起動されると、XML 形式で次の出力が表示されます。

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<OutputParameters xmlns="http://xmlns.oracle.
com/apps/ar/rest/autoinvoice/ra_interface_lines_all">
  <Summary>
    <SuccessCount>1</SuccessCount>
    <ErrorCount>0</ErrorCount>
  </Summary>
  <Result>
    <Output>
      <Index>0</Index>
      <Status>SUCCESS</Status>
      <Message></Message>
      <RA_INTERFACE_LINES_ALL_REC>
        <INTERFACE_LINE_ID/>
        <BATCH_SOURCE_NAME>PROJECTS INVOICES</BATCH_SOURCE_NAME>
      </RA_INTERFACE_LINES_ALL_REC>
    </Output>
  </Result>
</OutputParameters>
```

HTTP PUT メソッドを使用した Oracle Receivables でのデータの更新

Oracle Receivables でのデータの更新要求を受け取る場合、PUT メソッドを使用して、条件に基づいてデータを更新できます。

たとえば、TRX_NUMBER が Demo1 であり、ORG_ID が 204 であるレコードについて RA_INTERFACE_LINES_ALL で QUANTITY と COMMENTS の値を更新する必要があります。

入力ペイロードを含むファイルの作成

この場合は、まず、ファイル linesUpdate.xml を作成し、次の内容をペイロードとして含める必要があります。

```
<?xml version="1.0" encoding="UTF-8"?>
<RA_INTERFACE_LINES_ALL_Input>
  <RESTHeader>
    <Responsibility>RECEIVABLES_VISION_OPERATIONS</Responsibility>
    <RespApplication>AR</RespApplication>
    <SecurityGroupE>STANDARD</SecurityGroup>
    <NLSLanguage>AMERICAN</NLSLanguage>
    <Org_Id>204</Org_Id>
  </RESTHeader>
  <Select>DESCRIPTION,TRX_NUMBER</Select>
  <InputParameters>
    <Update>
      <Filter>ORG_ID==204;TRX_NUMBER==Demo1</Filter>
      <RA_INTERFACE_LINES_ALL_REC>
        <COMMENTS>Quantity updated to 5</COMMENTS>
        <QUANTITY>5</QUANTITY>
      </RA_INTERFACE_LINES_ALL_REC>
    </Update>
  </InputParameters>
</RA_INTERFACE_LINES_ALL_Input>
```

コマンドライン・ツールを使用した REST サービスの起動

次の curl コマンドを実行して、TRX_NUMBER が Demo1 であり、ORG_ID が 204 であるレコードについて RA_INTERFACE_LINES_ALL で QUANTITY と COMMENTS を更新します。

```
curl -H 'Content-Type: application/xml' -u <username>:<password> -X PUT
-d @ linesUpdate.xml
'https://<ebshost>:
<port>/webservices/rest/<alias>/RA_INTERFACE_LINES_ALL/'
```

- <username>は、オープン・インタフェース表 AR Autoinvoice にアクセスするための付与を持つ Oracle E-Business Suite ユーザーの名前に置き換えてください。
- <alias>は、この例では autoinvoice に置き換えてください。

応答メッセージの表示

REST サービスが正常に起動されると、XML 形式で次の出力が表示されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<OutputParameters xmlns="http://xmlns.oracle.com/apps/ar/rest/autoinvoice/ra_interface_lines_all">
  <Summary>
    <SuccessCount>1</SuccessCount>
    <ErrorCount>0</ErrorCount>
  </Summary>
  <Result>
    <Output>
      <Index>1</Index>
      <Status>SUCCESS</Status>
      <Message/>
      <UpdateCount>1</UpdateCount>
      <RA_INTERFACE_LINES_ALL_REC>
        <DESCRIPTION>Project Invoices</DESCRIPTION>
        <TRX_NUMBER>Demo1</TRX_NUMBER>
      </RA_INTERFACE_LINES_ALL_REC>
    </Output>
  </Result>
</OutputParameters>
```

HTTP GET メソッドを使用した Oracle Receivables からのデータのフェッチ

Oracle Receivables からのデータのフェッチ要求を受け取る場合、GET メソッドを使用して、条件に基づいてデータをフェッチできます。

コマンドライン・ツールを使用した REST サービスの起動

たとえば、次の curl コマンドを実行して、ORG_ID 204 に属する、TRX_NUMBER が Demo1 のレコードについて RA_INTERFACE_LINES_ALL から TRX_NUMBER、QUANTITY および COMMENTS の値をフェッチします。

```
curl -u <username>:<password> -X GET
'https://<ebshost>:
<port>/webservices/rest/<alias>/RA_INTERFACE_LINES_ALL/?
filter=ORG_ID==204;TRX_NUMBER==Demo1&select=TRX_NUMBER,QUANTITY,
COMMENTS'
```

- <username>は、オープン・インタフェース表 AR Autoinvoice へのアクセス権を持つ Oracle E-Business Suite ユーザーの名前に置き換えてください。
- <alias>は、この例では autoinvoice に置き換えてください。

応答メッセージの表示

REST サービスが正常に起動されると、次のような応答が返されます。

```
{
  "OutputParameters" : {
    "@xmlns" : "http://xmlns.oracle.com/apps/ar/concurrentprogram/rest/autoinvoice/ra_interface_lines_all",
    "Summary" : {
      "Select" : "TRX_NUMBER,QUANTITY,COMMENTS",
      "Filter" : "ORG_ID==204;TRX_NUMBER==Demo1",
      "Offset" : "0",
      "Limit" : "200",
      "Sort" : null,
      "GetCount" : "1",
      "TotalCount" : "1"
    },
    "Result" : {
      "Output" : {
        "RA_INTERFACE_LINES_ALL_REC" : {
          "TRX_NUMBER" : "Demo1",
          "QUANTITY" : "1",
          "COMMENTS" : "Create"
        }
      }
    }
  }
}
```

HTTP DELETE メソッドを使用した Oracle Receivables でのデータの削除

Oracle Receivables からのデータの削除要求を受け取る場合、DELETE メソッドを使用してデータを削除できます。

コマンドライン・ツールを使用した REST サービスの起動

たとえば、次の curl コマンドを実行して、204 の ORG_ID に属する、TRX_NUMBER が Demo1 のデータを RA_INTERFACE_LINES_ALL から削除します。

```
curl -u <username>:<password> -X DELETE
'https://<ebshost>:
<port>/webservices/rest/<alias>/RA_INTERFACE_LINES_ALL/?
filter=ORG_ID==204;TRX_NUMBER==Demo1'
```

- <username>は、オープン・インタフェース表 AR Autoinvoice へのアクセス権を持つ Oracle E-Business Suite ユーザーの名前に置き換えてください。
- <alias>は、この例では autoinvoice に置き換えてください。

応答メッセージの表示

REST サービスが正常に起動されると、次のような応答メッセージが返されます。

```
{
  "OutputParameters" : {
    "@xmlns" : "http://xmlns.oracle.
com/apps/ar/concurrentprogram/rest/autoinvoice/ra_interface_lines_all",
    "Result" : {
      "Status" : "SUCCESS",
      "Message" : null,
      "DeleteCount" : "1"
    }
  }
}
```

ビジネス・サービス・オブジェクトの使用

概要

ビジネス・サービス・オブジェクト(以前はサービス Bean と呼ばれていました)は、OA フレームワークまたは BC4J (Business Components for Java)コンポーネントを Web サービスとしてデプロイできるようにする、高レベルのサービス・コンポーネントです。Oracle E-Business Suite では、このツールによって、サービス指向アーキテクチャ(SOA)および Web サービスを採用し、相互の、およびサード・パーティの取引先との統合を容易にしています。

ビジネス・サービス・オブジェクト・インタフェースにより、SOA サービスにアクセスでき、Oracle E-Business Suite と取引先の間の統合が容易になります。多くの場合、複雑なデータを渡すためのパラメータとして、サービス・データ・オブジェクトが採用されます。

幅広い顧客がビジネス・サービス・オブジェクトをより有効に利用できるように、統合管理者または統合開発者がまず Web サービスを生成し、その後、管理者がこれを Oracle SOA Suite WebLogic 管理対象サーバーにデプロイできます。Oracle SOA Suite コンポーネントを活用することにより、デプロイ済サービスを、サービス・エンドポイントを持つ SOA コンポジット・アプリケーションとして、有用な BPEL プロセスにオーケストレーションできます。

このプロセスでは、ビジネス・パートナーからデータを取得して、必要に応じて Oracle E-Business Suite を挿入または更新できます。

実行時に、soa-infra アプリケーションが実行されている WebLogic 管理対象サーバーの SOA コンポジット・アプリケーションを顧客に公開し、Web サービス・クライアントまたはオーケストレーション・ツール(Oracle JDeveloper、Apache Axis、.NET Web サービス・クライアント、Oracle BPEL Process Manager、Oracle Enterprise Service Bus (ESB)など)によって起動できます。

アプリケーション・データの挿入または更新で各 Web サービスをどのように使用できるかについての理解を深めるために、この章では、設計時タスクおよび実行時タスクの詳細について説明します。この後の各項で説明する例では、Oracle JDeveloper 11g (11.1.1.6.0)を設計時ツールとして使用して BPEL プロセスを作成し、Oracle SOA Suite 11g (11.1.1.6.0)を使用してこのプロセスをデプロイします。

注意: Oracle JDeveloper とその他の Oracle Fusion Middleware コンポーネント (Oracle SOA Suite など) を使用している場合、SOA テクノロジを有効にするには、SOA テクノロジ用の JDeveloper の拡張機能である Oracle SOA Suite コンポジット・エディタを手動でダウンロードする必要があります。追加の Oracle Fusion Middleware 設計時コンポーネントのインストールの詳細は、*Oracle Fusion Middleware Oracle JDeveloper インストレーション・ガイド* を参照してください。

設計時におけるビジネス・サービス・オブジェクト WSDL の使用

BPEL プロセスによる SOA コンポジット・アプリケーションのシナリオ

この例では、PurchaseOrderService

/oracle/apps/fnd/framework/toolbox/tutorial/PurchaseOrderService ビジネス・サービス・オブジェクト・インタフェースを使用して、BPEL プロセスの作成について説明します。

発注承認要求を受け取ると、発注詳細の情報がファイル・アダプタによって読み取られます。次に、受注データが PurchaseOrderService 内の approvePurchaseOrder メソッドに渡され、単一発注承認プロセスが開始されます。次に、承認情報が要求者に返信されます。

デプロイ後に BPEL プロセスが正常に実行されると、発注ステータスが Incomplete から Approved に変わります。

ビジネス・サービス・オブジェクト Web サービスを使用して BPEL プロセスによる SOA コンポジット・アプリケーションを作成するための前提条件

- 統合管理者または統合開発者は、事前に Web サービスを生成しておく必要があります。管理者が、Oracle SOA Suite WebLogic 管理対象サーバーに、生成されたサービスをデプロイします。
- 統合開発者は、Web サービスとして公開されている BSO インタフェースのデプロイ済 WSDL URL を見つけて記録しておく必要があります。
- Web サービスの承認のために、ヘッダー変数を設定する必要があります。

ビジネス・サービス・オブジェクト Web サービス・コンポジットのデプロイ

統合管理者または統合開発者は、まず選択したインタフェース定義の Web サービスを作成する必要があり、その後管理者が Oracle Integration Repository から Oracle SOA Suite WebLogic 管理対象サーバーにサービスをデプロイできます。

たとえば、統合開発者がデプロイ済 WSDL を使用して BPEL プロセスを作成する前に、まず次のステップを実行する必要があります。

1. Web サービスを生成するには、統合管理者または統合開発者が、まず Oracle Integration Repository からビジネス・サービス・オブジェクト・インタフェース定義

(PurchaseOrderService

/oracle/apps/fnd/framework/toolbox/tutorial/PurchaseOrderService)を見つけてます。インタフェースの詳細ページの「Web サービス」リージョンで、インタフェースの同期相互作用パターンを選択するか、「相互作用パターン」表から目的のメソッドを選択します。「生成」をクリックして、サービスを生成します。

サービスが正常に生成されると、「Web サービス・ステータス」フィールドが「未生成」から「生成済」に変わります。

Web サービスの生成方法の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「Web サービスの生成」を参照してください。

2. 生成された Web サービスを配置するには、「配置」をクリックする前に、管理者が 1 つの認証タイプを選択します。Oracle SOA Suite に配置されたサービスは、有効なサービスであり、新しい SOAP 要求を受け入れる準備が整っています。

サービスが正常に配置されると、選択した認証タイプが表示され、「Web サービス・ステータス」フィールドが「配置済」および「有効」状態になります。認証タイプによる Web サービスの保護の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「Web サービス・セキュリティの管理」を参照してください。

Web サービスの配置方法の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「Web サービスの配置および配置解除」を参照してください。

WSDL URL の検索および記録

管理者が実行する必要があるタスクとは別に、統合開発者は、Oracle JDeveloper で有用な BPEL プロセスにオーケストレーションする必要があるインタフェースのデプロイ済 Web サービスの WSDL URL を見つけて記録する必要があります。

この WSDL 情報は、後で、設計時に BPEL プロセスを作成するとき、Web サービスとして公開されるインタフェースのパートナー・リンクの作成に使用されます。

デプロイ済 WSDL URL の確認と記録

The screenshot shows the 'Integration Repository' page for the 'Business Service Object : Purchase Order Service'. It includes a 'Full Description' section and a 'Web Service' section. In the 'Web Service' section, there is a table with columns 'Display Name', 'Internal Name', and 'Synchronous'. The 'Purchase Order Service' is listed with its internal name and a checked 'Synchronous' box. Below the table, there is a 'View WSDL' link highlighted with a red box. The 'Web Service Status' is 'Deployed | Active' and the 'Interaction Pattern' is 'Synchronous'. The 'Authentication Type' is 'Username Token'.

インタフェースを検索してインタフェースの詳細をレビューする方法の詳細は、「統合インタフェースの検索および表示」(ページ 2-1)を参照してください。

SOAP 要求のヘッダー変数の設定

サービスの実行中に、アプリケーション・コンテキストを設定するために要求される値を渡すには、ヘッダー要素に対する BPEL プロセスの特定の変数を設定する必要があります。これらのビジネス・サービス・オブジェクト・インタフェース・タイプのヘッダー要素は、

RESPONSIBILITY_NAME, *RESPONSIBILITY_APPL_NAME*,
SECURITY_GROUP_NAME, *NLS_LANGUAGE* および *ORG_ID* です。

注意: ユーザー名とパスワード情報は、Web サービス・セキュリティ・ポリシー (oracle/wss_username_token_service_policy など)によって定義されます。SOAP 要求とともにセキュリティ・ヘッダーを渡す方法の詳細な手順は、「Web サービス・ポリシーの構成」(ページ 9-24)を参照してください。

これらの要素の予期される値を次の表に示します。

ビジネス・サービス・オブジェクト・インタフェースのヘッダー変数と予期される値

要素名	予期される値
RESPONSIBILITY_NAME	responsibility_name (System Administrator など)または{key} responsibility_key ({key} SYSTEM_ADMINISTRATOR など)
RESPONSIBILITY_APPL_NAME	アプリケーション短縮名(FND など)
SECURITY_GROUP_NAME	セキュリティ・グループ・キー(STANDARD など)
NLS_LANGUAGE	NLS 言語(AMERICAN など)
ORG_ID	組織 ID (202 など)

注意: NLS_Language および ORG_ID は、オプションで渡される値です。

- NLS 言語の要素が指定された場合、渡された言語で SOAP 要求を使用できます。すべての対応する SOAP 応答とエラー・メッセージも、同じ言語で戻すことができます。言語が指定されていない場合、ユーザーのデフォルトの言語が使用されます。
- サービスの実行が特定組織に依存する場合は、その SOAP 要求の ServiceBean_Header で ORG_ID 要素を渡す必要があります。

BPEL PM で Invoke アクティビティの前に Assign アクティビティを構成することによって、コンテキスト情報を指定できます。

SOAP 要求のヘッダー変数の設定方法の詳細は、「ServiceBean_Header パラメータの割当て」(ページ 9-19)を参照してください。

BPEL プロセスによる SOA コンポジット・アプリケーションの作成フロー

この章では、シナリオに基づいて次の設計時タスクを説明します。

1. BPEL プロセスによる SOA コンポジット・アプリケーションの作成、ページ 9-6

このステップを使用して、ApprovePO.bpel という BPEL プロジェクトを含む新しい SOA コンポジット・アプリケーションを作成します。これにより、サード・パーティ・アプリケーションからの入力を受け入れる受入と、要求アプリケーションに BPEL プロセスの出力を返信する返信の 2 つのダミー・アクティビティが自動的に作成されます。

2. パートナ・リンクの作成、ページ 9-9

このステップを使用して、PurchaseOrderService Web サービスのパートナ・リンクを作成します。

3. ファイル・アダプタのパートナ・リンクの追加、ページ 9-9

これは、要求者から受け取った発注詳細を同期的に取り扱います。

4. Invoke アクティビティの追加、ページ 9-14

このステップを使用して、次の目的の 2 つの Invoke アクティビティを作成します。

1. ファイル・アダプタ - 同期読取り操作を指し示して、入力ファイルから発注を読み取ります。
2. PurchaseOrderService Web サービスを指し示して、単一発注承認プロセスを開始します。

5. Assign アクティビティの追加、ページ 9-18

このステップを使用して、次のことを目的とした 3 つの Assign アクティビティを作成します。

1. SOAHeader の詳細を設定します。
2. ファイル・アダプタから読み取った発注詳細を、PurchaseOrderService Web サービスの Invoke アクティビティに入力として渡します。
3. 単一発注承認情報を、返信アクティビティを介して要求者に渡します。

BPEL プロセスのサービス・コンポーネントを使用した SOA コンポジット・アプリケーションの一般情報と作成方法の詳細は、*Oracle Fusion Middleware Oracle SOA Suite 開発者ガイド*を参照してください。

BPEL プロセスによる SOA コンポジット・アプリケーションの作成

このステップを使用して、様々な BPEL プロセス・アクティビティを含む SOA コンポジット・アプリケーションを作成します。

BPEL プロジェクトを使用して新規 SOA コンポジット・アプリケーションを作成するには:

1. Oracle JDeveloper BPEL デザイナを開きます。
2. アプリケーション・ナビゲータで「**新規アプリケーション**」をクリックします。
「SOA アプリケーションの作成 - アプリケーションの名前付け」ページが表示されます。

「SOA アプリケーションの作成」の「アプリケーションの名前付け」ページ

Create SOA Application - Step 1 of 3

Name your application

Application Name: BSO

Directory: C:\JDeveloper\mywork\BSO Browse...

Application Package Prefix:

Application Template:

- Java Desktop Application (ADF)
Creates a databound rich client application. The application consists of one project for the client (ADF Swing), and another project for the ADF Model (ADF Business Components).
- Java EE Web Application
Creates a databound web application. The application consists of one project for the view and controller components (JSF), and another project for the data model (EJB session beans and JPA entities).
- SOA Application**
Creates a SOA (service-oriented architecture) application. The application consists of one SOA project for the SOA composite, components, and adapters.

Help < Back Next > Finish Cancel

3. 「アプリケーション名」フィールドにアプリケーションの適切な名前を入力して、「アプリケーション・テンプレート」リストから「SOA アプリケーション」を選択します。

「次へ」をクリックします。「SOA アプリケーションの作成 - プロジェクトの名前付け」ページが表示されます。

4. 「プロジェクト名」フィールドにプロジェクトの適切な名前を入力します(ApprovePO など)。

5. 「プロジェクト・テクノロジー」タブで、「Web サービス」を選択して、「使用可能」のテクノロジー・リストから「選択済」のテクノロジー・リストに「SOA」が選択されるようにします。

「次へ」をクリックします。「SOA アプリケーションの作成 - SOA 設定の構成」ページが表示されます。

6. 「コンポジット・テンプレート」リストからコンポジット **BPEL プロセス**を選択して、「終了」をクリックします。新規アプリケーションと SOA プロジェクトを作成しました。これにより、SOA コンポジットが自動的に作成されます。

「BPEL プロセスの作成」ページが表示されます。

「BPEL プロセスの作成」ページ

Create BPEL Process

BPEL Process

A BPEL process is a service orchestration, based on the BPEL specification, used to describe/execute a business process (or large grained service), which is implemented as a stateful service.

☒ BPEL 1.1 Specification ☐ BPEL 2.0 Specification

Name: ApprovePO

Namespace: http://xmlns.oracle.com/BSO_ws/ApprovePO/ApprovePO

Template: Asynchronous BPEL Process

Service Name: approvepo_client

☒ Expose as a SOAP service

Delivery: async.persist

Input: {http://xmlns.oracle.com/BSO_ws/ApprovePO/ApprovePO}process

Output: {http://xmlns.oracle.com/BSO_ws/ApprovePO/ApprovePO}processResponse

Help OK Cancel

7. デフォルトの「**BPEL 1.1 Specification**」の選択を変更せずに維持します。これにより、BPEL 1.1 Specificationをサポートする BPEL プロジェクトが作成されます。
「名前」フィールドに BPEL プロセスの適切な名前(ApprovePO など)を入力します。
「テンプレート」フィールドで「**同期 BPEL プロセス**」を選択します。
「トランザクション」ドロップダウン・リストから「**required**」を選択します。「**OK**」をクリックします。
受入アクティビティと返信アクティビティのある同期 BPEL プロセスが作成されます。指定した名前(ApprovePO.bpel や ApprovePO.wsdl など)を使用する bpel や wsdl などの必要なソース・ファイルおよび composite.xml も生成されます。
8. 「SOA コンテンツ」→「ビジネス・ルール」にナビゲートし、composite.xml をダブルクリックしてコンポジット・ダイアグラムを表示します。
ApprovePO コンポーネントをダブルクリックして、BPEL プロセスを開きます。

パートナ・リンクの作成

このステップを使用して、PurchaseOrderService というパートナ・リンクを構成します。

PurchaseOrderService のパートナ・リンクを作成するには:

1. Oracle JDeveloper で、「パートナ・リンク」領域にマウスを置いて右クリックして、プルダウン・メニューから「パートナ・リンクの作成」を選択します。または、「BPEL コンストラクト」リストから「パートナ・リンク」を、プロセス・ダイアグラムの右側の「パートナ・リンク」スイムレーンにドラッグ・アンド・ドロップできます。

「パートナ・リンクの作成」ウィンドウが表示されます。

2. 「WSDL ファイル」フィールドに、統合リポジトリから以前に記録した PurchaseOrderService

/oracle/apps/fnd/framework/toolbox/tutorial/PurchaseOrderService
ce に対応する WSDL URL をコピーして貼り付けます。[Tab]キーを押します。

3. デフォルトでパートナ・リンク・タイプを作成する新しい WSDL ファイルをシステムで作成するかどうかを尋ねる「パートナ・リンク・タイプ」メッセージ・ダイアログ・ボックスが表示されます。

パートナ名の値が自動的に移入されるように、「はい」をクリックします。

ドロップダウン・リストから「パートナ・リンク・タイプ」フィールドと「パートナ・ロール」フィールドを選択します。「適用」をクリックします。

6. 「OK」をクリックして、パートナ・リンクの構成を完了します。パートナ・リンクは、必要な WSDL 設定で作成され、BPEL プロジェクトでは、プロセス・ダイアグラムの枠線の領域の新しいアイコンによって表現されます。

ファイル・アダプタのパートナ・リンクの追加

このステップを使用して、サード・パーティ・アプリケーションから受け取った発注詳細を同期的に読み取ります。

ファイル・アダプタのパートナ・リンクを追加するには:

1. Oracle JDeveloper で、「BPEL サービス」リストから「ファイル・アダプタ」サービスを、プロセス・ダイアグラムの右側の「パートナ・リンク」スイムレーンにドラッグ・アンド・ドロップします。ファイル・アダプタ構成ウィザードへようこそページが表示されます。
2. 「次へ」をクリックします。「サービス名」ダイアログ・ボックスが表示されます。
3. ファイル・アダプタ・サービス名(ReadPO など)を入力します。
4. 「次へ」をクリックします。「アダプタ・インタフェース」ダイアログ・ボックスが表示されます。

5. 「操作およびスキーマから定義(後で指定)」ラジオ・ボタンを選択して、「次へ」をクリックします。「操作」ダイアログ・ボックスが表示されます。

操作の指定

Adapter Configuration Wizard - Step 4 of 8

Operation

The File Adapter supports four operations. There is a Read File operation that polls for incoming files in your local file system, a Write File operation that creates outgoing files, a Synchronous Read File operation that reads the current contents of a file, and a List Files operation that lists file names in specified locations. Specify the Operation type and Operation Name. Only one operation per Adapter Service may be defined using this wizard.

Operation Type: ☐ Read File ☐ Write File ☒ Synchronous Read File ☐ List Files

Operation Name:

Help < Back Next > Finish Cancel

6. 「同期 Read File」などの操作タイプを指定します。これにより、「操作名」フィールドが自動的に設定されます。

「次へ」をクリックして、「ファイル・ディレクトリ」ダイアログ・ボックスにアクセスします。

入力ファイル・ディレクトリの指定

Adapter Configuration Wizard - Step 5 of 8

File Directories

Enter directory information for the incoming file of the Synchronous Read File operation.

Directory names are specified as: ☒ Physical Path ☐ Logical Name

Directory for Incoming Files (physical path):

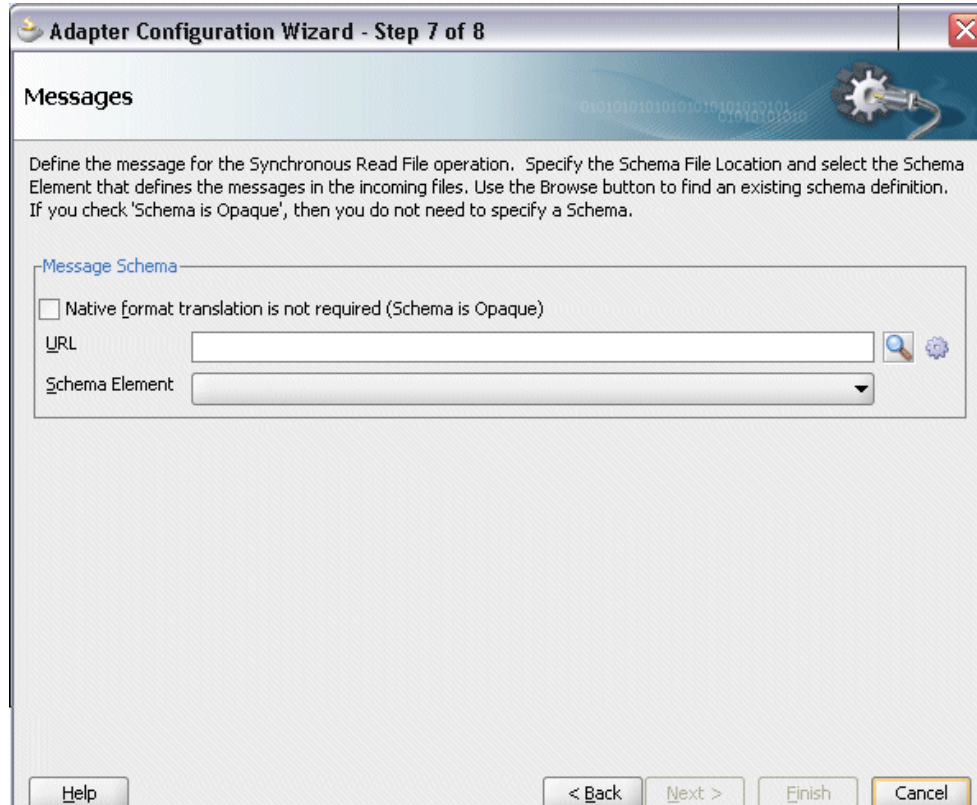
☐ Archive processed files
Archive Directory for Processed Files (physical path):

☒ Delete files after successful retrieval

7. 「物理パス」ラジオ・ボタンを選択して、入力ペイロードのファイル・ディレクトリの情報を
入力します。たとえば、ディレクトリ名として /usr/tmp/ と入力します。

「次へ」をクリックして、「ファイル名」ダイアログ・ボックスを開きます。
8. 同期 Read File 操作のファイル名を入力します。たとえば、Input.xml と入力します。「次
へ」をクリックします。「メッセージ」ダイアログ・ボックスが表示されます。

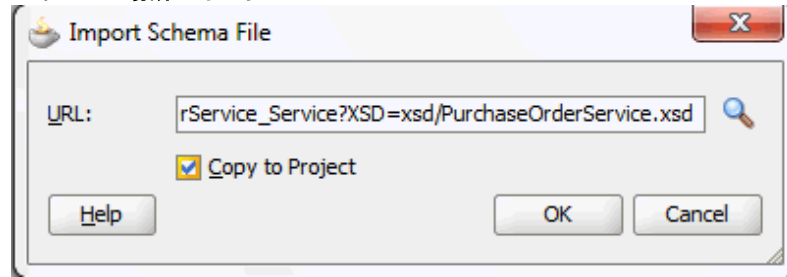
メッセージ・スキーマの指定



9. 「URL」フィールドの右にある「スキーマ・ファイルを参照」を選択します。「タイプ・チューザ」ウィンドウが表示されます。
 1. 「タイプ・チューザ」ウィンドウの右上隅にある「スキーマ・ファイルのインポート」ボタンをクリックします。
 2. サービスのスキーマの場所を入力します
(`http://<soa_suite_hostname>:<port>/soa-infra/services/default/<jndi_name>_PurchaseOrderService/PurchaseOrderService_Service/?XSD=xsd/PurchaseOrderService.xsd` など)。

サービスのスキーマの場所は、サービスの WSDL URL
(`http://<soa_suite_hostname>:<port>/soa-infra/services/default/<jndi_name>_PurchaseOrderService/PurchaseOrderService_Service?wsdl` など)から見つけることができます。

スキーマの場所のインポート



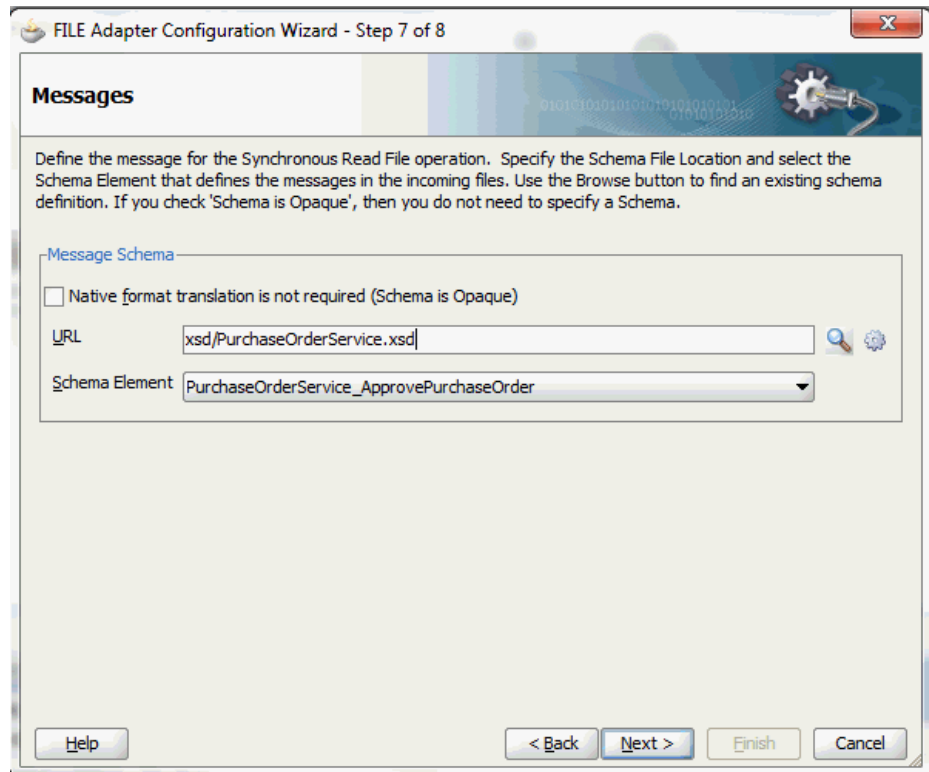
3. 「プロジェクトにコピー」チェック・ボックスを選択して、「**OK**」をクリックします。
4. 「ファイルのローカライズ」ウィンドウが表示されます。「**インポートされたファイルの元のディレクトリ構造を保持**」チェック・ボックスが選択されていることを確認し、「**OK**」をクリックします。

「タイプ・チューザ」ウィンドウに、「インポートしたスキーマ」フォルダが自動的に追加されます。

5. 「インポートしたスキーマ」フォルダを展開し、PurchaseOrderService.xsd から PurchaseOrderService_ApprovePurchaseOrder を選択します。「**OK**」をクリックします。

選択した.xsd が URL として表示され、

PurchaseOrderService_ApprovePurchaseOrder がスキーマ要素として選択されます。



10. 「次へ」→「終了」をクリックします。ウィザードによって、パートナー・リンクに対応する WSDL ファイルが生成されます。メインの「パートナー・リンクの作成」ダイアログ・ボックスが表示されるので、新しい WSDL ファイル ReadPO.wsdl を指定します。

「適用」→「OK」をクリックして構成を完了し、ファイル・アダプタ・サービスに必要な WSDL 設定を指定してパートナー・リンクを作成します。

BPEL プロセス・ダイアグラムに ReadPO パートナ・リンクが表示されます。

11. アプリケーション・ウィンドウで、ファイル ReadPO_file.jca にナビゲートします。DeleteFile プロパティの値を false に設定します。

Invoke アクティビティの追加

このステップでは、次の 2 つの Invoke アクティビティを構成します。

1. ファイル・アダプタ ReadPO を指し示して、受入アクティビティから発注を同期的に読み取ります。
2. PurchaseOrderService パートナ・リンクを指し示して、Assign アクティビティから受け取った取引情報を送信し、単一発注承認プロセスを開始します。

ReadPO パートナ・リンクの **Invoke** アクティビティを追加するには:

1. Oracle JDeveloper で、コンポーネント・パレットから「**BPEL コンストラクト**」を展開します。
Invoke アクティビティを、プロセス・ダイアグラムの中央のスイムレーンの、**receiveInput** と **replyOutput** のアクティビティの間にドラッグ・アンド・ドロップします。
2. **Invoke** アクティビティを ReadPO サービスにリンクします。「編集 Invoke」ダイアログ・ボックスが表示されます。

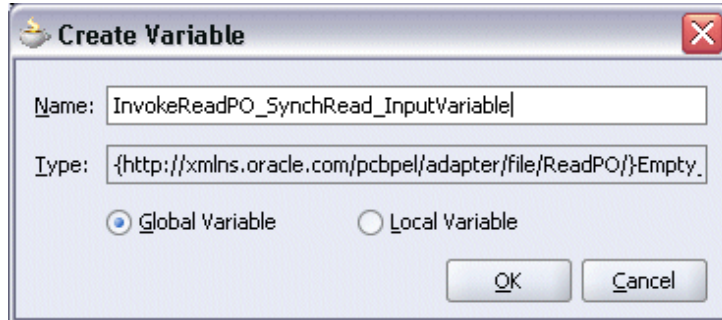
Invoke アクティビティの編集

The screenshot shows the 'Edit Invoke' dialog box. The 'General' tab is selected. The 'Name' field contains 'InvokeReadPO'. The 'Conversation ID' and 'Detail Label' fields are empty. The 'Invoke as Detail' checkbox is unchecked. The 'Interaction Type' is set to 'Partner Link'. The 'Partner Role Web Service Interface' section shows 'Partner Link' as 'ReadPO' and 'Operation' as 'SynchRead'. The 'Variables' section has 'Input' and 'Output' fields, each with a green plus icon and a magnifying glass icon. The dialog has 'Help', 'Apply', 'OK', and 'Cancel' buttons at the bottom.

3. **Invoke** アクティビティの名前(InvokeReadPO など)を入力して、「**入力変数**」フィールドの横にある「**入力変数の作成**」アイコンをクリックして、新しい変数を作成します。「変数の作成」ダイアログ・ボックスが表示されます。

「名前」フィールドに適切な名前を入力します。デフォルト名を受け入れることもできます。

変数の作成

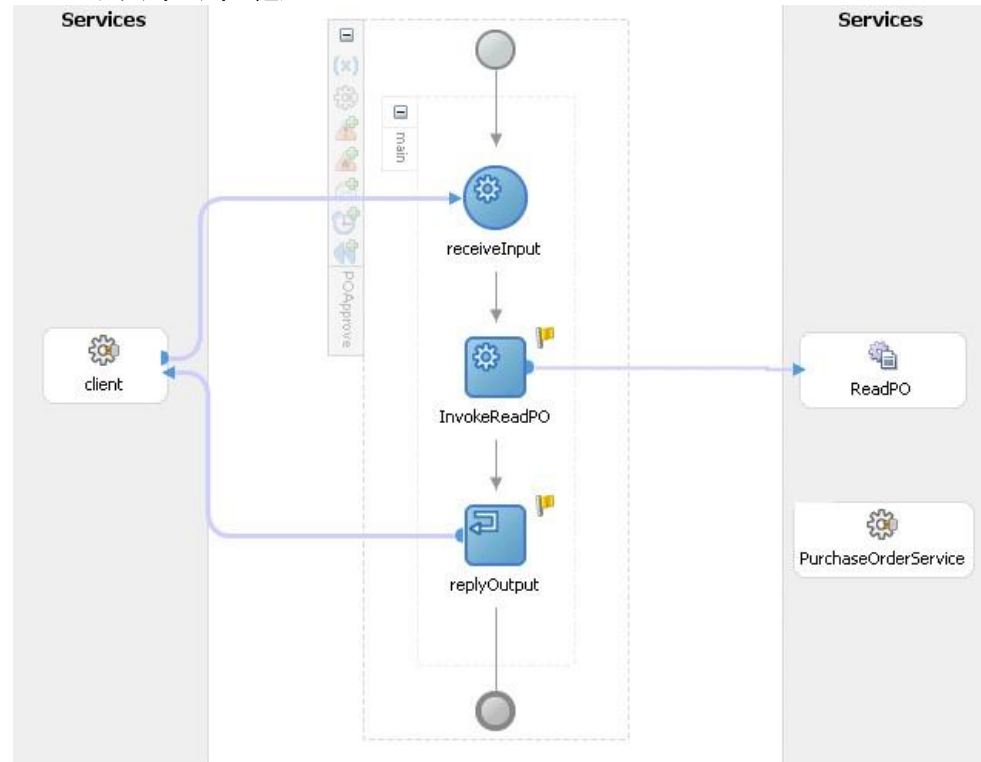


4. 「グローバル変数」を選択して、「OK」をクリックします。
5. 「出力変数」フィールドの横にある「入力変数の作成」アイコンをクリックして、新しい変数を作成します。「変数の作成」ダイアログ・ボックスが表示されます。

「名前」フィールドに適切な名前を入力します。デフォルト名を受け入れることもできます。「グローバル変数」を選択します。「OK」をクリックします。
6. 「編集 Invoke」ダイアログ・ボックスで「適用」→「OK」をクリックして、**Invoke** アクティビティの構成を完了します。

プロセス・ダイアグラムに **Invoke** アクティビティが表示されます。

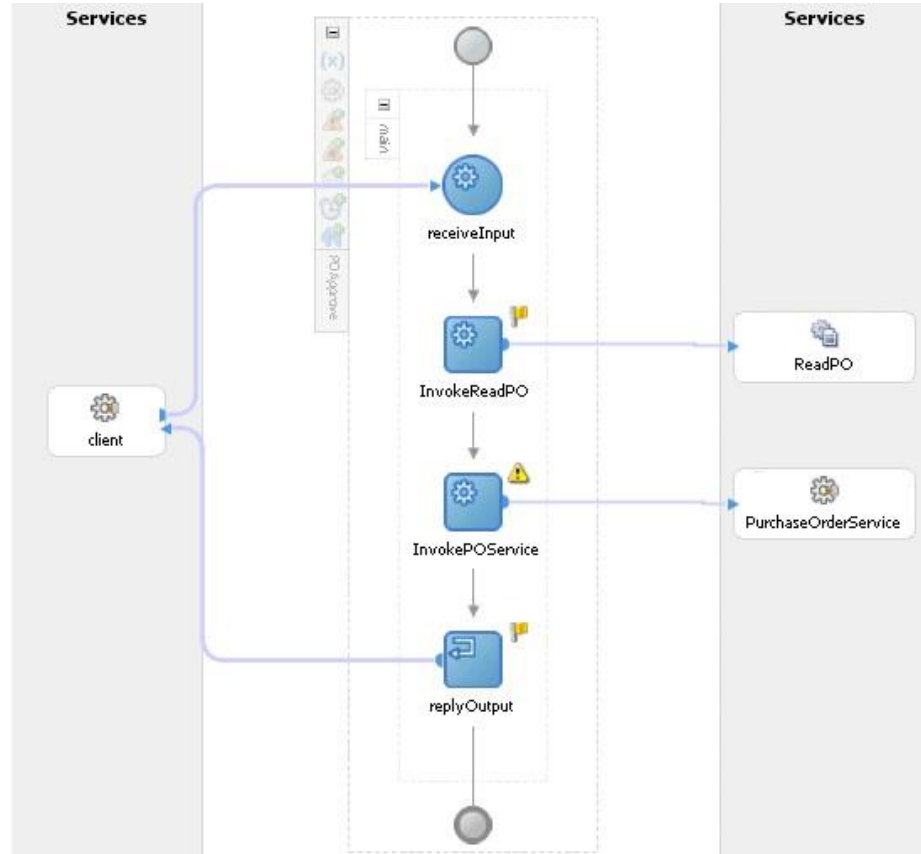
Invoke アクティビティの追加



PurchaseOrderService パートナ・リンクの **Invoke** アクティビティを追加するには:

1. Oracle JDeveloper で、コンポーネント・パレットから「**BPEL コンストラクト**」を展開します。
Invoke アクティビティを、プロセス・ダイアグラムの中央のスイムレーンの、最初の **Invoke** アクティビティと **replyOutput** のアクティビティの後にドラッグ・アンド・ドロップします。
2. **Invoke** アクティビティを **PurchaseOrderService** サービスにリンクします。「編集 **Invoke**」ダイアログ・ボックスが表示されます。
3. 2 つ目の **Invoke** アクティビティの名前(**InvokePOService** など)を入力します。1 つ目の **Invoke** アクティビティの説明に従って、入力変数および出力変数を作成します。「**OK**」をクリックして「変数の作成」ダイアログ・ボックスを閉じます。
4. 「適用」→「**OK**」をクリックして、2 つ目の **Invoke** アクティビティの構成を終了します。プロセス・ダイアグラムに 2 つ目の **Invoke** アクティビティが表示されます。

Invoke アクティビティの追加



Assign アクティビティの追加

このステップでは、次の 3 つの Assign アクティビティを構成します。

1. ヘッダーの詳細を設定します。

注意: Web サービスの承認のために、アプリケーション・コンテキストを SOAP エンベロープに埋め込むために要求される値を渡すには、ServiceBean_Header 要素に対する BPEL プロセスの特定の変数を設定する必要があります。これらのビジネス・サービス・オブジェクト・インタフェース・タイプの ServiceBean_Header 要素は、RESPONSIBILITY_NAME、RESPONSIBILITY_APPL_NAME、SECURITY_GROUP_NAME、NLS_LANGUAGE および ORG_ID です。

SOAP 要求の ServiceBean_Header の設定方法の詳細は、「ServiceBean_Header パラメータの割当て」(ページ 9-19)を参照してください。

2. ファイル・アダプタから読み取った発注詳細を、PurchaseOrderService パートナ・リンクの 2 つ目の **Invoke** アクティビティに入力として渡します。
3. 単一発注承認情報を、ダミーの返信アクティビティを介して要求者に渡します。

ヘッダーの詳細を渡す 1 つ目の **Assign** アクティビティを追加するには:

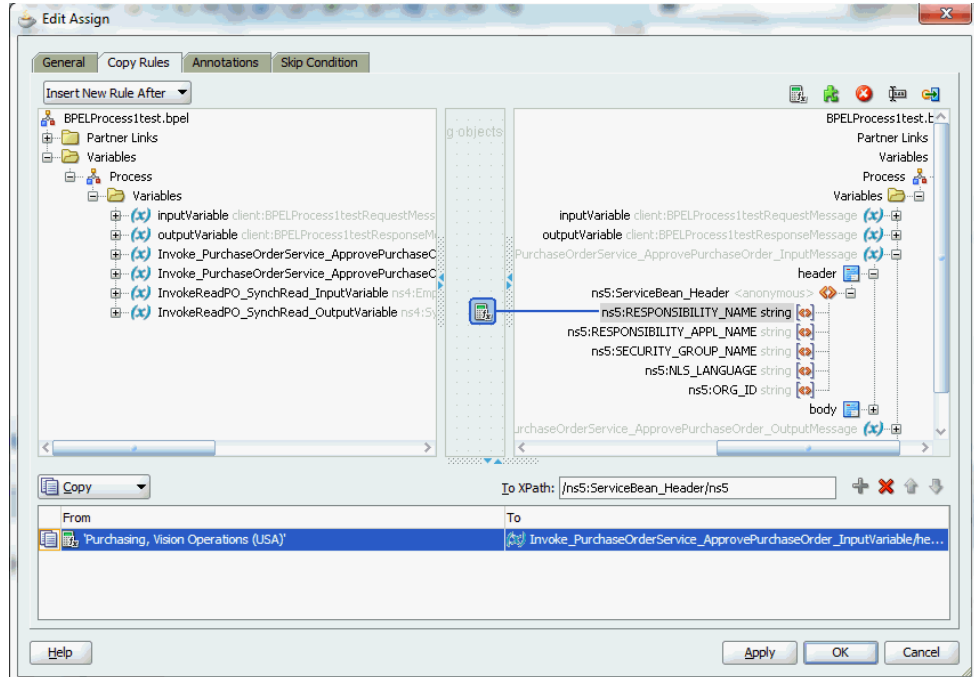
ServiceBean_Header パラメータの割当て:

1. Oracle JDeveloper で、コンポーネント・パレットから「**BPEL コンストラクト**」を展開します。**Assign** アクティビティを、プロセス・ダイアグラムの中央のスイムレーンの、先ほど作成したばかりの 2 つの **Invoke** アクティビティの間にドラッグ・アンド・ドロップします。
2. **Assign** アクティビティをダブルクリックして、「編集 Assign」ダイアログ・ボックスにアクセスします。
3. 「一般」タブをクリックして、Assign アクティビティの名前(SetServiceBeanHeader など)を入力します。
4. 「コピー・ルール」タブを選択し、ターゲット・ツリーを展開します。
 - 「式」アイコンをクリックして、「式ビルダー」ダイアログを起動します。

「式」ボックスに Purchasing, Vision Operations (USA)と入力します。「**OK**」をクリックします。「割当ての編集」ダイアログの中央(ソースとターゲットの各ナビゲーション・ツリー・ノードの間)に、「式」アイコンと式の値が表示されます。
 - 「先」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「InvokePurchaseOrderService_PurchaseOrderService_ApprovePurchaseOrder_InputVariable」→「header」→「ns5:ServiceBean_Header」にナビゲートして、「ns5:RESPONSIBILITY_NAME」を選択します。

「式」アイコンをドラッグして、識別したばかりのターゲット・ノード(ns5:RESPONSIBILITY_NAME)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

パラメータの割当て



5. 「式」アイコンをクリックして「式ビルダー」ダイアログを起動して、2 つ目のパラメータのペ
アを入力します。

- 「式」ボックスに **PUR** と入力します。「**OK**」をクリックします。「割当ての編集」ダイア
ログの中央の、「元」と「先」のナビゲーション・ツリー・ノードの間に、「式」アイコンと
式の値(**PUR**)が表示されます。
- 「先」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→
「InvokePurchaseOrderService_PurchaseOrderService_ApprovePurchaseOrder_Inpu
tVariable」→「header」→「ns5:ServiceBean_Header」にナビゲートして、「ns5:
RESPONSIBILITY_APPL_NAME」を選択します。

「式」アイコンをドラッグして、識別したばかりのターゲット・ノード(**ns5:
RESPONSIBILITY_APPL_NAME**)に接続します。これにより、ソース・ノードとターゲッ
ト・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある
「元」と「先」の各セクションに、コピー・ルールが表示されます。

6. 「式」アイコンをクリックして「式ビルダー」ダイアログを起動して、3 つ目のパラメータのペ
アを入力します。

- 「式」ボックスに **STANDARD** と入力します。「**OK**」をクリックします。「割当ての編
集」ダイアログの中央の、「元」と「先」のナビゲーション・ツリー・ノードの間に、式の
値(**STANDARD**)を持つ「式」アイコンが表示されます。

- 「先」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→
「InvokePurchaseOrderService_PurchaseOrderService_ApprovePurchaseOrder_InputVariable」→「header」→「ns5:ServiceBean_Header」にナビゲートして、
「ns5: SECURITY_GROUP_NAME」を選択します。

「式」アイコンをドラッグして、識別したばかりのターゲット・ノード(ns5:SECURITY_GROUP_NAME)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

7. 「式」アイコンをクリックして「式ビルダー」ダイアログを起動して、4 つ目のパラメータのペアを入力します。

- 「式」ボックスに AMERICAN と入力します。「OK」をクリックします。「割当ての編集」ダイアログの中央の、「元」と「先」のナビゲーション・ツリー・ノードの間に、式の値(AMERICAN)を持つ「式」アイコンが表示されます。
- 「先」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→
「InvokePurchaseOrderService_PurchaseOrderService_ApprovePurchaseOrder_InputVariable」→「header」→「ns5:ServiceBean_Header」にナビゲートして、
「ns5: NLS_LANGUAGE」を選択します。

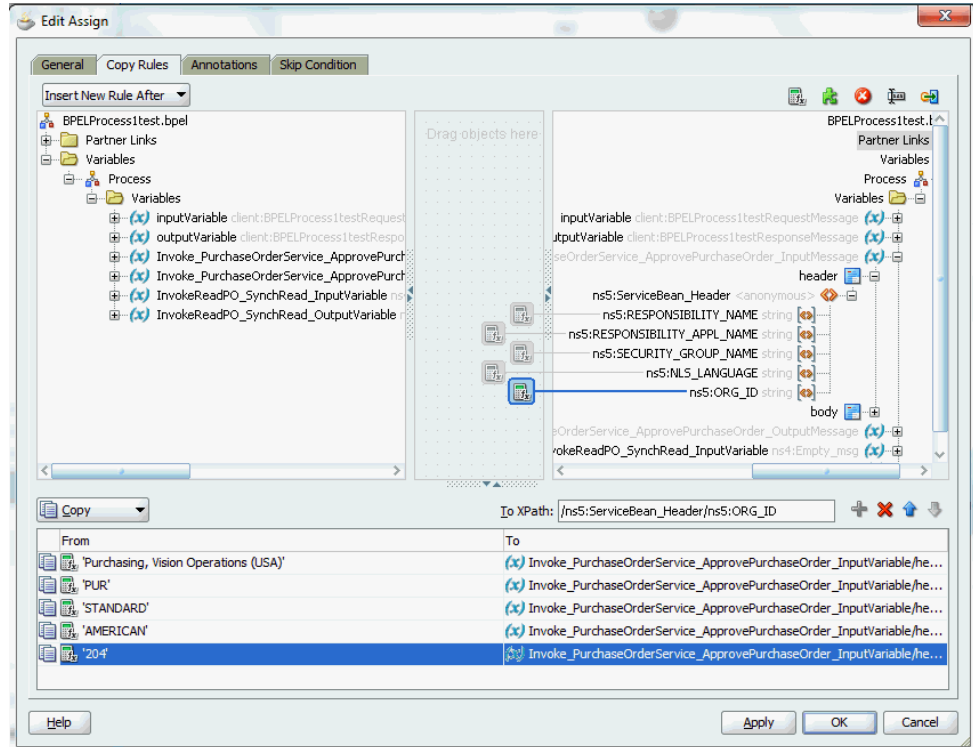
「式」アイコンをドラッグして、識別したばかりのターゲット・ノード(ns5:NLS_LANGUAGE)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

8. 「式」アイコンをクリックして「式ビルダー」ダイアログを起動して、5 つ目のパラメータのペアを入力します。

- 「式」ボックスに 204 と入力します。「OK」をクリックします。「割当ての編集」ダイアログの中央の、「元」と「先」のナビゲーション・ツリー・ノードの間に、式の値(204)を持つ「式」アイコンが表示されます。
- 「先」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→
「InvokePurchaseOrderService_PurchaseOrderService_ApprovePurchaseOrder_InputVariable」→「header」→「ns5:ServiceBean_Header」にナビゲートして、
「ns5:ORG_ID」を選択します。

「式」アイコンをドラッグして、識別したばかりのターゲット・ノード(ns5:ORG_ID)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

パラメータの割当て



9. 「適用」→「OK」をクリックして、Assign アクティビティの構成を完了します。

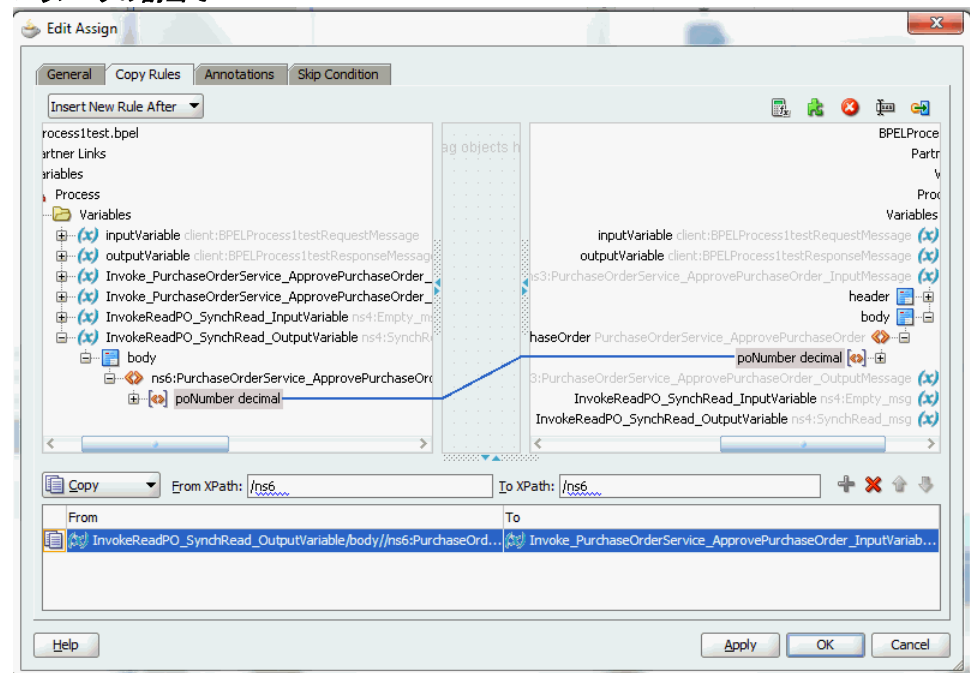
発注詳細を **InvokePOServiceInvoke** アクティビティに渡す 2 つ目の **Assign** アクティビティを入力するには:

1. Oracle JDeveloper BPEL デザイナで、2 つ目の **Assign** アクティビティを、「BPEL コンストラクト」からプロセス・ダイアグラムの中央のスイムレーンの、1 つ目の **Assign** アクティビティと **Invoke** アクティビティの間にドラッグ・アンド・ドロップします。
2. **Assign** アクティビティをダブルクリックして、「編集 Assign」ダイアログ・ボックスにアクセスします。
3. 「一般」タブをクリックして、Assign アクティビティの名前(SetPOApproval など)を入力します。
4. 「コピー・ルール」タブを選択し、ターゲット・ツリーを展開します。
 - 「元」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「InvokeReadPO_SynchRead_OutputVariable」→「PurchaseOrderService_ApprovePurchaseOrder」→「body」→「ns6:PurchaseOrderService_ApprovePurchaseOrder」にナビゲートして、「poNumber」を選択します。
 - 「先」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→

「InvokePurchaseOrderService_PurchaseOrderService_ApprovePurchaseOrder_InputVariable」→「body」→
「ns6:PurchaseOrderService_ApprovePurchaseOrder」にナビゲートして、
「poNumber」を選択します。

ソース・ノード(poNumber)をドラッグして、識別したばかりのターゲット・ノード(poNumber)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

パラメータの割当て



5. 「適用」→「OK」をクリックして、Assign アクティビティの構成を完了します。

出力する SOAP レスポンスを設定する 3 つ目の Assign アクティビティを入力するには:

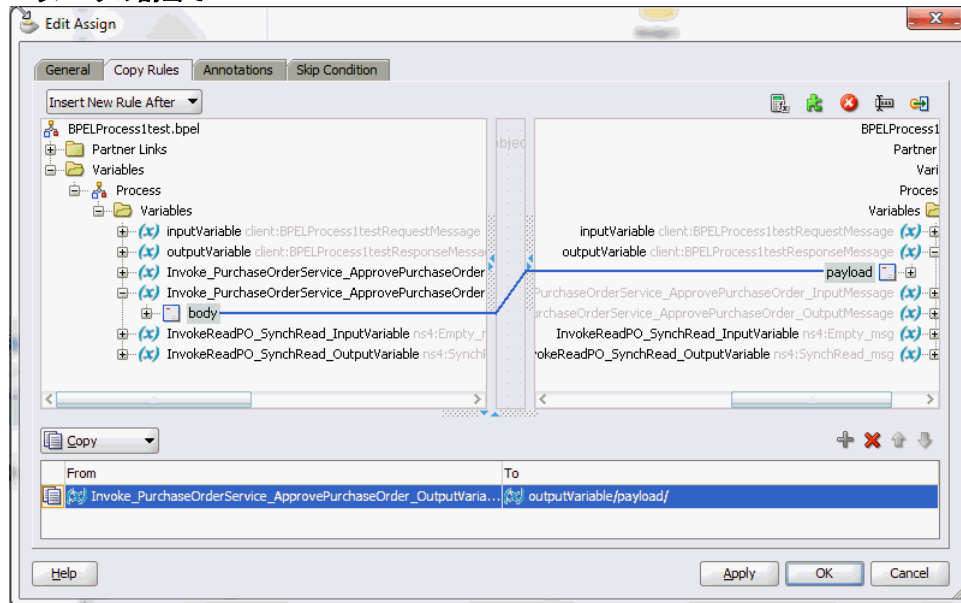
- 3 つ目の Assign アクティビティを追加するには、**Assign** アクティビティを、プロセス・ダイアグラムの中央のスイムレーンの、InvokePOService **Invoke** アクティビティと **ReplyOutput** アクティビティの間にドラッグ・アンド・ドロップします。
- 1 つ目の Assign アクティビティの作成で説明したステップ 2 とステップ 3 を繰り返して、SetPOStatus という 2 つ目の Assign アクティビティを追加します。
- 「コピー・ルール」タブを選択して、ソース・ツリーとターゲット・ツリーを展開します。
 - 「元」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→
「InvokePurchaseOrderService_PurchaseOrderService_ApprovePurchaseOrder」

r_OutputVariable」にナビゲートして、「**body**」を選択します。

- 「先」のナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「**outputVariable**」にナビゲートして、「**payload**」を選択します。

ソース・ノード(本文)をドラッグして、識別したばかりのターゲット・ノード(ペイロード)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

パラメータの割当て



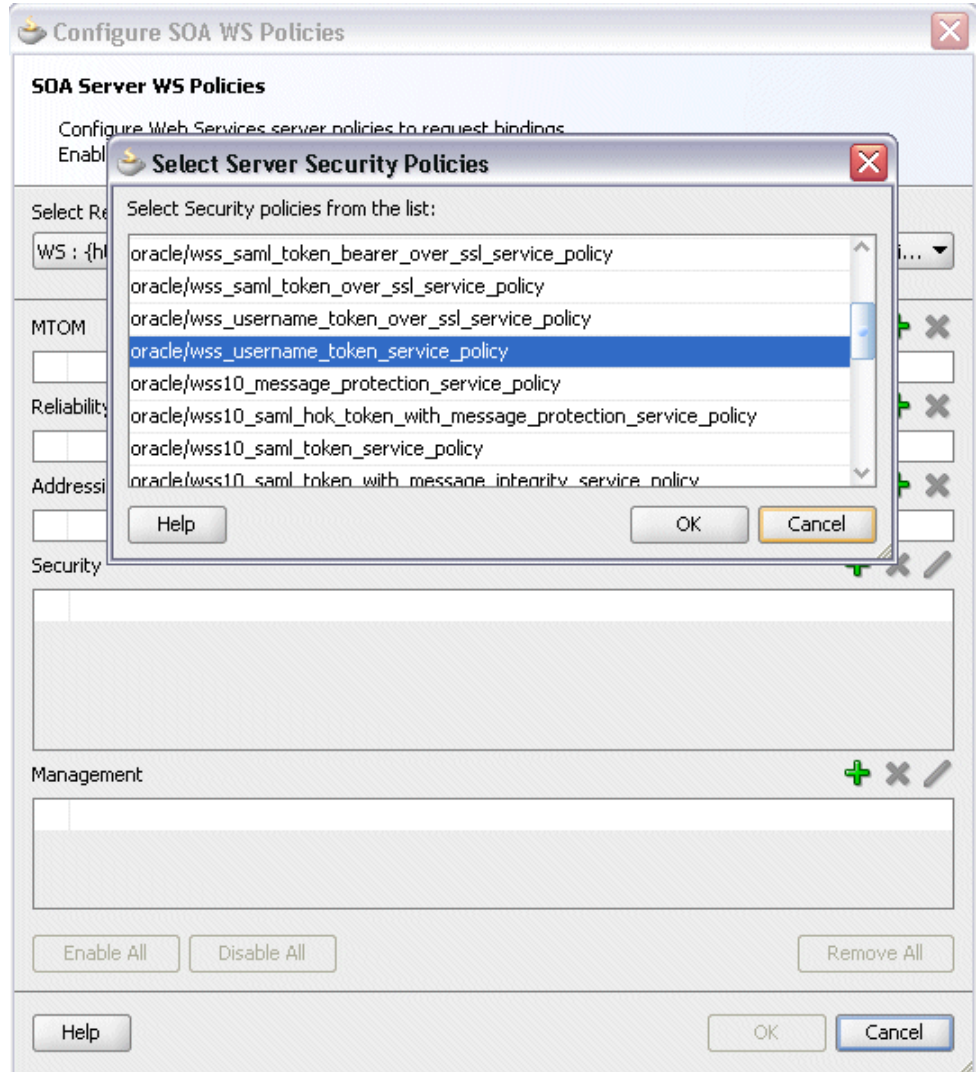
4. 「適用」→「OK」をクリックして、Assign アクティビティの構成を完了します。

Web サービス・ポリシーの構成

次のステップを使用して、設計時のセキュリティ・ポリシーを追加します。

1. 「SOA コンテンツ」→「ビジネス・ルール」→**composite.xml** に移動します。
PurchaseOrderService サービスを右クリックして、ドロップダウン・リストから「**WS** ポリシーの構成」を選択します。
2. 「SOA WS ポリシーの構成」ダイアログが表示されます。
「セキュリティ」セクションで「**セキュリティ・ポリシーの追加**」アイコン(+)をクリックします。
サーバー・セキュリティ・ポリシーの選択ダイアログが表示されます。

必要なセキュリティ・ポリシーの選択



「oracle/wss_username_token_service_policy」を選択して、「OK」をクリックします。添付されたセキュリティ・ポリシーが「セキュリティ」セクションに表示されます。

3. ナビゲーション・メニューから「表示」→「プロパティ・インスペクタ」を選択して、PurchaseOrderService サービス・コンポーネントの「プロパティ・インスペクタ」ウィンドウを表示します。

「プロパティ」セクションで、バインディング・プロパティの「セキュリティ・ポリシーの追加」アイコン(+)をクリックします。「プロパティの作成」ダイアログが表示されます。

「名前」フィールドに `oracle.webservices.auth.username` と入力して、値として `operations` と入力します。

「OK」をクリックします。

4. 「プロパティ」セクションのバインディング・プロパティで「セキュリティ・ポリシーの追加」アイコン(+)を再度クリックする、同じアプローチを使用します。「名前」フィールドに `oracle.webservices.auth.password` と入力します。「値」フィールドに、operations ユーザーに紐づくパスワードを入力します。

「OK」をクリックします。

「プロパティ」セクションに、選択したプロパティの名前と値の両方が表示されます。

実行時における BPEL プロセスによる SOA コンポジットのデプロイおよびテスト

SOA コンポジットに含まれる BPEL クライアントから同期 `PurchaseOrderService` を起動するには、SOA コンポジットを Oracle WebLogic 管理対象サーバーにデプロイする必要があります。これは、Oracle JDeveloper を使用して達成できます。コンポジットがデプロイされたら、Oracle Enterprise Manager Fusion Middleware Control コンソールからテストできます。

前提条件

Oracle JDeveloper を使用して BPEL プロセスによる SOA コンポジットをデプロイする前に、設計時環境とランタイム・サーバーの間の接続を確立する必要があります。必要なサーバー接続を構成する方法の詳細は、「サーバー接続の構成」(ページ B-1)を参照してください。

注意: WebLogic サーバーのローカル・インスタンスが使用される場合、Oracle JDeveloper から「実行」→「サーバー・インスタンスの起動」を選択して、WebLogic サーバーを起動します。WebLogic 管理サーバーの DefaultServer インスタンスが正常に起動されたら、「実行中:DefaultServer」と「メッセージ」のログに<Server started in Running mode>と DefaultServer started のメッセージが表示されるはずです。

次の実行時タスクを実行します。

1. BPEL プロセスによる SOA コンポジット・アプリケーションのデプロイ、ページ 9-26
2. BPEL プロセスによる SOA コンポジット・アプリケーションのテスト、ページ 9-29

BPEL プロセスによる SOA コンポジットのデプロイ

以前に作成した受注承認 SOA コンポジット(`POApprove.bpel`)をデプロイしてから実行する必要があります。

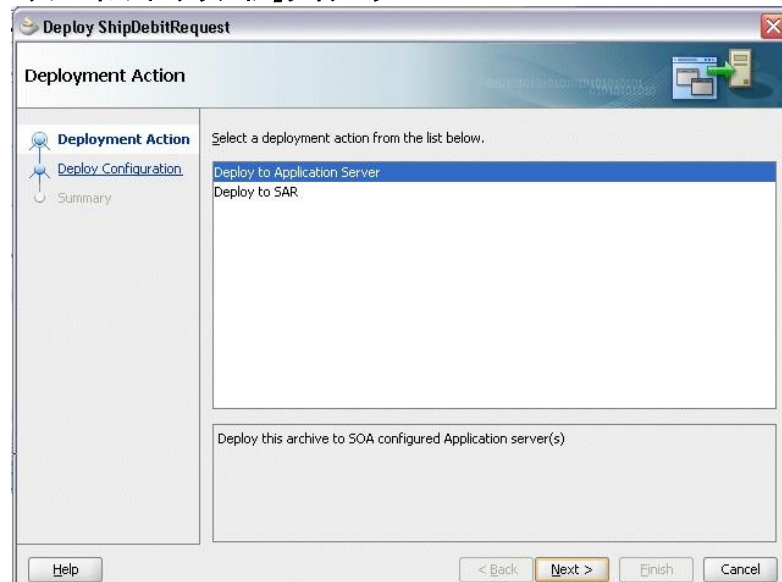
SOA コンポジット・アプリケーションをデプロイするには:

1. JDeveloper のアプリケーション・ナビゲータで、**POApprove** プロジェクトを選択します。
2. プロジェクトを右クリックして、メニューから「配置」→**[project name]**→**[serverConnection]**を選択します。

たとえば、接続が適切に設定されている場合は、「デプロイ」→「POApprove」→「SOAServer」を選択してプロセスをデプロイできます。

注意: サーバー接続を初めて設定する場合は、「デプロイメント・アクション」ダイアログが表示されます。「アプリケーション・サーバーにデプロイ」を選択して、「次へ」をクリックします。

「デプロイメント・アクション」ダイアログ



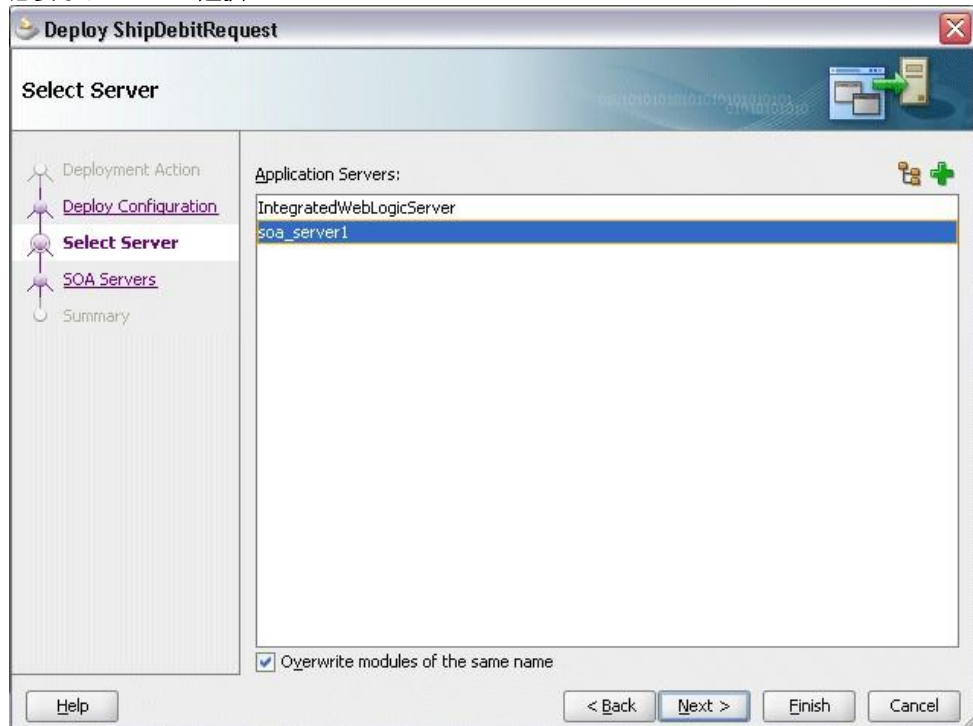
「構成のデプロイ」ダイアログで、「次へ」をクリックして新しいアプリケーション・サーバーを追加する前に、次の情報が選択されるようにします。

- 新規リビジョン ID: 1.0
- コンポジット・リビジョンをデフォルトとしてマークします。: このチェック・ボックスを選択します。
- 同じリビジョン ID で既存のコンポジットを上書きします。: このチェック・ボックスを選択します。

JDeveloper から新しい Oracle WebLogic Server 接続を作成するステップは、「サーバー接続の構成」(ページ B- 1)を参照してください。

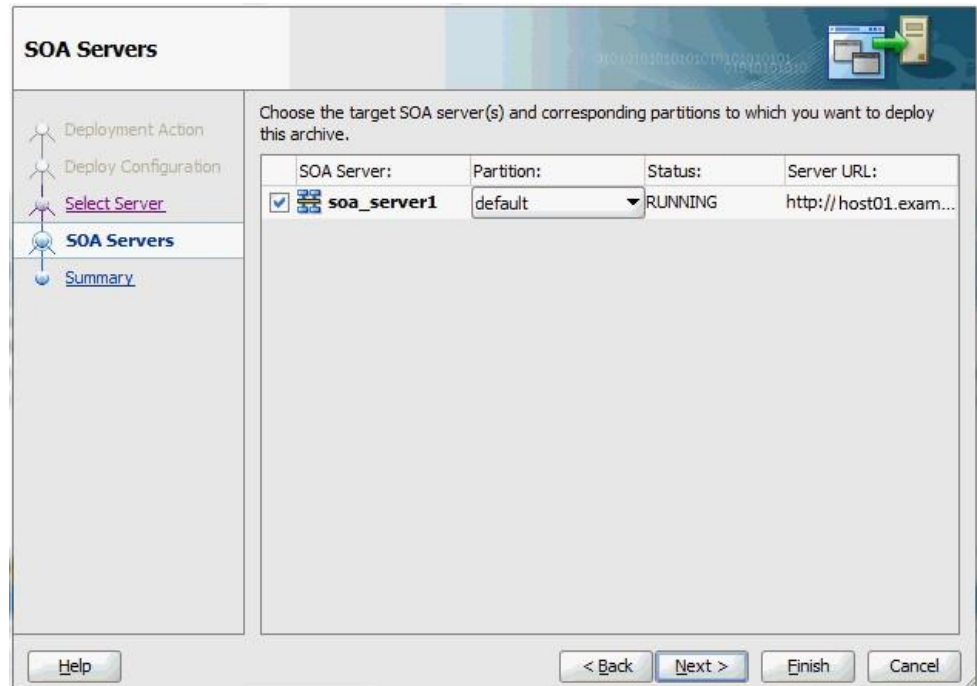
3. 「サーバーの選択」ダイアログで、以前にサーバー接続を確立した soa-server1 を選択します。「次へ」をクリックします。

必要なサーバーの選択



4. 「SOA サーバー」ダイアログで、デフォルトのターゲット SOA サーバー(soa-server1)の選択を受け入れます。

必要な SOA サーバーの選択



「次へ」→「終了」をクリックします。

5. Oracle JDeveloper セッションから初めてコンポジットをデプロイする場合は、「認証リクエスト」ウィンドウが表示されます。Oracle SOA Suite のインストール時に指定したユーザー名とパスワード情報を入力します。「OK」をクリックします。
6. デプロイ処理が開始します。SOA - ログ・ウィンドウで、デプロイ・プロセスをモニターして、コンパイルが成功したことを確認します。

デプロイメント - ログ・ウィンドウで、デプロイが成功したことを確認します。

BPEL プロセスによる SOA コンポジット・アプリケーションのテスト

SOA コンポジットに含まれる BPEL プロセスがデプロイされたら、Oracle Enterprise Manager Fusion Middleware Control コンソールからプロセスを管理およびモニターできます。

Oracle SOA Suite の詳細は、*Oracle Fusion Middleware Oracle SOA Suite 管理者ガイド*を参照してください。

BPEL プロセスによる SOA コンポジット・アプリケーションをテストするには:

1. Oracle Enterprise Manager Fusion Middleware Control コンソール
(<http://<hostname>:<port>/em>)にナビゲートします。ログイン・ページが表示されます。

2. インストール時に指定したユーザー名およびパスワード情報を入力し、「ログイン」をクリックしてファームにログインします。デプロイしたコンポジット(ApprovePO)がアプリケーション・ナビゲーション・ツリーに表示されます。

複数のターゲット Oracle Enterprise Manager Fusion Middleware Control コンソール・ファームがある場合、適切なターゲット・インスタンス・ファームを選択する必要がある場合があります。

3. 「ファーム」のナビゲーション・ペインから、ツリーの「SOA」→「soa-infra」ノードを展開して、「SOA インフラストラクチャ」のホーム・ページおよびメニューを使用して、soa-infra 管理対象サーバーで実行されているデプロイ済の SOA コンポジット・アプリケーションにアクセスします。

ApprovePO [1.0]のリンクをクリックします。

4. 「ポリシー」タブをクリックして、設計時に PurchaseOrderService サービス・バインディングに添付した oracle/wss_username_token_service_policy ポリシーがここに表示されていることを確認します。
5. ApprovePO [1.0]のホーム・ページで、「テスト」をクリックします。
6. インスタンスを開始するための「Web サービスのテスト」ページが表示されます。「引数の入力」セクションには、使用する XML ペイロード・データの情報を指定できます。

注意: 設計時に WS-Security 資格証明が入力されない場合、「リクエスト」タブの上部にある「セキュリティ」セクションの「WSS ユーザー名トークン」オプションを選択して、実行時に資格証明を入力できます。「ユーザー名」フィールドに operations と入力し、ユーザー operations に関連付けられているパスワードを入力します。

「Web サービスのテスト」をクリックして、プロセスを開始します。

完了時に、「レスポンス」タブにテスト結果が表示されます。

7. BPEL サービス・コンポーネント・インスタンスのリンク(ApprovePO など)をクリックして、「インスタンス」ページを表示し、その「監査証跡」タブで BPEL アクティビティの実行の詳細を確認できます。

「フロー」タブをクリックして、BPEL プロセスのフロー・ダイアグラムを確認します。プロセス・ダイアグラムのアクティビティをクリックして、アクティビティの詳細と、プロセスを介したペイロードのフローを表示します。

8. 発注が正常に承認されたことを確認します。

Oracle E-Business Suite でのプロセスの検証

さらに、Oracle E-Business Suite で BPEL プロセスを検証できます。「購買」職責を持つユーザーとして Oracle E-Business Suite にログインします。発注フォームを開き、仕入先を検索して発注詳細を表示します。「ステータス」フィールドが「承認」になっています。

コンポジット・サービス - BPELの使用

概要

コンポジット・サービスは、大まかに抽象化されたサービスです。ネイティブ・サービスをビルディング・ブロックとして使用して、Web サービス構成言語 BPEL でビジネス起動シーケンスを有用なエンド・ツー・エンドのビジネス・フローにオーケストレーションします。

設計時に、統合開発者は、Oracle JDeveloper 10g の BPEL プロセス・コンポーネントを使用して、一連のサービス・コンポーネントを 1 つのビジネス機能に編成し、統合リポジトリ・アノテーションの標準に基づいてコンポジット・サービス - BPEL 定義に注釈付けします。検証後、新しく作成されたコンポジット・サービス - BPEL を統合リポジトリにアップロードし、すべてのユーザーが表示および利用できるようにします。

統合開発者が各コンポジット・サービス - BPEL の詳細を表示するには、Oracle Integration Repository ブラウザから目的のサービスを選択し、選択したコンポジット・サービス - BPEL をローカル・ディレクトリにダウンロードし、必要に応じて Oracle JDeveloper 10g で BPEL プロセスを変更し、これを Oracle SOA Suite 10g BPEL Process Manager の BPEL サーバーまたは J2EE 環境のサード・パーティ BPEL PM にデプロイします。

この章では、コンポジット・サービス - BPEL を表示、ダウンロードおよび変更する方法について説明します。BPEL プロセスによる SOA コンポジット・サービスの作成方法に関する設計時タスクの詳細は、このマニュアルですでに説明した各インタフェースを参照してください。

- コンポジット・サービス - BPEL の表示、ページ 10-2
- コンポジット・サービス - BPEL のダウンロード、ページ 10-2
- BPEL プロセスの変更およびデプロイ、ページ 10-3

コンポジット - BPEL 定義への注釈付けの方法の詳細は、「コンポジット・サービス - BPEL アノテーション」(A-115 ページ)を参照してください。

統合リポジトリへのコンポジット - BPEL 定義のアップロード方法の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「カスタム統合インタフェース・プロセス・フローの有効化」を参照してください。

コンポジット・サービス - BPEL の表示

注釈付けされたカスタム・コンポジット - BPEL 定義が統合リポジトリにアップロードされると、リポジトリから「コンポジット - BPEL」オプションを表示して、すべてのユーザーが利用できます。

統合開発者がコンポジット・サービス - BPEL を表示するには、Oracle Integration Repository ブラウザ・ウィンドウで「表示別」に「インタフェース・タイプ」を指定して、直接「コンポジット - BPEL」インタフェース・タイプにナビゲートします。または、統合開発者は、「検索」ページで検索を実行して、コンポジット・サービス - BPEL を見つけることもできます。

コンポジット・サービス - BPEL の詳細を表示するには、ナビゲーション・ツリーまたは検索結果のコンポジット・サービス名のリンクをクリックします。コンポジット・サービス - BPEL インタフェースの詳細ページには、サービス名、説明、BPEL ファイル、WSDL ファイルおよび他の注釈付けされた情報が表示されます。

コンポジット・サービス - BPEL の詳細ページでは、「BPEL ファイル」リージョンで次のタスクを実行できます。

- 「要約 WSDL の表示」リンクをクリックして、要約 WSDL ファイルを表示します

参照: *Oracle E-Business Suite Integrated SOA Gateway ユーザーズ・ガイド*の「Web サービス WSDL ソースのレビュー」。

- 「BPEL ファイルの表示」リンクをクリックして、コンポジット - BPEL ファイルを表示します
BPEL コードはポップアップ・ウィンドウに表示され、そこには選択したコンポジット・サービスに含まれる主要な BPEL プロセス・コンポーネントおよびアクティビティが含まれています。

コンポジット - BPEL 定義への注釈付けの方法の詳細は、「コンポジット・サービス - BPEL アノテーション」(A-115 ページ)を参照してください。

統合リポジトリへのコンポジット - BPEL 定義のアップロード方法の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「カスタム統合インタフェース・プロセス・フローの有効化」を参照してください。

対応するコンポジット・サービス BPEL プロジェクト・ファイルをローカル・マシンにダウンロードできます。参照: コンポジット・サービス - BPEL のダウンロード、ページ 10-2。

コンポジット・サービス - BPEL のダウンロード

統合開発者は、コンポジット・サービス - BPEL の詳細を表示したり WSDL の要約をレビューする以外に、JAR ファイルに集約されたコンポジット・サービス関連ファイルをローカル・ディレクトリにダウンロードできます。

重要: 通常、統合開発者および統合管理者のみが、コンポジット・サービス - BPEL をダウンロードできます。ただし、「統合リポジトリ・コンポジット・サービスのダ

ダウンロード」権限セット(FND_REP_DOWNLOAD_PERM_SET)を介してダウンロード権限が付与されたユーザーも、ダウンロード・アクションを実行できます。それ以外の場合、「サービスのダウンロード」は、デフォルトで詳細ページに表示されない可能性があります。

コンポジット・サービスのダウンロード権限の付与方法の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「ロール・ベースのアクセス管理(RBAC)セキュリティ」を参照してください。

コンポジット・サービス - BPEL の .ZIP ファイルをダウンロードするには、ダウンロードするサービスのコンポジット・サービス - BPEL の詳細ページにナビゲートし、「サービスのダウンロード」をクリックして、ファイルをローカル・ディレクトリにダウンロードします。

ダウンロード後、BPEL .JAR ファイルを解凍し、Oracle JDeveloper 10g で BPEL プロセスを開いて、必要に応じてサービス・エンドポイントを変更できます。BPEL プロセスを変更してデプロイする方法の詳細は、「BPEL プロセスの変更およびデプロイ」(ページ 10-3)を参照してください。

コンポジット・サービス - BPEL をダウンロードするには:

1. 「統合開発者」ロールを持つユーザーとして、Oracle E-Business Suite にログインします。ナビゲーション・メニューから、「統合 SOA ゲートウェイ」職責を選択して、「統合リポジトリ」リンクを選択します。
2. 「統合リポジトリ」タブで、「表示別」ドロップダウン・リストから「インタフェース・タイプ」を選択します。
3. 「コンポジット - BPEL」インタフェース・タイプ・ノードを展開して、目的のコンポジット・サービス - BPEL を見つけます。
4. ダウンロードするコンポジット・サービス - BPEL をクリックして、コンポジット・サービス - BPEL インタフェースの詳細ページを開きます。
5. 「サービスのダウンロード」をクリックして、選択したコンポジット BPEL ファイルをローカル・ディレクトリにダウンロードします。

BPEL プロセスの変更およびデプロイ

コンポジット・サービス BPEL プロジェクトのダウンロード後、統合開発者は、必要に応じて BPEL プロジェクトを変更できます。これを実行するには、まず BPEL .JAR ファイルを解凍し、Oracle JDeveloper 10g で BPEL ファイルを開き、必要に応じて BPEL プロセスのエンドポイントを変更します。変更後、BPEL プロジェクトを Oracle SOA Suite 10g の BPEL サーバーにデプロイできます。

注意: Oracle SOA Suite 11g および Oracle SOA Suite 12c では、BPEL プロセスは、関連する SOA コンポジット・アプリケーションとともに管理およびデプロイされます。

コンポジット・サービス - BPEL タイプは、Oracle SOA Suite 10g でサポートされています。BPEL プロセスは、Oracle SOA Suite 10g の BPEL サーバーまたは J2EE 環境のサード・パーティ BPEL PM にデプロイされます。

BPEL プロセスの変更では、BPEL プロセスの作成時と同様のロジックを使用します。このマニュアルですでに説明した各インタフェース・タイプの設計時タスクを参照してください。コンポジット・アプリケーション内の BPEL プロセスをテストおよび検証する方法の詳細は、このマニュアルで説明するインタフェース・タイプの実行時タスクを参照してください。

この項で説明する BPEL プロセスの変更およびデプロイでは、Oracle JDeveloper 10g (10.1.3.3.0)を設計時ツールとして使用して BPEL プロセスを変更し、Oracle SOA Suite BPEL サーバー10.1.3.3.0を使用してこのプロセスをデプロイします。

BPEL プロセスを変更するには:

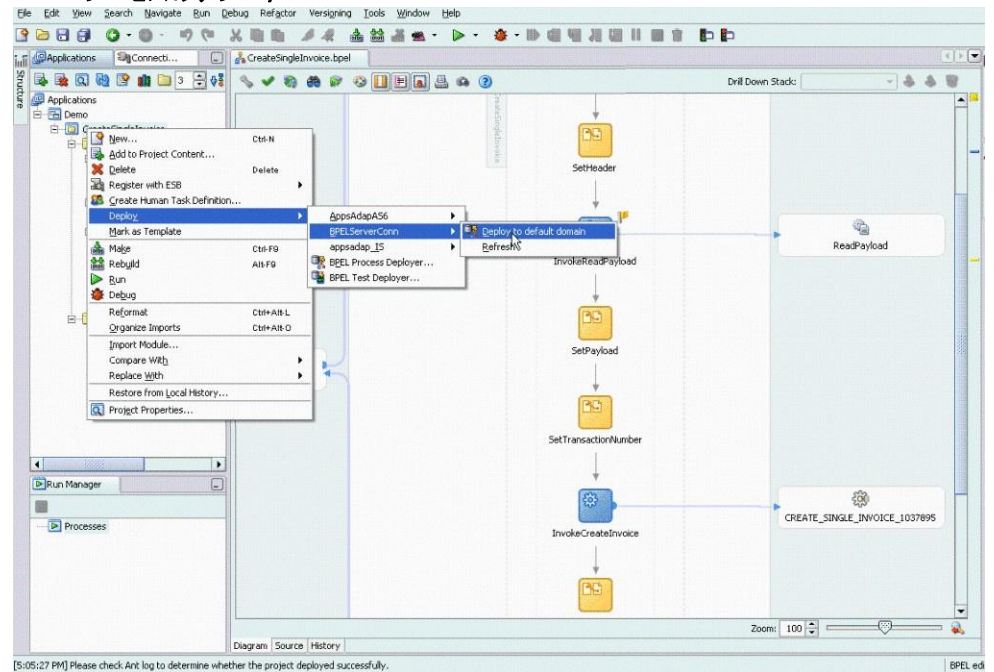
1. Oracle JDeveloper 10g BPEL デザイナで、BPEL ファイルを開きます。
2. 「ファイル」メニューから「開く」を選択します。
3. 変更するディレクトリから BPEL ファイルを見つけます。「開く」ウィンドウで、「開く」をクリックします。
4. 選択した BPEL プロセス・ダイアグラムが表示されます。
5. 必要に応じて、BPEL プロセスのエンドポイントを変更します。
6. 作業内容を保存します。

BPEL プロセスをデプロイするには:

1. Oracle JDeveloper 10g BPEL デザイナのアプリケーション・ナビゲータで、デプロイする BPEL プロジェクトを選択します。
2. プロジェクトを右クリックし、メニューから「デプロイ」アクションを選択します。「デプロイメント・ツールの起動」をクリックして、BPEL Process Manager の情報を入力します。

たとえば、BPEL Process Manager が適切に設定されている場合は、「デプロイ」→「BPEServerConn」→デフォルト・ドメインへのデプロイを選択してプロジェクトをデプロイできます。

BPEL プロセスのデプロイ



3. 「パスワード・プロンプト」ダイアログ・ボックスが表示されます。「ドメイン・パスワード」フィールドにデフォルト・ドメインのパスワードを入力して、「OK」をクリックします。BPEL プロジェクトがコンパイルされ、正常にデプロイされます。

カスタム統合インタフェースの作成および使用

概要

カスタム統合インタフェースおよびサービスは、統合リポジトリ・ブラウザ・ウィンドウと Oracle E-Business Suite の Oracle シード・インタフェースから表示できます。

統合開発者は、サポートされているインタフェース・タイプ(XML ゲートウェイ・マップ、ビジネス・イベント、PL/SQL、コンカレント・プログラム、ビジネス・サービス・オブジェクト、Java Bean サービス、アプリケーション・モジュール・サービス、Java API、コンポジット・サービス - BPEL タイプなど)の統合リポジトリ・アノテーションの標準に基づいて、カスタム統合インタフェースを作成して注釈付けします。

注意: EDI、オープン・インタフェース表、オープン・インタフェース・ビュー、および フォーム用 Java API の各インタフェースのカスタム・インタフェース・タイプは、このリリースではサポートされていないことに注意してください。

Oracle Integration Repository は、現時点で、カスタム製品ファミリーやカスタム・ビジネス・エンティティの作成には対応していません。

これらのカスタム・インタフェースが注釈付けされた後、統合管理者は、スタンドアロンの設計時ツールを使用して、これらの注釈付きソース・ファイルをアノテーションの標準に対して検証します。検証後、ローダー・ファイルが生成され、バックエンド処理を介して統合リポジトリにアップロードされます。これらのカスタム・インタフェースは、これらが属するインタフェース・タイプに基づいて表示され、統合リポジトリ・ユーザー・インタフェースの Oracle シード・インタフェースとともに表示されます。

インタフェース・タイプのカスタム統合インタフェースの場合

サポートされているインタフェース・タイプ用に作成されたカスタム・インタフェースが Oracle Integration Repository にアップロードされている場合に、このカスタム・インタフェースを使用するには、統合管理者はまず必要なセキュリティ付与を作成し、このカスタム・インタフェース・タイプでサービスを有効にできる場合は、Web サービスを生成してアプリケーション・サーバーに配置する必要があります。こうして、配置されたサービスは、サービス・プロバイダを介して顧客に公開され、Web サービス・クライアントまたはオーケストレーション・ツールによって

起動できます。

カスタム・コンポジット・サービス - BPEL タイプの場合

コンポジット・サービス - BPEL タイプにカスタム・インタフェースが必要な場合、統合開発者は、まず、Oracle JDeveloper 10g の BPEL プロセス・コンポーネントを使用して、個別のネイティブ・サービスを有用な BPEL プロセス・フローにオーケストレーションしてコンポジット・サービスを作成します。統合開発者は、コンポジット・サービス用のアノテーションの標準に基づいて、コンポジット・サービス - BPEL タイプに注釈付けします。他のインタフェース・タイプのカスタム・インタフェースと同様に、BPEL プロジェクト JAR ファイルを統合リポジトリにアップロードする前に適切な検証を行う必要があります。

カスタム・インタフェースの作成方法、およびカスタム・インタフェースを Web サービスとして使用する方法についてさらに理解を深めるために、この章では次のトピックについて説明します。

- カスタム統合インタフェースの作成、ページ 11-2
- Web サービスとしてのカスタム統合インタフェースの使用、ページ 11-23

カスタム統合インタフェースの作成

この項では次のトピックについて説明します。

- ネイティブ・インタフェース・タイプのカスタム統合インタフェースの作成、ページ 11-2
- カスタム・コンポジット・サービスの作成、ページ 11-9
- ワークフロー XML ロダーを使用したカスタム・ビジネス・イベントの作成、ページ 11-15

ネイティブ・インタフェース・タイプのカスタム統合インタフェースの作成

ほぼすべてのインタフェース・タイプのカスタム・インタフェース定義を作成して、注釈付けすることができます。適切な検証を行い、エラーが発生しなかった場合は、生成された iLDT ファイルにコンパイルされた検証済カスタム定義ソースを、バックエンド処理を介して Oracle Integration Repository にアップロードできます。

注意: EDI、オープン・インタフェース表およびオープン・インタフェース・ビューのカスタム・インタフェース・タイプは、このリリースではサポートされていないことに注意してください。

Oracle Integration Repository は、現時点で、カスタム製品ファミリやカスタム・ビジネス・エンティティの作成には対応していません。

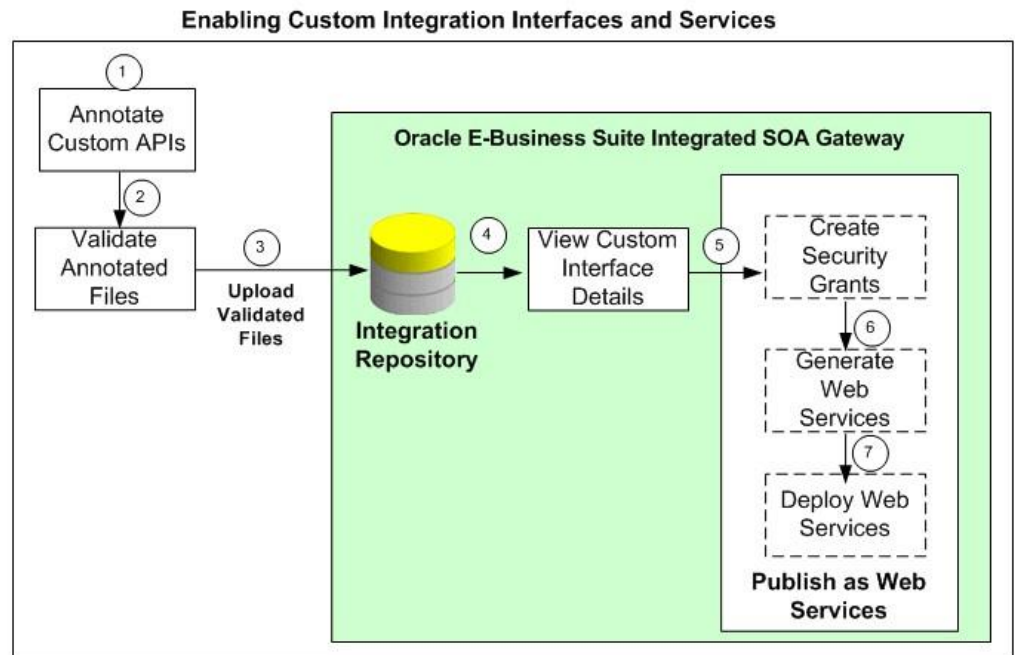
カスタム統合インタフェースの有効化

次の図は、カスタム・インタフェースの設計およびサービスの有効化プロセス・フローを示しています。

注意: すべての統合インタフェース定義でサービスを有効にできるわけではありません。Oracle Integration Repository では、次のインタフェース・タイプでのみサービスの有効化がサポートされています。

- PL/SQL
- XML ゲートウェイ・マップ(インバウンド)
- コンカレント・プログラム
- ビジネス・サービス・オブジェクト(サービス Bean)
- Java Bean サービス
- アプリケーション・モジュール・サービス

ビジネス・イベントおよび XML ゲートウェイ・マップ(アウトバウンド)のインタフェース・タイプは、サブスクリプション・モデルを介してサポートされることに注意してください。



1. 「統合開発者」ロールを持つユーザーは、サポートされているインタフェース・タイプの統合リポジトリ・アノテーションの標準に基づいて、カスタム統合インタフェース定義に注釈付けします。

参照: カスタム統合インタフェースの作成および注釈付け、ページ 11-5。

注意: カスタム・スキーマで作成されたカスタム PL/SQL API (単純なデータ型のみ)の場合は、こうしたカスタム API を Oracle Integration Repository で公開できます。また、カスタム・スキーマのこうした API に対して次のタスクを実行します。

1. APPS スキーマへのアクセス権限を付与します。

1. sysdba としてカスタム・スキーマに接続します。sqlplus '/ as sysdba '

2. 次のコマンドを使用して、アクセス権限を付与します。

```
GRANT EXECUTE on <custom_schema>.  
<custom_package> TO APPS;
```

2. カスタム・ストアド・プロシージャのシノニムを作成します。

1. 次のコマンドを使用して、APPS スキーマに接続します。

```
sqlplus <APPS Username> Enter password: <Password>
```

2. 次のコマンドを使用して、シノニムを作成します。

```
CREATE SYNONYM <custom_package> FOR  
<custom_schema>.<custom_package>;
```

2. 「統合管理者」ロールを持つユーザーは、注釈付きカスタム・インタフェース定義をアノテーションの標準に対して検証します。この検証では、設計時ツールの 1 つである統合リポジトリ・パーサー (IREP パーサー) を実行することで、注釈付きファイルを読み取り、エラーが発生しなければ、統合リポジトリ・ローダー・ファイル (iLDT) を生成します。

iLDT ファイルを生成してアップロードする方法の詳細は、「iLDT ファイルの生成およびアップロード」(ページ 11-8) を参照してください。

3. 「統合管理者」ロールを持つユーザーは、生成された iLDT ファイルを Oracle Integration Repository にアップロードします。

参照: *Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド* の「iLDT ファイルの統合リポジトリへのアップロード」。

4. すべてのユーザーは、統合リポジトリ・ユーザー・インタフェースから、アップロードされたカスタム・インタフェースを表示できます。
5. (オプション) 「統合管理者」ロールを持つユーザーは、必要に応じて、カスタム統合インタフェースに必要なセキュリティ付与を作成します。

これを実行するには、まず統合リポジトリからカスタム・インタフェースを見つけて、選択したカスタム・インタフェースに含まれるメソッドを選択してから、「付与の作成」をクリックします。「付与の作成」ページが表示され、管理者は選択したメソッドへのアクセス権限を

ユーザー、ユーザー・グループまたはすべてのユーザーに付与できます。参照: *Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「セキュリティ付与の作成」。

6. (オプション)「統合管理者」ロールを持つユーザーは、カスタム・インタフェースでサービスを有効にできる場合は、Web サービスを生成できます。

これを実行するには、まずカスタム・インタフェースを見つけて、インタフェース・レベルまたはメソッド・レベルで相互作用パターンを指定してから、選択したカスタム・インタフェースの詳細ページで「生成」をクリックします。参照: *Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「カスタム SOAP Web サービスの生成」。

7. (オプション)「統合管理者」ロールを持つユーザーは、Oracle Integration Repository からアプリケーション・サーバーに Web サービスを配置します。

生成された SOAP Web サービスを配置するには、管理者はまず選択した Web サービスごとに 1 つの認証タイプ(ユーザー名トークンまたは SAML トークン)を選択してから、選択したインタフェースの詳細ページで「配置」をクリックする必要があります。これにより、生成されたアクティブ状態のサービスが Oracle SOA Suite に配置され、Oracle E-Business Suite のサービスを、実行時にサービス実行の標準 Web サービスとして公開できます。参照: *Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「カスタム SOAP Web サービスの配置および配置解除」。

カスタム・インタフェースを REST サービスとして公開できる場合、管理者はサービスを配置する前に、選択したカスタム・インタフェースごとに一意のサービス別名を入力し、目的のサービス操作を指定する必要があります。また、インタフェース・タイプが Java Bean サービスまたはアプリケーション・モジュール・サービスの場合、管理者は選択したインタフェースに含まれるサービス操作の HTTP メソッドを指定する必要があります。

注意: オープン・インタフェース表およびオープン・インタフェース・ビューは REST サービスとして公開できますが、カスタム・オープン・インタフェース表およびカスタム・オープン・インタフェース・ビューはこのリリースではサポートされていません。

参照: *Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「カスタム REST Web サービスの配置」。

カスタム統合インタフェースのアノテーションの例

特定のインタフェース・タイプのカスタム統合定義が作成されると、統合開発者は、統合リポジトリ・アノテーションの標準に基づいてカスタム・ファイルに適切に注釈付けして、ブラウザ・インタフェースからカスタム・インタフェースと適切な説明を表示できるようにする必要があります。

たとえば、統合開発者は PL/SQL API を使用して、仕入先出荷およびデビット要求カスタム・インタフェースを作成できます。このカスタム PL/SQL API パッケージの仕様ファイル

(zz_sdrequest_s.pls)は次のとおりです。

```
set verify off
whenever sqlerror exit failure rollback;
WHENEVER OSERROR EXIT FAILURE ROLLBACK;

create or replace package ZZ_SDREQUEST as
/* $Header: zz_sdrequest_s.pls $ */

-- Custom procedure to create single supplier ship and debit request

procedure ZZ_CREATE_SDREQUEST (
  CP_API_VERSION_NUMBER IN NUMBER,
  CP_INIT_MSG_LIST IN VARCHAR2 := FND_API.G_FALSE,
  CP_COMMIT IN VARCHAR2 := FND_API.G_FALSE,
  CP_VALIDATION_LEVEL IN NUMBER := FND_API.G_VALID_LEVEL_FULL,
  CX_RETURN_STATUS OUT VARCHAR2,
  CX_MSG_COUNT OUT NUMBER,
  CX_MSG_DATA OUT VARCHAR2,
  CP_SDR_HDR_REC IN OZF_SD_REQUEST_PUB.SDR_HDR_REC_TYPE,
  CP_SDR_LINES_REC IN OZF_SD_REQUEST_PUB.SDR_lines_rec_type,
  CP_SDR_CUST_REC IN OZF_SD_REQUEST_PUB.SDR_cust_rec_type,
  CP_SDR_BILLTO_REC IN OZF_SD_REQUEST_PUB.SDR_cust_rec_type,
  CX_REQUEST_HEADER_ID OUT NUMBER
)
;
end ZZ_SDREQUEST;

/
commit;
exit;
```

統合開発者は、PL/SQL API アノテーションの標準に基づいて、次の場所にアノテーション情報を追加することにより、仕入先出荷およびデビット要求カスタム・パッケージの仕様ファイルに注釈付けする必要があります。

- PL/SQL API パッケージの仕様に注釈付けします
- PL/SQL プロシージャに注釈付けします

プロシージャのアノテーションは、定義と;の間に配置する必要があります。

カスタム・パッケージの仕様ファイルにのみ注釈付けする必要があり、パッケージの本体ファイルには注釈付けする必要がないことに注意してください。Oracle Integration Repository でサポートされているインタフェース・タイプのカスタム・インタフェースに注釈付けする方法の詳細は、「統合リポジトリ・アノテーションの標準」(ページ A-1)および My Oracle Support のナレッジ・ドキュメント 1315485.1 から入手できる *Oracle Application Framework 開発者ガイド*を参照してください。


```

set verify off
whenever sqlerror exit failure rollback;
WHENEVER OSERROR EXIT FAILURE ROLLBACK;

create or replace package ZZ_SDREQUEST as
/* $Header: zz_sdrequest_s.pls $ */
/**
* This custom PL/SQL package can be used to create supplier ship and
debit request for single product.
* @rep:scope public
* @rep:product OZF
* @rep:displayname Single ship and debit request
* @rep:category BUSINESS_ENTITY OZF_SSD_REQUEST
*/

-- Custom procedure to create single supplier ship and debit request

procedure ZZ_CREATE_SDREQUEST (
CP_API_VERSION_NUMBER IN NUMBER,
CP_INIT_MSG_LIST IN VARCHAR2 := FND_API.G_FALSE,
CP_COMMIT IN VARCHAR2 := FND_API.G_FALSE,
CP_VALIDATION_LEVEL IN NUMBER := FND_API.G_VALID_LEVEL_FULL,
CX_RETURN_STATUS OUT VARCHAR2,
CX_MSG_COUNT OUT NUMBER,
CX_MSG_DATA OUT VARCHAR2,
CP_SDR_HDR_REC IN OZF_SD_REQUEST_PUB.SDR_HDR_REC_TYPE,
CP_SDR_LINES_REC IN OZF_SD_REQUEST_PUB.SDR_lines_rec_type,
CP_SDR_CUST_REC IN OZF_SD_REQUEST_PUB.SDR_cust_rec_type,
CP_SDR_BILLTO_REC IN OZF_SD_REQUEST_PUB.SDR_cust_rec_type,
CX_REQUEST_HEADER_ID OUT NUMBER
)
/**
* Use this procedure to create single supplier ship and debit request
* @param CP_API_VERSION_NUMBER Version of the custom API
* @param CP_INIT_MSG_LIST Flag to initialize the message stack
* @param CP_COMMIT Indicates Flag to commit within the program
* @param CP_VALIDATION_LEVEL Indicates the level of the validation
* @param CX_RETURN_STATUS Indicates the status of the program
* @param CX_MSG_COUNT Provides the number of the messages returned by
the program
* @param CX_MSG_DATA Returns messages by the program
* @param CP_SDR_HDR_REC Contains details of the new Ship Debit Request
to be created
* @param CP_SDR_LINES_REC Contains the product line information for
the new Ship Debit Request
* @param CP_SDR_CUST_REC Contains the Customer information for the new
Ship Debit Request
* @param CP_SDR_BILLTO_REC Contains the Bill-to information for the
new Ship Debit Request
* @param CX_REQUEST_HEADER_ID Returns the id of the new Ship Debit
Request created
* @rep:displayname Create ship and debit request
* @rep:category BUSINESS_ENTITY OZF_SSD_REQUEST
* @rep:scope public
* @rep:lifecycle active
*/
;
end ZZ_SDREQUEST;

/
commit;
exit;

```

iLDT ファイルの生成およびアップロード

注釈付きカスタム統合インタフェース定義が作成されたら、Oracle Integration Repository にアップロードする前に、これらの注釈付きソース・ファイルをアノテーションの標準に対して検証する必要があります。この検証では、設計時ツールの 1 つである統合リポジトリ・パーサー (IREP パーサー) を実行することで、注釈付きファイルを読み取り、エラーが発生しなければ、統合リポジトリ・ローダー・ファイル (iLDT) を生成します。

注意: 統合リポジトリ・パーサーは、カスタム・アプリケーションに登録された統合インタフェースには対応していないことに注意してください。

現時点で、Linux、Unix、Oracle Solaris on SPARC、HP-UX Itanium および IBM AIX on Power Systems に対してはテストされ、動作が確認されています。

Microsoft Windows プラットフォームは、現時点でこのリリースではサポートされていません。

iLDT ファイルが生成されると、統合管理者は生成されたファイルを Oracle Integration Repository にアップロードして、カスタム・インタフェースをすべてのユーザーに公開できます。

統合リポジトリ・パーサーを設定および使用して iLDT ファイルを生成する方法、および生成されたファイルをリポジトリにアップロードする方法の詳細は、次を参照してください。

- *Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド* の「統合リポジトリ・パーサーの設定および使用」
- *Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド* の「iLDT ファイルの生成」
- *Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド* の「iLDT ファイルの統合リポジトリへのアップロード」

カスタム・インタフェースの表示および管理タスクの実行

カスタム・インタフェースの検索および表示

注釈付きカスタム・インタフェース定義が正常にアップロードされると、それらは属しているインタフェース・タイプにマージされ、統合リポジトリ・ブラウザ・ウィンドウの Oracle シード・インタフェースとともに表示されます。注釈付きカスタム・インタフェース定義を Oracle のインタフェース定義と簡単に区別するには、インタフェース・ソース「カスタム」を使用して、これらのカスタム・インタフェースを分類します (Oracle E-Business Suite の Oracle シード・インタフェースの場合は、インタフェース・ソース「Oracle」)。

カスタム統合インタフェースを検索するには、次の方法のいずれかを使用できます。

- 「インタフェース・リスト」ページで、「インタフェース・ソース」ドロップダウン・リストから「カスタム」を選択し、「範囲」フィールドに値を指定して、カスタム統合インタフェースの表示を絞り込みます。

- 「検索」ページで、「**詳細検索オプションの表示**」をクリックして、「インタフェース・ソース」ドロップダウン・リストから「カスタム」を選択し、必要に応じて検索基準としてインタフェース・タイプ、製品ファミリまたは範囲を選択します。

検索を実行すると、一致するカスタム統合インタフェースがすべて表示されます。カスタム統合インタフェースの検索および表示方法の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「カスタム統合インタフェースの検索」、および *Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「カスタム統合インタフェースの表示」を参照してください。

管理タスクの実行

カスタム統合インタフェースが正常にアップロードされ、統合リポジトリ・ユーザー・インタフェースから表示されるようになると、統合管理者はこれらのカスタム・インタフェースで、ネイティブ統合インタフェースの場合と同じ管理タスクを実行できます。これらの管理タスクには、新しく作成されたカスタム・インタフェースのセキュリティ付与の作成(必要な場合)、Web サービスの生成、Web サービスの配置、および配置ライフサイクル全体にわたるサービスの管理が含まれます。*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「カスタム統合インタフェースおよびサービスの管理」を参照してください。

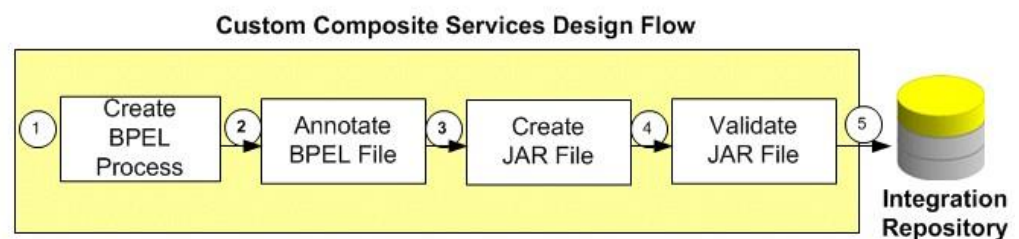
カスタム統合インタフェースを Web サービスとして使用方法の詳細は、「Web サービスとしてのカスタム統合インタフェースの使用」(ページ 11-23)を参照してください。

カスタム・コンポジット・サービス - BPEL の作成

統合開発者は、BPEL 言語を使用して、個別の Web サービスを有用なビジネス・プロセスにオーケストレーションすることにより、新しいコンポジット・サービスを作成できます。コンポジット・サービス - BPEL タイプ用の適切な注釈付け、およびアノテーションの標準に対する検証を行い、エラーが発生しなかった場合は、検証されたコンポジット・サービス BPEL プロジェクトの iLDT ファイルを生成できます。生成された iLDT ファイルは、バックエンド処理を介して統合リポジトリにアップロードできます。

カスタム・コンポジット・サービスの作成

次の図は、コンポジット・サービス - BPEL タイプのカスタム統合インタフェースの設計フローを示しています。



1. 統合開発者は、Oracle JDeveloper 10g の BPEL プロセス・コンポーネントを使用して、コンポジット・サービスをオーケストレートします。
2. 統合開発者は、コンポジット・サービス - BPEL タイプ用の統合リポジトリ・アノテーションの標準に基づいて、コンポジット・サービスに注釈付けします。
参照: カスタム・コンポジット・サービス - BPEL の作成および注釈付け、ページ 11-10。
3. 統合開発者は、コンポジット・サービス - BPEL プロジェクトの JAR ファイルを作成します。
4. 統合管理者は、まず JAR ファイルを解凍し、注釈付きカスタム・インタフェース定義を、コンポジット・サービス - BPEL 用のアノテーションの標準に対して検証します。この検証では、統合リポジトリ・パーサーを実行することで、注釈付きファイルを読み取り、エラーが発生しなければ、統合リポジトリ・ローダー・ファイル(iLDT)を生成します。
5. 統合管理者は、生成された iLDT ファイルをバックエンド処理を介して Oracle Integration Repository にアップロードします。

参照: iLDT ファイルの生成およびアップロード、ページ 11-13。

アップロード後、統合リポジトリ・ユーザー・インタフェースから、カスタム・コンポジット・サービス - BPEL を確認します。

カスタム統合インタフェース定義が正常にアップロードされ、統合リポジトリ・ブラウザから表示されるようになると、統合管理者および統合開発者は、必要に応じて、コンポジット・サービス - BPEL をダウンロードして変更できます。コンポジット・サービス - BPEL のダウンロード方法の詳細は、「カスタム・コンポジット・サービス - BPEL の表示およびダウンロード」(ページ 11-14)を参照してください。

カスタム・コンポジット・サービスのアノテーションの例

カスタム統合インタフェースの作成における重要な本質は、新しいインタフェース機能または定義の適切な説明に依存しています。カスタム・コンポジット・サービス - BPEL 定義が作成されると、統合開発者は、統合リポジトリ・アノテーションの標準に基づいてカスタム・ソース・ファイルに適切に注釈付けして、ブラウザ・インタフェースからソース・ファイルと適切な説明を表示できるようにする必要があります。

たとえば、請求書の作成コンポジット - BPEL プロジェクトが作成されます。コンポジット・サービスに注釈付けするには、テキスト・エディタで*.bpe1 ファイルを開き、ここで強調表示されているように、ファイルの先頭のコメント・セクション内に注釈を配置します。

```

////////////////////////////////////
Oracle JDeveloper BPEL Designer

Created: Tue Oct 30 17:10:13 IST 2007
Author: jdole
Purpose: Synchronous BPEL Process
/*#
* This is a bpel file for creating invoice.
* @rep:scope public
* @rep:displayname Create Invoice
* @rep:lifecycle active
* @rep:product PO
* @rep:compatibility S
* @rep:interface oracle.apps.po.CreateInvoice
* @rep:category BUSINESS_ENTITY INVOICE
*/

////////////////////////////////////

-->
<process name="CreateInvoice">
  targetNamespace="http://xmlns.oracle.com/CreateInvoice"
  xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-
process/"
  xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.
tip.pc.services.functions.Xpath20"
  xmlns:ns4="http://xmlns.oracle.
com/pcbpel/adaptor/file/ReadPayload/"
  xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:ns5="http://xmlns.oracle.com/bpel/workflow/xpath"
  xmlns:client="http://xmlns.oracle.com/CreateInvoice"
  xmlns:ns6="http://xmlns.oracle.
com/bpel/services/IdentityService/xpath"
  xmlns:ora="http://schemas.oracle.com/xpath/extension"
  xmlns:ns1="http://xmlns.oracle.
com/soapprovider/plsql/AR_INVOICE_API_PUB_2108/CREATE_SINGLE_INVOICE_10
37_895/"
  xmlns:ns3="http://xmlns.oracle.
com/soapprovider/plsql/AR_INVOICE_API_PUB_2108/APPS/BPEL_CREATE_SINGLE
IN VOICE_1037895/AR_INVOICE_API_PUB-24CREATE_INV/"
  xmlns:ns2="http://xmlns.oracle.com/pcbpel/adaptor/appscontext/"
  xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
  xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc
. services.functions.ExtFunc">

  <!--
  //////////////////////////////////
  PARTNERLINKS
    List of services participating in this BPEL process
  //////////////////////////////////
  -->
  <partnerLinks>
    <!--
    The 'client' role represents the requester of this service.It is
    used for callback.The location and correlation information
    associated
    with the client role are automatically set using WS-Addressing.
    -->
    <partnerLink name="client" partnerLinkType="client:CreateInvoice"
myRole="CreateInvoiceProvider"/>
    <partnerLink name="CREATE_SINGLE_INVOICE_1037895"
partnerRole="CREATE_SINGLE_INVOICE_1037895_ptt_Role"
partnerLinkType="ns1:
CREATE_SINGLE_INVOICE_1037895_ptt_PL"/>

```

```

<partnerLink name="ReadPayload" partnerRole="SynchRead_role"
    partnerLinkType="ns4:SynchRead_plt"/>
</partnerLinks>
<!--
////////////////////////////////////
VARIABLES
    List of messages and XML documents used within this BPEL process
////////////////////////////////////
-->
<variables>
<!--Reference to the message passed as input during initiation-->
    <variable name="inputVariable"
        messageType="client:CreateInvoiceRequestMessage"/>
<!--Reference to the message that will be returned to the requester-->
    <variable name="outputVariable"
        messageType="client:CreateInvoiceResponseMessage"/>
    <variable name="
Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
        messageType="ns1:Request"/>
    <variable name="
Invoke_1_CREATE_SINGLE_INVOICE_1037895_OutputVariable"
        messageType="ns1:Response"/>
    <variable name="Invoke_2_SynchRead_InputVariable"
        messageType="ns4:Empty_msg"/>
    <variable name="Invoke_2_SynchRead_OutputVariable"
        messageType="ns4:InputParameters_msg"/>
</variables>
<!--
////////////////////////////////////
ORCHESTRATION LOGIC
    Set of activities coordinating the flow of messages across the
    services integrated within this business process
////////////////////////////////////
-->
<sequence name="main">
    <!--Receive input from requestor.(Note: This maps to operation
defined in CreateInvoice.wsdl)-->
    <receive name="receiveInput" partnerLink="client"
        portType="client:CreateInvoice" operation="process"
        variable="inputVariable" createInstance="yes"/>
    <!--Generate reply to synchronous request-->
    <assign name="SetHeader">
        <copy>
            <from expression="'operations'">
                <to variable="
Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
                    part="header"
                    query="/ns1:SOAHeader/ns2:ProcedureHeaderType/ns2:Username"
                />
            </copy>
            <copy>
                <from expression="'Receivables, Vision Operations (USA)'">
                    <to variable="
Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
                        part="header"
                        query="/ns1:SOAHeader/ns2:ProcedureHeaderType/ns2:
Responsibility"/>
                </copy>
                <copy>
                    <from expression="'204'">
                        <to variable="
Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
                            part="header"
                            query="/ns1:SOAHeader/ns2:ProcedureHeaderType/ns2:ORG_ID"/>
                    </copy>

```

```

        <copy>
<from expression="'Receivables, Vision Operations (USA)'">
    <to variable="
Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
    part="header"
    query="/ns1:SOAHeader/ns1:SecurityHeader/ns1:
ResponsibilityName"/>
    </copy>
</assign>
<invoke name="InvokeReadPayload" partnerLink="ReadPayload"
    portType="ns4:SynchRead_ptt" operation="SynchRead"
    inputVariable="Invoke_2_SynchRead_InputVariable"
    outputVariable="Invoke_2_SynchRead_OutputVariable"/>
<assign name="SetPayload">
    <copy>
        <from variable="Invoke_2_SynchRead_OutputVariable"
            part="InputParameters" query="/ns3:InputParameters"/>
        Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
        part="body" query="/ns1:SOARequest/ns3:InputParameters"/>
    </copy>
</assign>
<assign name="SetDate">
    <copy>
        <from expression="xp20:current-date()">
        <to to variable="
Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
        part="body"
        query="/ns1:SOARequest/ns3:InputParameters/ns3:
P_TRX_HEADER_TBL/ns3:P_TRX_HEADER_TBL_ITEM/ns3:TRX_DATE"/>
    </copy>
</assign>
<invoke name="Invoke_1" partnerLink="CREATE_SINGLE_INVOICE_1037895"
    portType="ns1:CREATE_SINGLE_INVOICE_1037895_ptt"
    operation="CREATE_SINGLE_INVOICE_1037895"
    inputVariable="
Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
    outputVariable="
Invoke_1_CREATE_SINGLE_INVOICE_1037895_OutputVariable"/>
    <assign name="AssignResult">
        <copy>
            <from variable="
Invoke_1_CREATE_SINGLE_INVOICE_1037895_OutputVariable"
            part="body"
            query="/ns1:SOAResponse/ns3:OutputParameters/ns3:
X_MSG_DATA"/>
            <to variable="outputVariable" part="payload"
            query="/client:CreateInvoiceProcessResponse/client:result"/>
        </copy>
    </assign>
<reply name="replyOutput" partnerLink="client"
    portType="client:CreateInvoice" operation="process"
    variable="outputVariable"/>
</sequence>
</process>

```

コンポジット・サービス - BPEL タイプに注釈付けする方法の詳細は、「コンポジット・サービス - BPEL アノテーション」(ページ A-115)を参照してください。

ilDT ファイルの生成およびアップロード

注釈付きカスタム・コンポジット・サービス - BPEL が作成されたら、Oracle Integration Repository にアップロードする前に、これらの注釈付きソース・ファイルを、コンポジット・サー

ビス - BPEL タイプ用のアノテーションの標準に対して検証する必要があります。この検証では、設計時ツールの 1 つである統合リポジトリ・パーサー (IREP パーサー) を実行することで、注釈付きファイルを読み取り、エラーが発生しなければ、統合リポジトリ・ローダー・ファイル (iLDT) を生成します。

注意: 統合リポジトリ・パーサーは、カスタム・アプリケーションに登録された統合インタフェースには対応していないことに注意してください。

現時点で、Linux、Unix、Oracle Solaris on SPARC、HP-UX Itanium および IBM AIX on Power Systems に対してはテストされ、動作が確認されています。

Microsoft Windows プラットフォームは、現時点でこのリリースではサポートされていません。

iLDT ファイルが正常に生成されると、統合管理者は生成されたファイルを Oracle Integration Repository にアップロードして、カスタム・インタフェースをすべてのユーザーに公開できます。

統合リポジトリ・パーサーを設定および使用して iLDT ファイルを生成する方法、および生成された iLDT ファイルをアップロードする方法の詳細は、次を参照してください。

- *Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド* の「統合リポジトリ・パーサーの設定および使用」
- *Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド* の「iLDT ファイルの生成」
- *Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド* の「iLDT ファイルの統合リポジトリへのアップロード」

カスタム・コンポジット・サービス - BPEL の表示およびダウンロード

注釈付きカスタム・コンポジット・サービス定義が統合リポジトリ・ユーザー・インタフェースに正常にアップロードされると、それらはコンポジット・サービス BPEL タイプにマージされ、Oracle シード・インタフェースとともに表示されます。カスタム・コンポジット・サービスを Oracle シード・サービスと簡単に区別するには、インタフェース・ソース「カスタム」を使用して、これらのカスタム・インタフェースを分類します (Oracle E-Business Suite の Oracle シード・インタフェースの場合は、インタフェース・ソース「Oracle」)。

カスタム・コンポジット・サービスを検索するには、「検索」ページで「詳細検索オプションの表示」をクリックして、検索基準を展開します。「インタフェース・ソース」ドロップダウン・リストから「カスタム」を選択し、必要に応じて検索基準として「コンポジット・サービス」インタフェース・タイプ、製品ファミリまたは範囲を選択します。検索を実行すると、一致するカスタム・コンポジット・サービスがすべて表示されます。

カスタム・コンポジット・サービス - BPEL のダウンロード

ネイティブ・パッケージ・コンポジット・サービスのダウンロードと同様に、統合管理者および統合開発者は、コンポジット・サービス - BPEL インタフェースの詳細ページで「サービスのダウンロード」をクリックして、JAR ファイルに集約された関連するカスタム・コンポジット・ファイルをローカル・ディレクトリにダウンロードできます。

カスタム・コンポジット・サービス - BPEL を検索してダウンロードする方法の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「コンポジット・サービス - BPEL のダウンロード」を参照してください。

ワークフローXML ローダーを使用したカスタム・ビジネス・イベントの作成

Oracle E-Business Suite Integrated SOA Gateway では、ワークフローXML ローダーを使用して、ビジネス・イベント・システムでカスタム・ビジネス・イベントを作成し、作成したイベントをダウンロードし、イベント・ソース・コードに注釈付けし、ソース・ファイルを検証して、イベント・システムにファイルをアップロードできます。

ワークフローXML ローダーは、データベースとフラット・ファイルの間でビジネス・イベント・システム・オブジェクトの XML 定義をアップロードおよびダウンロードできるコマンドライン・ユーティリティです。ビジネス・イベント・システム・オブジェクト定義をデータベースからダウンロードすると、Oracle Workflow によりこの定義は XML ファイルとして保存されます。オブジェクト定義をデータベースにアップロードすると、Oracle Workflow によりこの定義がソース XML ファイルからデータベース内のビジネス・イベント・システム表にロードされるため、必要に応じて新しい定義を作成するか、既存の定義を更新します。

ワークフローXML ローダーによってアップロードまたはダウンロードされた XML ファイルには、拡張子 .wfx を付けて、ワークフロー・オブジェクトの XML 定義として識別する必要があります。

次のステップを使用して、カスタム・ビジネス・イベントを作成します。

1. ビジネス・イベントの特定およびダウンロード、ページ 11-15
2. XML 定義ファイルへの注釈付け、ページ 11-18
3. 統合リポジトリ・パーサーを使用した注釈付きソース・ファイルの検証、ページ 11-20
4. データベースへの注釈付きファイルのアップロード、ページ 11-21
5. iLDT ファイルの統合リポジトリへのアップロード、ページ 11-22

ステップ 1: ビジネス・イベントの特定およびダウンロード

Oracle Workflow ビジネス・イベント・システムでカスタム・ビジネス・イベントを作成したら、まずそれらを見つけて、ワークフローXML ローダーを使用してダウンロードします。

ビジネス・イベントの特定

Home Developer Studio **Business Events** Status Monitor Notifications Administration

Events Subscriptions Agents Systems

Business Events: Events >

Events

A business event is an occurrence in an internet or intranet application or program that might be significant to other objects in a system or to external agents. An event group is a type of event composed of a set of individual member events. Event groups let you associate any events you want with each other and reference them as a group in event subscriptions.

Search

Enter search criteria and select the "Go" button to find your event definitions.

Name Go

(Example: Entering "abc" returns "abode" and "efgabc")

Show More Search Options

Results: Events

Create Event Create Event Group

Select Event(s) and ... Delete

Select All | Select None

Select	Name	Display Name	Type	Status	Subscription	Update	Test
☐	oracle.apps.ecx.inbound.message.process	Generic Inbound Message Process Event	Event	Enabled			
☐	oracle.apps.ecx.inbound.message.receive	Generic Inbound Message Event	Event	Enabled			
☐	oracle.apps.ecx.inbound.process_at_java	Oracle XML Gateway processing inbound messages at middle tier event	Event	Enabled			

データベースとフラット・ファイルの間でビジネス・イベント・システム・オブジェクトの XML 定義をダウンロードするには、次のコマンド構文で `oracle.apps.fnd.wf.WFXload` に対して Java を実行して、ワークフロー XML ロードを実行します。

```
jre oracle.apps.fnd.wf.WFXload -d{e} <user> <password> <connect
string>
<protocol> <language> <xml file> <object> {<key>} {<OWNER_TAG>}
{<owner>}
```

たとえば、単一のイベントまたはイベントのグループをダウンロードできます。

- 次のコマンドを使用して、単一のビジネス・イベント(**wfdemoe.wfx** など)をダウンロードします。ファイル名の最初の 2 文字または 3 文字は製品を示し、最後の文字 **e** はイベントを示します。

```
java oracle.apps.fnd.wf.WFXLoad -d apps_read_only password hostdb:
12345:sid100 thin US wfdemoe.wfx EVENTS abc.apps.wf.bes.demo.event
```

- 次のコマンドを使用して、ワイルドカードでビジネス・イベントのグループをダウンロードします。

```
java oracle.apps.fnd.wf.WFXLoad -d apps_read_only password hostdb:
12345:sid100 thin US wfdemoe.wfx EVENTS abc.apps.wf.bes.%
```

イベントの XML 定義を正常にダウンロードしたら、テキスト・エディタで `.wfx` ファイルを開きます。たとえば、次のような 1 つのイベントを含む `wfdemoe.wfx` ファイルのコンテンツが見つかります。

```

<?xml version = '1.0' encoding = 'UTF-8'?>
...

<oracle.apps.wf.event.all.sync><ExternalElement>
<OraTranslatibility>
<XlatElement Name="WF_EVENTS">
<XladID>
<Key>NAME</Key>
</XladID>
<XlatElement Name="DISPLAY_EVENTS" MaxLen="80" Expansion="50"/>
<XladID>
<Key Type="CONSTANT">DISPLAY_EVENTS</Key>
</XladID>
<XlatElement Name="DESCRIPTION" MaxLen="2000" Expansion="50"/>
<XladID>
<Key Type="CONSTANT">DESCRIPTION</Key>
</XladID>
</XlatElement>
</OraTranslatibility>
</ExternalElement>
<WF_TABLE_DATA>
  <WF_EVENTS>
    <VERSION>1.0</VERSION>
    <GUID>#NEW</GUID>
    <NAME>abc.apps.wf.demo.event</NAME>
    <TYPE>EVENT</TYPE>
    <STATUS>ENABLED</STATUS>
    <GENERATE_FUNCTION/>
    <OWNER_MAME>Oracle Workflow</OWNER_MAME>
    <OWNER_TAG>FMD</OWNER_TAG>
    <CUSTOMIZATION_LEVEL>U</CUSTOMIZATION_LEVEL>
    <LICENSED_FLAG>Y</LICENSED_FLAG>
    <JAVA_GENERATE_FUNC/>
    <DISPLAY_NAME>Demo Business Event</DISPLAY_NAME>
    <DESCRIPTION>Business event created for annotation demo.</DESCRIPTION>
    <IREP_ANNOTATION>/**
* Business event created for annotation demo.
*
* @rep:scope public
* @rep:displayname Demo Business Event
* @rep:product FND
* @rep:category BUSINESS_ENTITY
*/
</IREP_ANNOTATION>
  </WF_EVENTS>
</WF_TABLE_DATA>
</oracle.apps.wf.event.all.sync>

```

ワークフローXMLローダーにより、<IREP_ANNOTATION>と</IREP_ANNOTATION>の間の太字で強調表示されている統合リポジトリ・アノテーションのテンプレートが自動的に作成されます。ここでは、ビジネス・イベント・アノテーションの標準に基づいて、ビジネス・イベントに適切なアノテーションを配置または変更する必要があります。

ビジネス・イベントの XML 定義をダウンロードするには:

1. 「ワークフロー管理者 Web アプリケーション」職責を持つユーザーとして Oracle E-Business Suite にログインします。ナビゲータから「ビジネス・イベント」のリンクを選択し、「イベント」ページを開きます。
2. 「検索」リージョンに検索基準を入力して、ビジネス・イベントを見つけます。
3. アプリケーションが実行されている環境と同じ環境にディレクトリを変更します。

たとえば、アプリケーションが seed100 で実行されている場合は、ビジネス・イベントが存在する seed100 にディレクトリを変更します。

```
/slot/ems3404/appmgr/apps/apps_st/appl  
.. /APPSseed100.env
```

4. 次の構文で oracle.apps.fnd.wf.WFXload を使用して、データベースからイベントをダウンロードします。

```
jre oracle.apps.fnd.wf.WFXload -d{e} <user> <password> <connect  
string> <protocol> <language> <xml file> <object> {<key>}  
{<OWNER_TAG>} {<owner>}
```

5. テキスト・エディタで .wfx ファイルを開き、1 つのビジネス・イベントがそこに配置されていることを確認します。

ステップ 2: XML 定義への注釈付け

データベースから XML 定義ファイルを正常にダウンロードしたら、1 つのビジネス・イベントを含む .wfx ファイルをテキスト・エディタで開き、統合リポジトリ・ビジネス・イベント・アノテーションの標準に基づいて、アノテーションを適切に変更する必要があります。

適切なアノテーションには次のものが含まれます。

- わかりやすい説明を入力します。
- ビジネス・イベントが発生する条件を入力します。
- ビジネス・イベントを起動する UI アクションを入力します(該当する場合)。
- 範囲を確認します。デフォルトでは、ワークフロー XML ロードーは範囲に public と注釈付けします。
- 表示名を確認します。デフォルトでは、ワークフロー XML ロードーは、ビジネス・イベント定義で指定されている表示名と同じ表示名を使用します。
- 製品を確認します。デフォルトでは、ワークフロー XML ロードーは、アプリケーション短縮名として所有者タグを使用します。

所有者タグが FND_APPLICATION のアプリケーション短縮名に対応していることを確認してください。所有者名は通常アプリケーション名に対応しますが、製品がより大きなアプリケーションの一部である場合は、所有者名に適切な名前を入力できます。

- 各ビジネス・イベントが属する BUSINESS_ENTITY コードを入力します。
- 必要に応じて、追加のアノテーション・プロパティを入力します。

IREP プロパティは空白にしないでください。たとえば、ワークフロー XML ロードーでは、ビジネス・エンティティのテンプレートは rep:category BUSINESS_ENTITY (ページ A-132)としてのみ追加されますが、イベントが属する適切なビジネス・エンティティを追加する必要があります。

ります。同様に、他の@rep プロパティも空白にすることはできません。次に、Oracle Workflow のビジネス・イベント・アノテーションのサンプルを示します。

```
* Business Event created to demonstrate using WFXLoad to annotate
Business Events.
*
* @rep:scope internal
* @rep:displayname Demo Business Event
* @rep:product OWF
* @rep:lifecycle active
* @rep:category BUSINESS_ENTITY WF_EVENT
*/
```

重要: Oracle Integration Repository でイベントへの注釈付けまたはイベントの公開を行わない場合は、アノテーションのみを削除し、次のタグは変更しないでください。これらのタグの存在は、アノテーションについてイベントが検証されたことを示しています。

<IREP_ANNOTATION/>

または

<IREP_ANNOTATION></IREP_ANNOTATION>

ローダーがこれらの空のタグを検出すると、アノテーションについてビジネス・イベントが検証済みで、統合リポジトリに公開する必要はないと解釈します。次回、ユーザーがこれらのイベントをダウンロードすると、ローダーは次の例に示すように、空の IREP_ANNOTATION タグを挿入します。

ただし、ビジネス・イベントの IREP_ANNOTATION タグ全体を削除してアップロードすると、その後のダウンロード時に、ローダーはビジネス・イベントの部分的に入力されたアノテーション・テンプレートを挿入します。

```

<WF_TABLE_DATA>
  <WF_EVENTS>
    <VERSION>1.0</VERSION>
    <GUID>#NEW</GUID>
    <NAME>oracle.apps.wf.demo.event.noannotate</NAME>
    <TYPE>EVENT</TYPE>
    <STATUS>ENABLED</STATUS>
    <GENERATE_FUNCTION/>
    <OWNER_MAME>Oracle Workflow</OWNER_MAME>
    <OWNER_TAG>FMD</OWNER_TAG>
    <CUSTOMIZATION_LEVEL>U</CUSTOMIZATION_LEVEL>
    <LICENSED_FLAG>Y</LICENSED_FLAG>
    <JAVA_GENERATE_FUNC/>
    <DISPLAY_NAME>Demo Business Event with no
annotation</DISPLAY_NAME>
    <DESCRIPTION>Business second event created for
annotation demo.</DESCRIPTION><IREP_ANNOTATION>/*#
* Business event created for annotation demo.
*
* @rep:scope public
* @rep:displayname Demo Business Event
* @rep:product FND
* @rep:category BUSINESS_ENTITY
*/
</IREP_ANNOTATION>
  </WF_EVENTS>
</WF_TABLE_DATA>

```

統合リポジトリ・ビジネス・イベント・アノテーションの標準の詳細は、「ビジネス・イベント・アノテーション」(ページ A-35)を参照してください。

ステップ 3: 統合リポジトリ・パーサーを使用した注釈付きソース・ファイルの検証

統合リポジトリ・パーサーは、スタンドアロンの設計時ツールです。これを実行すると、注釈付きカスタム・インタフェース定義をアノテーションの標準に対して検証し、エラーが発生しなければ iLDT ファイルを生成できます。

ビジネス・イベントの XML 定義に注釈付けした後、次のコマンド構文を使用してスタンドアロンの統合リポジトリ・パーサー (IREP パーサー) を実行し、.wfx ファイルのアノテーションが有効かどうかを検証します。

コマンド構文:

```

$IAS_ORACLE_HOME/perl/bin/perl $FND_TOP/bin/irep_parser.pl -g -v
-username=<a fnd username> <product>:<relative path from product
top>:<fileName>:<version>=<Complete File Path, if not in current
directory>

```

例:

```

$IAS_ORACLE_HOME/perl/bin/perl $FND_TOP/bin/irep_parser.pl -g -v
-username=sysadmin owf:patch/115/xml/US:wfdemoe.wfx:12.0=.
/wfdemoe.wfx

```

パーサーの実行中は、コンソール上のエラー・メッセージに注意してください。通常、これらのエラーは、注釈付きファイルの不適切なアノテーションまたは構文エラーが原因です。アノテーションが適切で、ファイルの構文が正しいことを確認してください。

注釈付きインタフェース・ファイルでエラーが発生しなかった場合は、iLDT (*.ildt)ファイル

が生成されます。統合管理者は、生成された iLDT ファイルを統合リポジトリ(カスタム・ビジネス・イベントをすべてのユーザーに公開できる)にアップロードする必要があります。「ステップ 5: iLDT ファイルの統合リポジトリへのアップロード」(ページ 11-22)を参照してください。

統合リポジトリ・パーサー(irep_parser.pl)

irep_parser は、設計時ツールです。これは、プログラム・ソース・ファイル内のインタフェース・アノテーション・ドキュメントを読み取り、そのファイル・タイプに従って検証します。-generate フラグが指定されている(および他の条件が満たされる)と、iLDT ファイルが生成されます。検証エラーは通常、grep -n の結果のように、ファイル名および行番号とともに報告されます。

また、ほぼすべての種類のアプリケーション・ソース・ファイルを処理できます。注釈付きファイルを、サポートされているインタフェース・タイプのアノテーションの標準に対して検証しているときは、一致しないファイルは無視されます。

処理中にエラーが発生しなかった場合、パーサーは終了値 0 を返します。エラーが発生した場合は、エラーがあったファイルの数を返します。生成(クラス解決)時に不完全な情報を含むファイルは、-generate フラグが使用されている場合にのみエラーとみなされます。

ただし、統合リポジトリ・パーサーを実行する前に、perl モジュールをインストールして、必要なパッチを適用する必要があります。設定情報の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「統合リポジトリ・パーサーの設定および使用」を参照してください。

統合リポジトリ・パーサー(irep_parser.pl)の使用詳細(サポートされているファイル・タイプおよびオプション、ファイル仕様、環境など)は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「統合リポジトリ・パーサー(irep_parser.pl)の使用詳細」を参照してください。

ステップ 4: 注釈付きファイルのデータベースへのアップロード

注釈付きソース・ファイル .wfx の検証後、このファイルを以前にダウンロードしたデータベースにアップロードして、将来の参照のために、この注釈付きファイルをビジネス・イベント・システムの適切な表に保存できるようにします。

注意: 統合リポジトリ・ブラウザ・ウィンドウでカスタム・ビジネス・イベントを表示するには、統合管理者が、生成された iLDT ファイルを統合リポジトリにアップロードする必要があります。iLDT ファイルのアップロードの詳細は、「ステップ 5: iLDT ファイルの統合リポジトリへのアップロード」(ページ 11-22)を参照してください。

ワークフローXML ローダーを使用すると、標準アップロード・モード(-u)または強制アップロード・モード(-uf)でビジネス・イベント・システムの XML 定義をアップロードできます。

- 標準アップロード・モード(-u): カスタマイズ・レベルが「コア」または「制限」のイベントを作成した場合、ワークフローXML ローダーは、IREP_ANNOTATION をデータベースのビジネス・イベント・システム WF_EVENTS 表に更新できます。この標準モードでは、カスタマイズ・レベルが「ユーザー」のイベントまたはサブスクリプションは更新されません。

次のコマンドを使用して、注釈付き.wfx ファイルをデータベースにアップロードします。

```
java oracle.apps.fnd.wf.WFXLoad -u apps_read_only password  
hostdb:12345:sid100 thin US wfdemoe.wfx
```

- 強制アップロード・モード(-uf): ワークフローXML ロードは、オブジェクト定義をソース XML ファイルからデータベースのビジネス・イベント・システム表にロードし、カスタマイズ・レベルが「ユーザー」のイベントまたはサブスクリプションであっても、既存の定義を上書きします。

そのため、カスタマイズ・レベルが「ユーザー」のイベントを作成した場合は、次の強制アップロード・オプションを使用して、IREP_ANNOTATION をデータベースにアップロードできるようにします。

```
java oracle.apps.fnd.wf.WFXLoad -uf apps_read_only password  
hostdb:12345:sid100 thin US wfdemoe.wfx
```

ワークフローXML ロードの使用の詳細は、*Oracle Workflow 管理者ガイド*の「ワークフローXML ロードの使用」を参照してください。

ステップ 5: ILDT ファイルの統合リポジトリへのアップロード

統合リポジトリ・パーサーを使用して検証した後、iLDT の生成中にエラーが発生しなかった場合は、iLDT ファイルが生成されます。ユーザーが統合リポジトリからカスタム・ビジネス・イベントを表示するには、統合管理者が FNDLOAD コマンドを使用して、生成された iLDT ファイルを統合リポジトリに手動でアップロードする必要があります。

```
$FND_TOP/bin/FNDLOAD <APPS username> 0 Y UPLOAD  
$fnd/patch/115/import/wfirep.lct <ildt file>  
ORACLE Password:
```

たとえば、次のようにします。

```
FND_TOP/bin/FNDLOAD apps @instance_name 0 Y UPLOAD  
$FND_TOP/patch/115/import/wfirep.lct SOAIS_pls.ildt  
ORACLE Password: password
```

iLDT ファイルのアップロード方法の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「iLDT ファイルの統合リポジトリへのアップロード」を参照してください。

カスタム・インタフェースの表示および管理タスクの実行

カスタム・インタフェースの検索および表示

注釈付きカスタム・インタフェース定義は、正常にアップロードされると、それらが属しているインタフェース・タイプにマージされ、統合リポジトリ・ブラウザ・ウィンドウの Oracle シード・インタフェースとともに表示されます。カスタム・インタフェース定義を Oracle インタフェースと簡単に区別するには、インタフェース・ソース「カスタム」を使用して、これらのカスタム統合インタフェースを分類します(Oracle E-Business Suite の Oracle シード・インタフェースの場合は、

インタフェース・ソース「Oracle」)。

カスタム統合インタフェースを検索するには、次の方法のいずれかを使用できます。

- 「インタフェース・リスト」ページで、「インタフェース・ソース」ドロップダウン・リストから「カスタム」を選択し、「範囲」フィールドに値を指定して、カスタム統合インタフェースの表示を絞り込みます。
- 「検索」ページで、「**詳細検索オプションの表示**」をクリックして、「インタフェース・ソース」ドロップダウン・リストから「カスタム」を選択し、必要に応じて検索基準としてインタフェース・タイプ(「ビジネス・イベント」など)、製品ファミリまたは範囲を選択します。

検索を実行すると、一致するカスタム統合インタフェースがすべて表示されます。カスタム統合インタフェースの検索および表示方法の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「カスタム統合インタフェースの検索」、および *Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「カスタム統合インタフェースの表示」を参照してください。

管理タスクの実行

カスタム・ビジネス・イベントが正常にアップロードされ、統合リポジトリ・ユーザー・インタフェースから表示されるようになると、統合管理者はこれらのカスタム・イベントで、ネイティブ・イベントの場合と同じ管理タスクを実行できます。これらの管理タスクには、新しく作成されたカスタム・イベントのセキュリティ付与の作成(必要な場合)、およびカスタム・ビジネス・イベントの購読が含まれます。*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「カスタム統合インタフェースおよびサービスの管理」を参照してください。

Web サービスとしてのカスタム統合インタフェースの使用

概要

適切な注釈付けおよび検証を行うと、Oracle Integration Repository がサポートするインタフェース・タイプのカスタム統合インタフェースを作成できます。カスタム・インタフェースが属するインタフェース・タイプでサービスを有効にできる場合は、カスタム・インタフェースを Web サービスとして使用して、Oracle E-Business Suite からデータを更新または取得したり、Web を介して他のビジネス・トランザクションを実行することができます。

たとえば、統合開発者は PL/SQL API を使用して、仕入先出荷およびデビット要求ビジネス・エンティティの新しいインタフェースまたはカスタマイズ・インタフェースを作成できます。インタフェースが Oracle Integration Repository にアップロードされると、統合リポジトリ・ブラウザの PL/SQL API インタフェース・タイプの下に表示されます。カスタム・インタフェースと Oracle ネイティブ・パッケージ・インタフェースをリポジトリから表示する場合に見分けるには、すべてのカスタム統合インタフェースのインタフェース・ソースは「カスタム」で、Oracle シード・インタフェースのインタフェース・ソースは「Oracle」です。

ビジネス・ニーズを満たすために配置済のカスタム・インタフェースを Web サービスとして使用する方法についてさらに理解を深めるために、この項では、BPEL プロセスによる SOA コ

ンボジット・アプリケーションを作成および配置する際の設計時タスクおよび実行時タスクの詳細について説明します。次の項で説明する例では、Oracle JDeveloper 11g (11.1.1.6.0)を設計時ツールとして使用して BPEL プロセスによる SOA コンボジット・アプリケーションを作成し、Oracle SOA Suite 11g (11.1.1.6.0)を使用してこのプロセスを配置します。

注意: Oracle JDeveloper とその他の Oracle Fusion Middleware コンポーネント (Oracle SOA Suite など)を使用している場合、SOA テクノロジを有効にするには、SOA テクノロジ用の JDeveloper の拡張機能である Oracle SOA Suite コンボジット・エディタを手動でダウンロードする必要があります。追加の Oracle Fusion Middleware 設計時コンポーネントのインストールの詳細は、*Oracle Fusion Middleware Oracle JDeveloper インストレーション・ガイド*を参照してください。

設計時に BPEL プロセスによる SOA コンボジット・アプリケーションを作成する場合のカスタム・インタフェース WSDL の使用

BPEL プロセスによる SOA コンボジット・アプリケーションのシナリオ

この例では、カスタム PL/SQL API ZZ_SDREQUEST を使用して、SOA コンボジット・アプリケーションでの BPEL プロセスの作成について説明します。

仕入先出荷およびデビット要求の作成要求を受け取ると、ペイロードおよび要求番号を含む要求情報が読み取られて渡され、仕入先出荷およびデビット要求が作成されます。製品の仕入先出荷およびデビット要求が作成されると、要求番号が要求者に返されます。

配置後に BPEL プロセスによる SOA コンボジット・アプリケーションが正常に実行されると、Oracle E-Business Suite で仕入先出荷およびデビット要求が作成されます。要求番号は、ペイロード入力値と同じである必要があります。

カスタム Web サービスを使用して BPEL プロセスを作成するための前提条件

Web サービスとして公開されているカスタム・インタフェースに対して設計時タスクを実行する前に、次のタスクを実行しておく必要があります。

- 統合管理者または統合開発者は、事前に Web サービスを生成しておく必要があります。管理者は、生成されたカスタム・サービスを Oracle SOA Suite WebLogic 管理対象サーバーに配置します。
- 統合開発者は、Web サービスとして公開されているカスタム・インタフェースの配置済 WSDL URL を見つけて記録しておく必要があります。

カスタム・インタフェースのセキュリティ付与の作成

このカスタム・インタフェースを確認して使用できるようにするには、管理者はまずリポジトリからカスタム・インタフェース(インタフェース・ソースが「カスタム」)を見つけ、カスタム・インタフェースに対するセキュリティ付与を作成して、適切な権限を持つユーザーがこのインタフェースにアクセスできるようにします。

たとえば、管理者は、「Oracle Trade Management」職責を持つユーザーにカスタム API アクセス権限を付与できます。

このカスタム API の実行後、ユーザーは「Oracle Trade Management」にログオンし、仕入先およびデビット要求の作成の詳細を確認できます。

セキュリティ付与の作成方法の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「付与の作成」を参照してください。

カスタム・インタフェースの Web サービス・コンポジットのデプロイ

統合管理者または統合開発者は、まず選択したカスタム・インタフェースの Web サービスを作成する必要があり、その後管理者が Oracle Integration Repository から Oracle SOA Suite WebLogic 管理対象サーバーにカスタム・サービスをデプロイできます。

たとえば、統合開発者がデプロイ済 WSDL を使用して BPEL を作成する前に、まず次のステップを実行する必要があります。

1. Web サービスを生成するには、統合管理者または統合開発者がまずインタフェース定義(カスタム PL/SQL インタフェース ZZ_SDREQUEST など)を見つけ、相互作用パターン表から目的の相互作用パターン情報を選択します。これは、インタフェースの詳細ページで「生成」をクリックする前に、インタフェース・レベルまたはメソッド・レベルで選択できます。

サービスが正常に生成されると、インタフェースの詳細ページの「Web サービス・ステータス」フィールドが「未生成」から「生成済」に変わります。Web サービスの生成方法の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「Web サービスの生成」を参照してください。

2. 生成した Web サービスを配置するには、統合管理者がいずれかの認証タイプを選択してから、「配置」をクリックします。Oracle SOA Suite に配置されたサービスは、有効なサービスであり、新しい SOAP 要求を受け入れる準備が整っています。

サービスが正常に配置されると、選択した認証タイプが表示され、「Web サービス・ステータス」フィールドが「配置済」および「有効」状態になります。認証タイプによる Web サービスの保護の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「Web サービス・セキュリティの管理」を参照してください。

Web サービスの配置方法の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「Web サービスの配置および配置解除」を参照してください。

WSDL URL の検索および記録

管理者が実行する必要があるタスクとは別に、統合開発者は、Oracle JDeveloper で有用なビジネス・プロセスにオーケストレーションする必要があるカスタム・インタフェースのデプロイ済 Web サービスの WSDL URL を見つけて記録する必要があります。

これを行うには、インタフェースの詳細ページの「WSDL の表示」リンクをクリックします。新しいポップアップ・ウィンドウから WSDL URL をコピーします。この URL は、後で BPEL プロセスでパートナ・リンク・サービスを作成するときに使用されます。

インタフェースを検索してインタフェースの詳細を表示する方法の詳細は、「統合インタフェースの検索および表示」(ページ 2-1)を参照してください。

BPEL プロセスによる SOA コンポジット・アプリケーションの作成フロー

この章では、仕入先およびデビット要求の作成シナリオに基づいて、次の設計時タスクを説明します。

1. 新しい BPEL プロセスによる SOA コンポジット・アプリケーションの作成、ページ 11-26

このステップを使用して、同期 BPEL プロセス・テンプレートで

ZZ_CreateSingle_ShipDebitRequest.bpel という新しい BPEL プロセスによる SOA コンポジット・アプリケーションを作成します。これにより、サード・パーティ・アプリケーションからの入力を受け入れる受入と、要求アプリケーションに BPEL プロセスの出力を返信する返信の 2 つのダミー・アクティビティが自動的に作成されます。

2. パートナ・リンクの作成、ページ 11-28

このステップを使用して、Web サービスとして公開されている単一出荷およびデビット要求カスタム API ZZ_SDREQUEST を使用して、Oracle E-Business Suite での起動を作成します。

3. ファイル・アダプタのパートナ・リンクの追加、ページ 11-28

このステップを使用して、1 つ目の Assign アクティビティから渡される入力データの詳細を同期的に読み取り、仕入先出荷およびデビット要求を作成します。

4. Invoke アクティビティの追加、ページ 11-33

このステップを使用して、次の目的の 2 つの Invoke アクティビティを構成します。

- ファイル・アダプタを指し示して、1 つ目の Assign アクティビティから渡される入力データの詳細を同期的に読み取ります。
- ZZ_SDREQUEST パートナ・リンクを指し示して、Assign アクティビティから受け取るペイロードおよび要求番号の詳細を使用して、仕入先出荷およびデビット要求の作成を開始します。

5. Assign アクティビティの追加、ページ 11-34

このステップを使用して、アプリケーション・コンテキスト・ヘッダー変数、ペイロード情報および要求番号を適切な Invoke アクティビティに渡して、単一出荷およびデビット要求の作成を容易にするように、Assign アクティビティを構成します。最後に、ダミーの返信アクティビティを介して、要求アプリケーションに要求番号を渡します。

BPEL プロセスのサービス・コンポーネントを使用した SOA コンポジット・アプリケーションの一般情報と作成方法の詳細は、*Oracle Fusion Middleware Oracle SOA Suite 開発者ガイド* を参照してください。

新しい BPEL プロセスによる SOA コンポジット・アプリケーションの作成

このステップを使用して、様々な BPEL プロセス・アクティビティによる新規 SOA コンポジット・アプリケーションを作成します。

新しい BPEL プロセスによる SOA コンポジット・アプリケーションを作成するには:

1. Oracle JDeveloper を開きます。
2. アプリケーション・ナビゲータで「**新規アプリケーション**」をクリックします。
「SOA アプリケーションの作成 - アプリケーションの名前付け」ページが表示されます。
3. 「アプリケーション名」フィールドにアプリケーションの適切な名前を入力して、「アプリケーション・テンプレート」リストから「**SOA アプリケーション**」を選択します。
「**次へ**」をクリックします。「SOA アプリケーションの作成 - プロジェクトの名前付け」ページが表示されます。
4. 「プロジェクト名」フィールドにプロジェクトの適切な名前を入力します
(ZZ_CreateSingle_ShipDebitRequest など)。
5. 「プロジェクト・テクノロジー」タブで、「Web サービス」を選択して、「使用可能」のテクノロジー・リストから「**選択済**」のテクノロジー・リストに「**SOA**」が選択されるようにします。
「**次へ**」をクリックします。「SOA アプリケーションの作成 - SOA 設定の構成」ページが表示されます。
6. 「コンポジット・テンプレート」リストから**コンポジット BPEL プロセス**を選択して、「**終了**」をクリックします。新規アプリケーションと SOA プロジェクトを作成しました。これにより、SOA コンポジットが自動的に作成されます。
「BPEL プロセスの作成」ページが表示されます。
7. デフォルトの「**BPEL 1.1 Specification**」の選択を変更せずに維持します。これにより、BPEL 1.1 Specification をサポートする BPEL プロジェクトが作成されます。
「名前」フィールドに BPEL プロセスの適切な名前を入力します
(ZZ_CreateSingle_ShipDebitRequest など)。
「テンプレート」フィールドで「**同期 BPEL プロセス**」を選択します。
「トランザクション」ドロップダウン・リストから「**required**」を選択します。「**OK**」をクリックします。
受入アクティビティと返信アクティビティのある同期 BPEL プロセスが作成されます。指定した名前を使用した、bpel および wsdl を含む必須のソース・ファイル
(ZZ_CreateSingle_ShipDebitRequest.bpel、
ZZ_CreateSingle_ShipDebitRequest.wsdl)と composite.xml も生成されます。
注意: サービス・プロバイダでは、パラメータ・タイプ名に\$の文字がある PL/SQL ストアド・プロシージャまたはパッケージのサービス作成はサポートされません。
名前に\$があると、XSD 生成に失敗します。
8. 「SOA コンテンツ」→「ビジネス・ルール」にナビゲートし、composite.xml をダブルクリックしてコンポジット・ダイアグラムを表示します。
ZZ_CreateSingle_ShipDebitRequest コンポーネントをダブルクリックして、

BPEL プロセスを開きます。

Web サービスのパートナ・リンクの作成

このステップを使用して、ZZ_CreateSD_Request というパートナ・リンクを作成します。

単一出荷およびデビット要求 Web サービスのパートナ・リンクを作成するには:

1. Oracle JDeveloper で、「パートナ・リンク」領域にマウスを置いて右クリックして、プルダウン・メニューから「パートナ・リンクの作成」を選択します。または、「BPEL コンストラクト」リストから「パートナ・リンク」を、プロセス・ダイアグラムの右側の「パートナ・リンク」スイムレーンにドラッグ・アンド・ドロップできます。

「パートナ・リンクの作成」ウィンドウが表示されます。

2. 「WSDL ファイル」フィールドに、統合リポジトリから以前に記録したカスタム・サービス (ZZ_SDREQUEST) に対応する WSDL URL をコピーします。

[Tab] キーを押します。

3. デフォルトでパートナ・リンク・タイプを作成する新しい WSDL ファイルをシステムで作成するかどうかを尋ねる「パートナ・リンク・タイプ」メッセージ・ダイアログ・ボックスが表示されます。

「はい」をクリックします。パートナ・リンク名を手動で入力できます。

4. ドロップダウン・リストから「パートナ・リンク・タイプ」と「パートナ・ロール」の値を選択します。

「適用」をクリックします。

パートナ・リンクは、必要な WSDL 設定で作成され、BPEL プロセスでは、プロセス・ダイアグラムの枠線の領域の新しいアイコンによって表現されます。

5. 「OK」をクリックして、パートナ・リンクの構成を完了します。

BPEL プロセス・ダイアグラムの「パートナ・リンク」セクションに、パートナ・リンク ZZ_SDREQUEST が追加されます。

ファイル・アダプタのパートナ・リンクの追加

このステップを使用して、入力ペイロードを読み取るように BPEL プロセスを構成します。

ファイル・アダプタのパートナ・リンクをペイロードの読取りに追加するには:

1. Oracle JDeveloper で、「BPEL サービス」リストから「ファイル・アダプタ」サービスを、プロセス・ダイアグラムの右側の「パートナ・リンク」スイムレーンにドラッグ・アンド・ドロップします。ファイル・アダプタ構成ウィザードへようこそページが表示されます。
2. 「次へ」をクリックします。「サービス名」ダイアログ・ボックスが表示されます。
3. ファイル・アダプタ・サービス名 (Read_Payload など) を入力します。

4. 「次へ」をクリックします。「アダプタ・インタフェース」ダイアログ・ボックスが表示されます。
5. 「操作およびスキーマから定義(後で指定)」ラジオ・ボタンを選択して、「次へ」をクリックします。「操作」ダイアログ・ボックスが表示されます。

操作の指定

Operation

The File Adapter supports four operations. There is a Read File operation that polls for incoming files in your local file system, a Write File operation that creates outgoing files, a Synchronous Read File operation that reads the current contents of a file, and a List Files operation that lists file names in specified locations. Specify the Operation type and Operation Name. Only one operation per Adapter Service may be defined using this wizard.

Operation Type: ☐ Read File ☐ Write File ☒ Synchronous Read File ☐ List Files

Operation Name:

Help < Back Next > Finish Cancel

6. 「同期 Read File」などの操作タイプを指定します。これにより、「操作名」フィールドが自動的に設定されます。
- 「次へ」をクリックして、「ファイル・ディレクトリ」ダイアログ・ボックスにアクセスします。

入力ファイル・ディレクトリの指定

Adapter Configuration Wizard - Step 5 of 8

File Directories

Enter directory information for the incoming file of the Synchronous Read File operation.

Directory names are specified as: ☒ Physical Path ☐ Logical Name

Directory for Incoming Files (physical path):

☐ Archive processed files
Archive Directory for Processed Files (physical path):

☒ Delete files after successful retrieval

7. 「物理パス」ラジオ・ボタンを選択して、着信ファイル・ディレクトリの物理パスの情報を入力します。たとえば、ディレクトリ名として /usr/tmp/ と入力します。

注意: ここで指定した物理ディレクトリからこのファイルを見つけることができるようにするには、まず入力ペイロード・ファイル(Inputzzsdrequest.xml)を、指定したディレクトリに配置する必要があります。

または、「参照」をクリックして、着信ファイルのディレクトリ情報を見つけます。「次へ」をクリックして、「ファイル名」ダイアログ・ボックスを開きます。

8. 同期 Read File 操作のファイル名を入力します。たとえば、Inputzzsdrequest.xml と入力します。「次へ」をクリックします。「メッセージ」ダイアログ・ボックスが表示されます。

メッセージ・スキーマの指定

Adapter Configuration Wizard - Step 7 of 8

Messages

Define the message for the Synchronous Read File operation. Specify the Schema File Location and select the Schema Element that defines the messages in the incoming files. Use the Browse button to find an existing schema definition. If you check 'Schema is Opaque', then you do not need to specify a Schema.

Message Schema

☐ Native format translation is not required (Schema is Opaque)

URL 

Schema Element

Help < Back Next > Finish Cancel

9. 「URL」フィールドの右にある「スキーマ・ファイルを参照」を選択します。「タイプ・チューザ」ウィンドウが表示されます。

「タイプ・チューザ」ウィンドウの右上隅にある「スキーマ・ファイルのインポート」ボタンをクリックします。

サービスのスキーマの場所を入力します。例: http:

```
//<soa_suite_hostname>:<port>/soa-  
infra/services/default/<jndi_name>_PLSQL_ZZ_SDREQUEST/ZZ_SD  
REQ_UEST_Service?  
XSD=xsd/APPS_XX_BPPEL_ZZ_CREATE_SDREQUEST_RE_ZZ_SDREQUEST_ZZ  
_CR EATE_SDREQU.xsd.
```

サービスのスキーマの場所は、サービスの WSDL URL

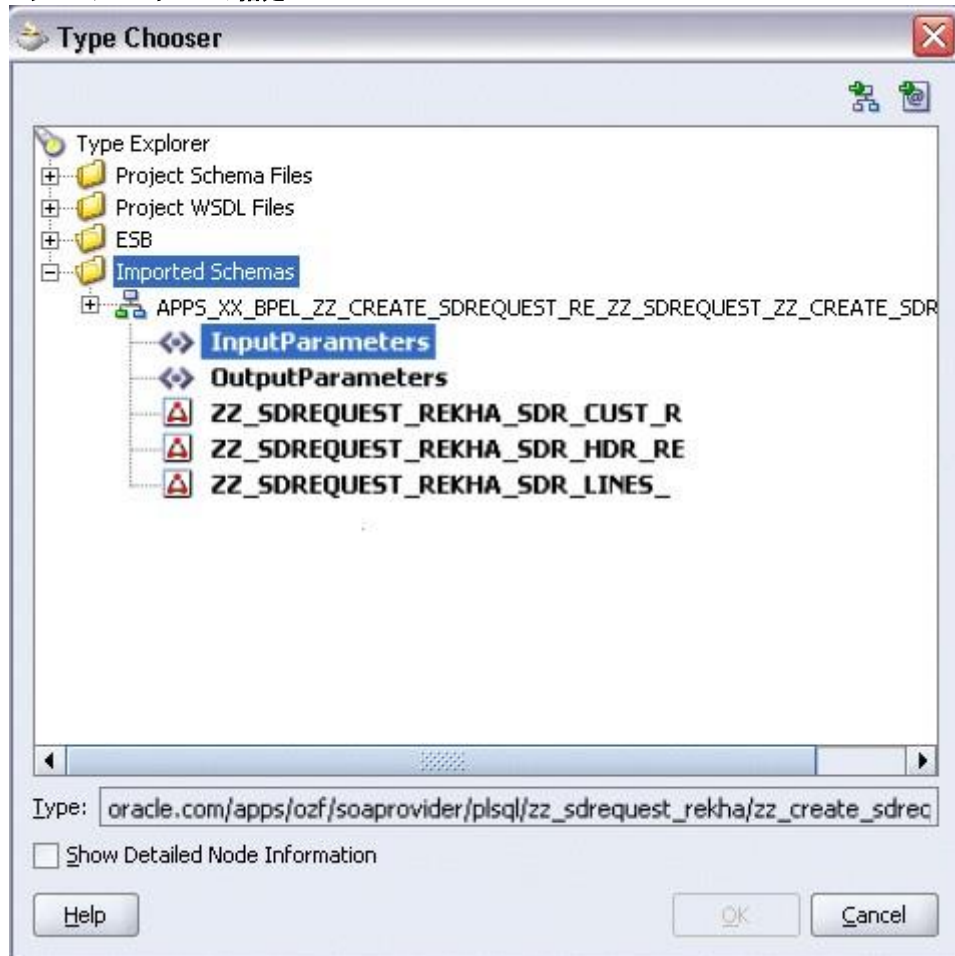
(http://<soa_suite_hostname>:<port>/soa-
infra/services/default/<jndi_name>_PLSQL_ZZ_SDREQUEST/ZZ_SD
REQ_UEST_Service/?wsdl など)から見つけることができます。

「プロジェクトにコピー」チェック・ボックスを選択して、「OK」をクリックします。

「ファイルのローカライズ」ウィンドウが表示されます。「インポートされたファイルの元のディレクトリ構造を保持」チェック・ボックスが選択されていることを確認し、「OK」をクリックします。

「タイプ・チューザ」ウィンドウに、「インポートしたスキーマ」フォルダが自動的に追加されます。

メッセージ・スキーマの指定



「インポートしたスキーマ」フォルダを展開し、
APPS_XX_BP_EL_ZZ_CREATE_SDREQUEST_RE_ZZ_SDREQUEST_ZZ_CREATE_
SDREQU.xsd で InputParameters メッセージを選択します。「OK」をクリックします。

選択した.xsd が URL として表示されて、InputParameters がスキーマ要素として選択されます。

10. 「次へ」→「終了」をクリックします。ウィザードによって、パートナー・リンクに対応する WSDL ファイルが生成されます。メインの「パートナー・リンクの作成」ダイアログ・ボックスが表示されるので、新しい WSDL ファイル Read_Payload.wsdl を指定します。

「適用」、「OK」の順にクリックして構成を完了し、ファイル・アダプタ・サービスに必要な WSDL 設定を指定してパートナー・リンクを作成します。

BP_EL プロセス・ダイアグラムに Read_Payload パートナ・リンクが表示されます。

11. アプリケーション・ウィンドウで、ファイル ReadPayload_file.jca に移動します。DeleteFile プロパティの値を false に設定します。

Invoke アクティビティの追加

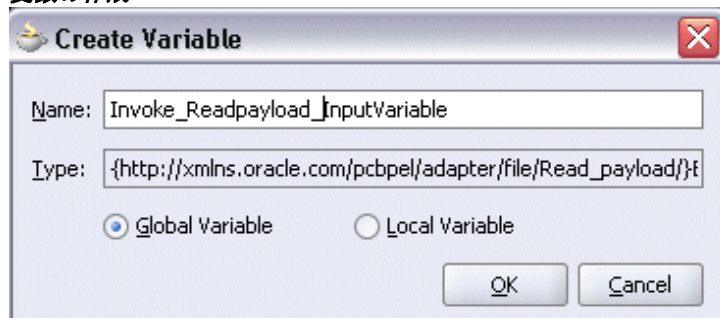
このステップでは、次の 2 つの **Invoke** アクティビティを構成します。

- ファイル・アダプタの Read_Payload パートナ・リンクを使用して、1 つ目の **Assign** アクティビティから渡される仕入先出荷およびデビット要求作成の詳細を読み取ります。
- **Assign** アクティビティから受け取るペイロードおよび要求番号の詳細を送信し、ZZ_SDREQUEST パートナ・リンクを使用して、単一仕入先出荷およびデビット要求を作成します。

Read_Payload パートナ・リンクの **Invoke** アクティビティを追加するには:

1. Oracle JDeveloper で、コンポーネント・パレットから「**BPEL コンストラクト**」を展開します。**Invoke** アクティビティを、プロセス・ダイアグラムの中央のスイムレーンの、**receiveInput** と **replyOutput** のアクティビティの間にドラッグ・アンド・ドロップします。
2. **Invoke** アクティビティを Read_Payload サービスにリンクします。**Invoke** アクティビティでは、要求データをパートナ・リンクに送信します。「編集 Invoke」ダイアログ・ボックスが表示されます。
3. **Invoke** アクティビティの名前(Invoke_Readpayload など)を入力して、「入力変数」フィールドの横にある「**入力変数の作成**」アイコンをクリックして、新しい変数を作成します。「変数の作成」ダイアログ・ボックスが表示されます。

変数の作成



4. 変数の名前(Invoke_Readpayload_InputVariable など)を入力して、「**グローバル変数**」ラジオ・ボタンを選択します。「変数の作成」ダイアログ・ボックスで「**OK**」をクリックします。

「出力変数」フィールドの横にある「**入力変数の作成**」アイコンをクリックして、新しい変数を作成します。「変数の作成」ダイアログ・ボックスが表示されます。

出力変数の名前(Invoke_Readpayload_OutputVariable など)を入力して、「**グローバル変数**」ラジオ・ボタンを選択します。「変数の作成」ダイアログ・ボックスで「**OK**」をクリックします。

「編集 Invoke」ダイアログ・ボックスで「適用」、「OK」の順にクリックして、Invoke アクティビティの構成を完了します。

プロセス・ダイアグラムに Invoke アクティビティが表示されます。

ZZ_SDREQUEST パートナ・リンクの Invoke アクティビティを追加するには:

1. Oracle JDeveloper で、コンポーネント・パレットから「BPEL コンストラクト」を展開します。Invoke アクティビティを、プロセス・ダイアグラムの中央のスイムレーンの、最初の Invoke アクティビティと **replyOutput** のアクティビティの後にドラッグ・アンド・ドロップします。
2. Invoke アクティビティを ZZ_SDREQUEST サービスにリンクします。Invoke アクティビティでは、要求番号をパートナ・リンクに送信します。「編集 Invoke」ダイアログ・ボックスが表示されます。
3. Invoke アクティビティの名前(Invoke_zzsdrequest など)を入力します。「操作」を「ZZ_CREATE_SDREQUEST」として選択します。
4. 「入力変数」フィールドの横にある「作成」アイコンをクリックして、新しい変数 (Invoke_zzsdrequest_InputVariable など)を作成します。「変数の作成」ダイアログ・ボックスで「グローバル変数」ラジオ・ボタンを選択し、「OK」をクリックします。
5. 「出力変数」フィールドの横にある「作成」アイコンをクリックして、新しい変数 (Invoke_zzsdrequest_OutVariable など)を作成します。「変数の作成」ダイアログ・ボックスで「グローバル変数」ラジオ・ボタンを選択し、「OK」をクリックします。「編集 Invoke」ダイアログ・ボックスで「適用」、「OK」の順にクリックして、Invoke アクティビティの作成を完了します。

プロセス・ダイアグラムに Invoke アクティビティが表示されます。

Assign アクティビティの追加

このステップでは、次の目的の 4 つの Assign アクティビティを構成します。

1. ダミーの受入アクティビティから取得したアプリケーション・コンテキスト情報を設定します。この情報は、SOAP 要求の SOAHeader 要素の変数を渡す際に使用されます。
注意: サービスの実行中に、アプリケーション・コンテキストを設定するために要求される値を SOAHeader 要素で渡すには、BPEL プロセスの特定の変数を設定する必要があります。これらの SOAHeader の要素は、Responsibility、RespApplication、SecurityGroup、NLSLanguage および Org_Id です。
2. ペイロード情報を Invoke_zzsdrequestInvoke アクティビティに渡します。
3. 仕入先出荷およびデビット要求番号情報を Invoke_zzsdrequestInvoke アクティビティに渡します。

- 仕入先出荷およびデビット要求番号情報を出力としてダミーの**返信**アクティビティに渡します。

アプリケーション・コンテキストの詳細を **Invoke_ReadpayloadInvoke** アクティビティに渡す 1 つ目の **Assign** アクティビティを追加するには:

- Oracle JDeveloper で、コンポーネント・パレットから「**BPEL コンストラクト**」を展開します。**Assign** アクティビティを、プロセス・ダイアグラムの中央のスイムレーンの、**ReceiveInput** アクティビティと 1 つ目の **Invoke** アクティビティの間にドラッグ・アンド・ドロップします。
- Assign** アクティビティをダブルクリックして、「編集 Assign」ダイアログ・ボックスにアクセスします。
- 「一般」タブをクリックして、**Assign** アクティビティの名前(SetHeader など)を入力します。
- 「コピー・ルール」タブを選択し、ターゲット・ツリーを展開します。
 - 「式」アイコンをクリックして、「式ビルダー」ダイアログを起動します。

「式」ボックスに 204 と入力します。「OK」をクリックします。「割当ての編集」ダイアログの中央の、「元」と「先」のナビゲーション・ツリー・ノードの間に、式の値(204)を持つ「式」アイコンが表示されます。
 - 「先」ナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「Invoke_zzsdrequest_InputVariable」→「header」→「ns5:SOAHeader」にナビゲートして、「ns5: ORG_ID」を選択します。

「式」アイコンをドラッグして、識別したばかりのターゲット・ノード(ns5:ORG_ID)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。
- 「式」アイコンをクリックして「式ビルダー」ダイアログを起動することによって、2 つ目のパラメータ・ペアを入力します。
 - 「式」ボックスに TRADE_MANAGEMENT_USER と入力します。「OK」をクリックします。「割当ての編集」ダイアログの中央(ソースとターゲットの各ナビゲーション・ツリー・ノードの間)に、「式」アイコンと式の値が表示されます。
 - 「先」ナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「Invoke_zzsdrequest_InputVariable」→「header」→「ns5:SOAHeader」にナビゲートして、「ns5: Responsibility」を選択します。

「式」アイコンをドラッグして、識別したばかりのターゲット・ノード(ns5:Responsibility)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。
- 「式」アイコンをクリックして「式ビルダー」ダイアログを起動して、3 つ目のパラメータのペアを入力します。

- 「式」ボックスに **OZF** と入力します。「**OK**」をクリックします。「割当ての編集」ダイアログの中央の、「元」と「先」のナビゲーション・ツリー・ノードの間に、式の値(**OZF**)を持つ「式」アイコンが表示されます。
- 「先」ナビゲーション・ツリーで、「**変数**」→「**プロセス**」→「**変数**」→「**Invoke_zzsdrequest_InputVariable**」→「**header**」→「**ns5:SOAHeader**」にナビゲートして、「**ns5: RespApplication**」を選択します。

「式」アイコンをドラッグして、識別したばかりのターゲット・ノード(**ns5:RespApplication**)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

7. 「式」アイコンをクリックして「式ビルダー」ダイアログを起動して、4 つ目のパラメータのペアを入力します。

- 「式」ボックスに **STANDARD** と入力します。「**OK**」をクリックします。「割当ての編集」ダイアログの中央の、「元」と「先」のナビゲーション・ツリー・ノードの間に、式の値(**STANDARD**)を持つ「式」アイコンが表示されます。
- 「先」ナビゲーション・ツリーで、「**変数**」→「**プロセス**」→「**変数**」→「**Invoke_zzsdrequest_InputVariable**」→「**header**」→「**ns5:SOAHeader**」にナビゲートして、「**ns5: SecurityGroup**」を選択します。

「式」アイコンをドラッグして、識別したばかりのターゲット・ノード(**ns5:SecurityGroup**)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

8. 「式」アイコンをクリックして「式ビルダー」ダイアログを起動して、5 つ目のパラメータのペアを入力します。

- 「式」ボックスに **AMERICAN** と入力します。「**OK**」をクリックします。「割当ての編集」ダイアログの中央の、「元」と「先」のナビゲーション・ツリー・ノードの間に、式の値(**AMERICAN**)を持つ「式」アイコンが表示されます。
- 「先」ナビゲーション・ツリーで、「**変数**」→「**プロセス**」→「**変数**」→「**Invoke_zzsdrequest_InputVariable**」→「**header**」→「**ns5:SOAHeader**」にナビゲートして、「**ns5: NLSLanguage**」を選択します。

「式」アイコンをドラッグして、識別したばかりのターゲット・ノード(**ns5:NLSLanguage**)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 Assign」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

9. 「**OK**」をクリックして、**Assign** アクティビティの構成を完了します。

ペイロード情報を **Invoke_zzsdrequestInvoke** アクティビティに渡す 2 つ目の **Assign** アクティビティを入力するには:

1. 2つ目の **Assign** アクティビティを追加するには、**Assign** アクティビティを「**BPEL コンストラクト**」からプロセス・ダイアグラムの 2 つの **Invoke** アクティビティの間にドラッグ・アンド・ドロップします。
2. 1つ目の **Assign** アクティビティの作成で説明したステップ 2 とステップ 3 を繰り返して、**SetPayload** という 2 つ目の **Assign** アクティビティを追加します。
3. 「コピー・ルール」タブを選択して、ソース・ツリーとターゲット・ツリーを展開します。
 - 「元」ナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「**Invoke_ReadPayload_OutVariable**」→「**ZZ_CreateSingle_ShipDebitRequestProcessRequest**」にナビゲートします。
 - 「先」ナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「**Invoke_zzsdrequest_InputVariable**」→「**body**」にナビゲートします。

ソース・ノード(**ZZ_CreateSingle_ShipDebitRequestProcessRequest**)をドラッグして、識別したばかりのターゲット・ノード(**body**)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 **Assign**」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

4. 「**OK**」をクリックして、**Assign** アクティビティの構成を完了します。

仕入先出荷およびデビット要求番号を **Invoke_zzsdrequestInvoke** アクティビティに渡す 3 つ目の **Assign** アクティビティを入力するには:

1. 3つ目の **Assign** アクティビティを追加するには、**Assign** アクティビティを、「**BPEL コンストラクト**」からプロセス・ダイアグラムの 2 つ目の **Assign** アクティビティと **Invoke_zzsdrequest Invoke** アクティビティの間にドラッグ・アンド・ドロップします。
2. 1つ目の **Assign** アクティビティの作成で説明したステップ 2 とステップ 3 を繰り返して、**SetRequestNumber** という 3 つ目の **Assign** アクティビティを追加します。
3. 「コピー・ルール」タブを選択して、ソース・ツリーとターゲット・ツリーを展開します。
 - 「元」ナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「**inputVariable**」→「**Payload**」→「**client: ZZ_CreateSingle_ShipDebitRequestProcessRequest**」→「**client:input**」にナビゲートします。
 - 「先」ナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「**Invoke_zzsdrequest_InputVariable**」→「**Body**」→「**ns3:InputParameters**」→「**ns3: CP_SDR_HDR_REC**」→「**ns3:REQUEST_NUMBER**」にナビゲートして、「**ns3: TRX_NUMBER**」を選択します。

ソース・ノード(**client:input**)をドラッグして、識別したばかりのターゲット・ノード(**ns3: TRX_NUMBER**)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 **Assign**」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

4. 「割当て」ウィンドウで「OK」をクリックして、**Assign** アクティビティの構成を完了します。

仕入先出荷およびデビット要求番号を返信する 4 つ目の **Assign** アクティビティを追加するには:

1. 3 つ目の **Assign** アクティビティを追加するには、**Assign** アクティビティを、「**BPEL コンストラクト**」からプロセス・ダイアグラムの **Invoke_zzsdrequest****Invoke** アクティビティと **ReplyOutput** アクティビティの間にドラッグ・アンド・ドロップします。
2. 1 つ目の **Assign** アクティビティの作成で説明したステップ 2 とステップ 3 を繰り返して、**SetRequestNumber** という 4 つ目の **Assign** アクティビティを追加します。
3. 「コピー・ルール」タブを選択して、ソース・ツリーとターゲット・ツリーを展開します。
 - 「元」ナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「**Invoke_zzsdrequest_OutputVariable**」→「**body**」にナビゲートします。
 - 「先」ナビゲーション・ツリーで、「変数」→「プロセス」→「変数」→「**outputVariable**」→「**payload**」にナビゲートします。

ソース・ノード(本文)をドラッグして、識別したばかりのターゲット・ノード(ペイロード)に接続します。これにより、ソース・ノードとターゲット・ノードを接続する線が作成されます。「編集 **Assign**」ダイアログ・ボックスの下部にある「元」と「先」の各セクションに、コピー・ルールが表示されます。

4. 「割当て」ウィンドウで「OK」をクリックして、**Assign** アクティビティの構成を完了します。

Web サービス・ポリシーの構成

次のステップを使用して、設計時にセキュリティ・ポリシーを追加します。

1. 「SOA コンテンツ」→「ビジネス・ルール」→**composite.xml** に移動します。
ZZ_SDREQUEST サービスを右クリックして、ドロップダウン・リストから「**WS ポリシーの構成**」を選択します。
2. 「SOA WS ポリシーの構成」ダイアログが表示されます。
「セキュリティ」セクションで「**セキュリティ・ポリシーの追加**」アイコン(+)をクリックします。
サーバー・セキュリティ・ポリシーの選択ダイアログが表示されます。
「**oracle/wss_username_token_service_policy**」を選択して、「OK」をクリックします。添付されたセキュリティ・ポリシーが「セキュリティ」セクションに表示されます。
3. ナビゲーション・メニューから「表示」→「**プロパティ・インスペクタ**」を選択して、
ZZ_SDREQUEST サービス・コンポーネントの「**プロパティ・インスペクタ**」ウィンドウを表示します。
「プロパティ」セクションで、バインディング・プロパティの「**セキュリティ・ポリシーの追加**」アイコン(+)をクリックします。「**プロパティの作成**」ダイアログが表示されます。

「名前」フィールドに `oracle.webservices.auth.username` と入力して、値として `operations` と入力します。

「OK」をクリックします。

4. 「プロパティ」セクションのバインディング・プロパティで「**セキュリティ・ポリシーの追加**」アイコン(+)を再度クリックする、同じアプローチを使用します。「名前」フィールドに `oracle.webservices.auth.password` と入力します。「値」フィールドに、`operations` ユーザーに紐づくパスワードを入力します。

「OK」をクリックします。

「プロパティ」セクションに、選択したプロパティの名前と値の両方が表示されます。

実行時における BPEL プロセスによる SOA コンポジットのデプロイおよびテスト

SOA コンポジットに含まれる BPEL クライアントから同期カスタム・サービス (ZZ_SDREQUEST) を起動するには、SOA コンポジットを Oracle WebLogic 管理対象サーバーにデプロイする必要があります。これは、Oracle JDeveloper を使用して達成できます。コンポジットがデプロイされたら、Oracle Enterprise Manager Fusion Middleware Control コンソールからテストできます。

前提条件

Oracle JDeveloper を使用して BPEL プロセスによる SOA コンポジットをデプロイする前に、設計時環境とランタイム・サーバーの間の接続を確立する必要があります。必要なサーバー接続を構成する方法の詳細は、「サーバー接続の構成」(ページ B-1)を参照してください。

注意: WebLogic サーバーのローカル・インスタンスが使用される場合、Oracle JDeveloper から「実行」→「サーバー・インスタンスの起動」を選択して、WebLogic サーバーを起動します。WebLogic 管理サーバーの DefaultServer インスタンスが正常に起動されたら、「実行中:DefaultServer」と「メッセージ」のログに<Server started in Running mode>と DefaultServer started のメッセージが表示されるはずです。

次の実行時タスクを実行します。

1. BPEL プロセスによる SOA コンポジット・アプリケーションのデプロイ、ページ 11-39
2. BPEL プロセスによる SOA コンポジット・アプリケーションのテスト、ページ 11-42

BPEL プロセスによる SOA コンポジットのデプロイ

以前に作成した SOA コンポジット・アプリケーションに含まれる単一仕入先出荷およびデビット要求の作成 BPEL プロセス(ZZ_CreateSingle_ShipDebitRequest.bpel)をデプロイしてから実行する必要があります。

SOA コンポジット・アプリケーションをデプロイするには:

1. JDeveloper のアプリケーション・ナビゲータで、**ZZ_CreateSingle_ShipDebitRequest** プロジェクトを選択します。
2. プロジェクトを右クリックして、メニューから「**配置**」→[project name]→[serverConnection]を選択します。

たとえば、接続が適切に設定されている場合は、「**デプロイ**」→「**ZZ_CreateSingle_ShipDebitRequest**」→「**SOAServer**」を選択してプロセスをデプロイできます。

注意: サーバー接続を初めて設定する場合は、「デプロイメント・アクション」ダイアログが表示されます。「アプリケーション・サーバーにデプロイ」を選択して、「**次へ**」をクリックします。

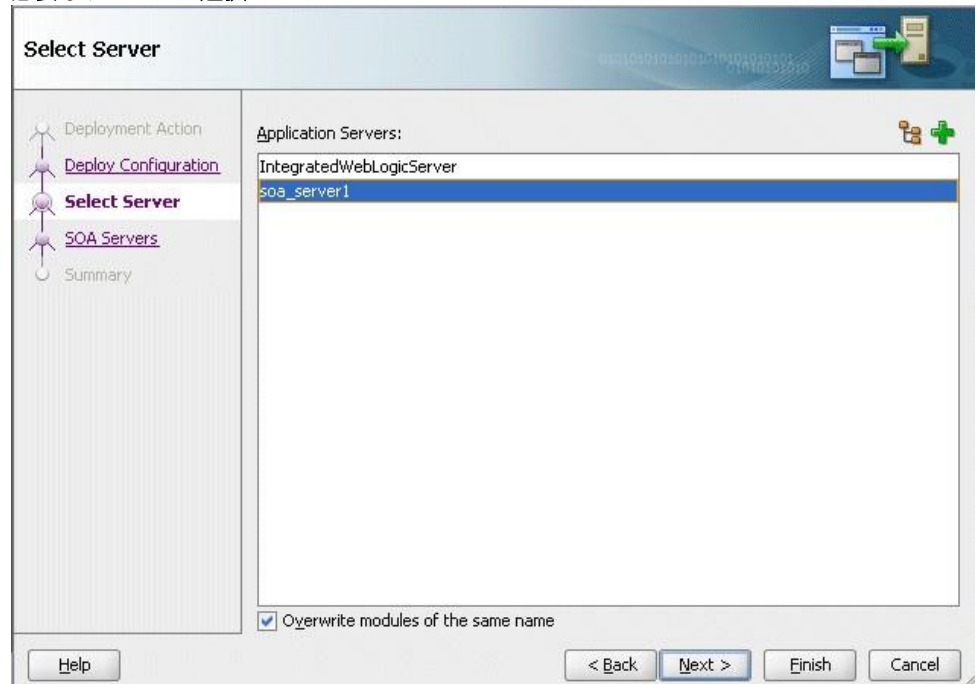
「構成のデプロイ」ダイアログで、「**次へ**」をクリックして新しいアプリケーション・サーバーを追加する前に、次の情報が選択されるようにします。

- 新規リビジョン ID: 1.0
- コンポジット・リビジョンをデフォルトとしてマークします。: このチェック・ボックスを選択します。
- 同じリビジョン ID で既存のコンポジットを上書きします。: このチェック・ボックスを選択します。

JDeveloper から新しい Oracle WebLogic Server 接続を作成するステップは、「サーバー接続の構成」(ページ B- 1)を参照してください。

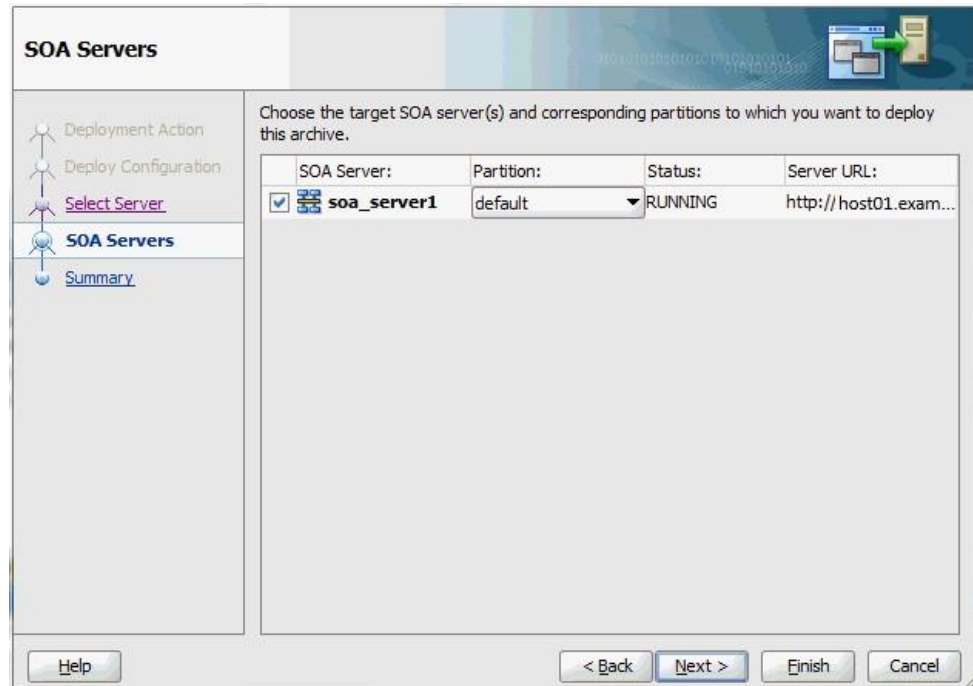
3. 「サーバーの選択」ダイアログで、以前にサーバー接続を確立した soa-server1 を選択します。「**次へ**」をクリックします。

必要なサーバーの選択



4. 「SOA サーバー」ダイアログで、デフォルトのターゲット SOA サーバー(soa-server1)の選択を受け入れます。

必要な SOA サーバーの選択



「次へ」→「終了」をクリックします。

5. Oracle JDeveloper セッションから初めてコンポジットをデプロイする場合は、「認証リクエスト」ウィンドウが表示されます。Oracle SOA Suite のインストール時に指定したユーザー名とパスワード情報を入力します。「OK」をクリックします。
6. デプロイ処理が開始します。SOA - ログ・ウィンドウで、デプロイ・プロセスをモニターして、コンパイルが成功したことを確認します。

デプロイメント - ログ・ウィンドウで、デプロイが成功したことを確認します。

BPEL プロセスによる SOA コンポジット・アプリケーションのテスト

SOA コンポジット・アプリケーションに含まれる BPEL プロセスが正常にデプロイされたら、Oracle Enterprise Manager Fusion Middleware Control コンソールからプロセスを管理およびモニターできます。Oracle SOA Suite の詳細は、*Oracle Fusion Middleware Oracle SOA Suite 管理者ガイド*を参照してください。

また、プロセスを手動で開始してプロセスおよび統合インタフェースをテストし、Oracle E-Business Suite にログオンして、仕入先出荷およびデビット要求が指定した要求番号で正常に作成されたことを検証することもできます。

BPEL プロセスによる SOA コンポジット・アプリケーションをテストするには:

1. Oracle Enterprise Manager Fusion Middleware Control コンソール
(<http://<hostname>:<port>/em>)にナビゲートします。ログイン・ページが表示されます。

2. インストール時に指定したユーザー名とパスワード情報を入力します。「**ログイン**」をクリックして、ファームにログインします。デプロイしたコンポジット(ShipNotice)がアプリケーション・ナビゲーション・ツリーに表示されます。

複数のターゲット Oracle Enterprise Manager Fusion Middleware Control コンソール・ファームがある場合、適切なターゲット・インスタンス・ファームを選択する必要がある場合があります。

3. 「ファーム」のナビゲーション・ペインから、ツリーの「SOA」→「soa-infra」ノードを展開して、「SOA インフラストラクチャ」のホーム・ページおよびメニューを使用して、soa-infra 管理対象サーバーで実行されているデプロイ済の SOA コンポジット・アプリケーションにアクセスします。

ZZ_CreateSingle_ShipDebitRequest [1.0]のリンクをクリックします。

4. ZZ_CreateSingle_ShipDebitRequest [1.0]のホーム・ページで、「**テスト**」をクリックします。
5. インスタンスを開始するための「**Web サービスのテスト**」ページが表示されます。「引数の入力」セクションには、使用する XML ペイロード・データとして SD- Request1 を指定できます。

注意: ここで入力された要求番号は、Oracle Trade Management のユーザー間で仕入先出荷およびデビット番号として使用されるため、この番号はプロセスを開始するたびに一意である必要があります。

「**Web サービスのテスト**」をクリックして、プロセスを開始します。

完了時に、「レスポンス」タブにテスト結果が表示されます。

6. コンソールでの SOAP レスポンスの確認

「レスポンス」タブで、「メッセージ・フロー・トレースの起動」リンクをクリックして、同期コンポジット・アプリケーションの結果を表示します。「フローのトレース」ページが表示されます。

「トレース」セクションで、すべてのコンポーネントの状態が、アプリケーションが正常に実行されたことを示す「完了」であることを確認します。

「フォルト」セクションで、テスト中にエラーが発生したかどうかを確認できます。

7. BPEL サービス・コンポーネント・インスタンスのリンク

(ZZ_CreateSingle_ShipDebitRequest など)をクリックして、「インスタンス」ページを表示し、その「監査証跡」タブで BPEL アクティビティの実行の詳細を確認できます。

「フロー」タブをクリックして、BPEL プロセスのフロー・ダイアグラムを確認します。プロセス・ダイアグラムのアクティビティをクリックして、アクティビティの詳細と、プロセスを介したペイロードのフローを表示します。

8. プロセス・フロー・チャートの「Invoke_ZZ_CreateSingle_ShipDebitRequest」アイコンをダブルクリックし、「**XML 文書の表示**」リンクをクリックして XML ファイルを開きます。このファイルには、トランザクション用に返されるリクエスト ID が記録されます。

9. Oracle E-Business Suite に trademgr ユーザーとしてログインし、「Oracle Trade Management ユーザー」職責を選択します。ナビゲーション・メニューから「仕入先出荷およびデビット」のリンクを選択して、「出荷およびデビットの概要」ウィンドウを開きます。
10. ステップ 5 で入力した要求番号 SD-Request1 がリストに表示されているかを確認します。
11. 要求番号 SD-Request のリンクをクリックします。「出荷およびデビット要求詳細」ページが表示され、要求の詳細を確認できます。

Oracle Workflowビジネス・イベント・システム を使用したWebサービスの起動

この章では、次のトピックを取り上げます。

- Oracle Workflow およびサービス起動フレームワークの概要
- サービス起動フレームワークを使用した Web サービス起動
- Web サービス応答での Oracle E-Business Suite へのコールバック
- Web サービスの起動
- エラーの管理
- Web サービス起動のテスト
- Web サービス起動失敗のトラブルシューティング
- シード Java ルール関数の拡張
- その他の起動使用の考慮事項

Oracle Workflow およびサービス起動フレームワークの概要

Oracle E-Business Suite Integrated SOA Gateway は、Oracle Workflow Java ビジネス・イベント・システムを利用することで、Oracle E-Business Suite からのネイティブな Web サービス起動を実現するインフラストラクチャとして機能します。

Oracle Workflow は、Oracle E-Business Suite 内の主要プロセス管理ソリューションです。これはいくつかの主要コンポーネントで構成されており、これらのコンポーネントを利用して、プロセス・ダイアグラムに含まれるビジネス・プロセスやアクティビティをユーザー定義のビジネス・ルールに基づいてモデル化および自動化したり、ルーティング・メカニズムを通じてプロセスにおける各意思決定者を支援できます。また、システム間でのイベントやサービスのサブスクリプションを円滑化したり、実行時にワークフロー・プロセス定義を実装したり、さらにはエラー処理を行うことができます。Oracle E-Business Suite Integrated SOA Gateway は、Oracle E-Business Suite における複雑なビジネス・プロセスの管理と効率化、および高度に統合されたワークフローのサポートを実現する総合ソリューションとして、Oracle Workflow を介してサービス起動プロセスを有効にし、次の機能を提供します。

- ビジネス・イベント・システムを利用して、イベントやイベント・サブスクリプションを作成したり、サブスクリプション・パラメータとして使用される Web サービスを表す特定の WSDL を解析します。
- Oracle Workflow のシード Java ルール関数
`oracle.apps.fnd.wf.bes.WebServiceInvokerSubscription` を使用して、Web サービスを容易に起動できるようにします。
- Oracle Workflow の「ビジネス・イベントのテスト」ページを利用して、PL/SQL または Java から起動側イベントを呼び出し、そのイベントに対する同期/非同期サブスクリプションを実行することによって、サービス起動をテストします。
- ビジネス・イベント・システムに用意されているエラー処理機能を利用して、サブスクリプション実行時のエラーを管理し、エラー通知を Web サービス定義、エラーおよびイベント詳細とともに SYSADMIN ユーザーに送信します。
- ワークフローの通知システムを利用して、SYSADMIN ユーザーにエラー通知を送信したり、同ユーザーからの応答を処理します。

Oracle Workflow の詳細は、*Oracle Workflow ユーザーズ・ガイド*、*Oracle Workflow 開発者ガイド*および *Oracle Workflow 管理者ガイド*を参照してください。

サービス起動フレームワークを使用して Web サービスの起動を円滑化する方法をよりよく理解できるように、この章では、次のトピックについて説明します。

- サービス起動フレームワークを使用した Web サービス起動、ページ 12-2
- Web サービス応答での Oracle E-Business Suite へのコールバック、ページ 12-27
- Web サービスの起動、ページ 12-28
- エラーの管理、ページ 12-35
- Web サービス起動のテスト、ページ 12-36
- Web サービス起動失敗のトラブルシューティング、ページ 12-40
- シード Java ルール関数の拡張、ページ 12-46
- その他の起動使用の考慮事項、ページ 12-53

サービス起動フレームワークを使用した Web サービス起動

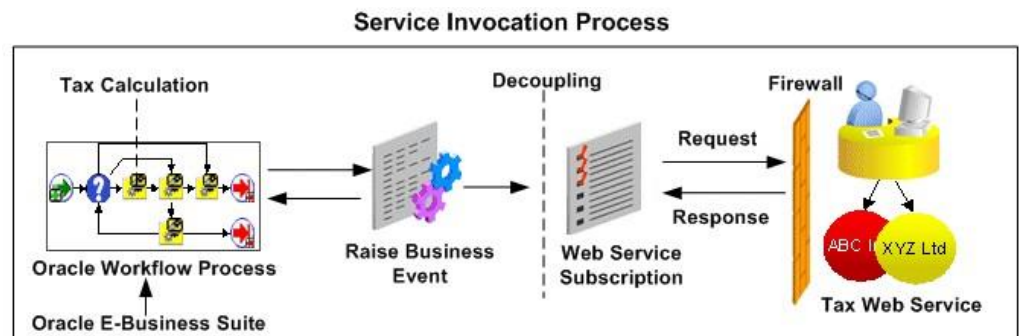
サービス起動フレームワークは、開発者が WSDL の記述を通じて Web サービスと対話したり、Oracle E-Business Suite から Web サービスを起動することを可能にするインフラストラクチャです。

この目的を満たすために、起動フレームワークでは、Oracle Workflow ビジネス・イベント・システムのウィザード・ベースのユーザー・インタフェースを使用して、サブスクリプション作成時に Web サービスの特定の WSDL URL を解析し、識別したサービス情報またはメタデータをサブスクリプション・パラメータとして格納し、後でサービスを起動する際に使用します。

WSDL URL は、Web サービスを表すために使用されます。基礎となるサービスは、単純なネイティブ Web サービスでも、BPEL プロセスでもかまいません。

ここで説明するサービス起動フレームワークでは、文書ベースの Web サービス起動のみがサポートされていることに注意してください。起動フレームワークでは、RPC (リモート・プロシージャ・コール) 形式の Web サービス起動はサポートされていません。

次の図は、サービス起動プロセス・フローの概要を示しています。



実行時に Web サービスを正常に起動するには、Web サービス起動メタデータが存在する必要があります。この項では、起動メタデータの定義に加え、サービス起動フレームワークでサポートされているメッセージ・パターン、Web サービス入力メッセージ・パートおよび Web サービス・セキュリティの概念についても説明します。

この項では、次のトピックを取り上げます。

- メッセージ・パターンについて、ページ 12-3
- Web サービス起動メタデータの定義、ページ 12-5
- 1. Web サービス起動側ビジネス・イベントの作成、ページ 12-6
- 2. 起動側イベントに対するローカル・サブスクリプションおよびエラー・イベント・サブスクリプションの作成、ページ 12-7
- 3. 受入イベントおよびイベント・サブスクリプションの作成(オプション)、ページ 12-14
- Web サービス入力メッセージ・パートについて、ページ 12-16
- WS セキュリティのサポート、ページ 12-21

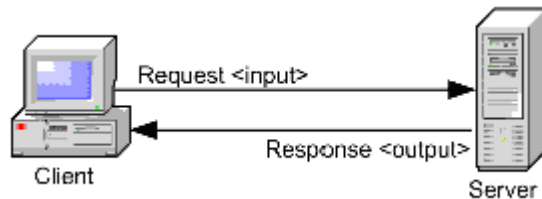
WSDL のメッセージ・パターンについて

主なメッセージ交換パターンには、要求応答パターンと一方向(要求のみ)パターンの 2 つがあります。

要求応答メッセージ・パターン

要求応答メッセージ交換パターンでは、クライアントがサービス・プロバイダに質問をし、その質問に対する回答を受け取ります。回答はフォルトまたは例外の形で返されることもあります。要求と応答の両方が個別のメッセージです。要求応答パターンは、通常、単純な操作であれば同期操作を使用して実装されます。長期にわたる操作に対しては、通常、非同期(メッセージ相関あり)が選択されます。

要求応答メッセージ・パターン



- **同期操作**とは、応答を待ってから処理を進める操作のことです。この場合、操作は強制的に順番に発生します。このような操作に対しては「ブロックされる(つまり応答待ちになる)」という表現がよく使用されます。多くのオンライン・バンキング・タスクは要求応答モードでプログラムされています。

たとえば、口座残高問合せ要求は次のように実行されます。

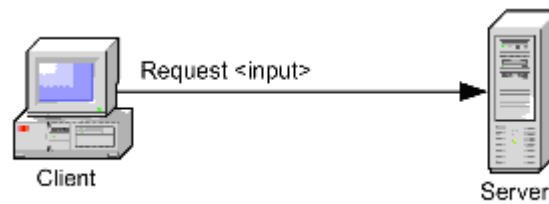
- 顧客(クライアント)が口座記録保管システム(サーバー)に口座残高問合せ要求を送信します。
 - 口座記録保管システム(サーバー)が顧客(クライアント)に指定口座の金額を示す応答を送信します。
-
- **非同期操作**とは、応答を待たずに処理を進める操作のことです。この場合、操作は同時に発生することが可能になります。このため、操作がブロックされる(つまり応答待ちになる)ことはありません。非同期操作では、クライアントは、遅延している応答を待っている間に処理を続行できます。この場合、バックグラウンドでスレッドを実行する非同期ハンドルを返すことによって、応答の準備が整うまでの間にクライアントが処理を続行できるようにします。

重要: このリリースの Web サービス起動フレームワークでは、同期要求応答メッセージ・パターンと一方向(要求のみ)メッセージ・パターンのみがサポートされています。

要求のみメッセージ・パターン

要求のみ操作モデルでは、1つの**入力要素**、つまりサーバーに対するクライアントの要求のみを扱います。応答はありません。

要求のみメッセージ・パターン



たとえば、要求のみ操作では、クライアントの郵便番号位置における天候状況が変化したときに、更新された天候データがサービスに送信されます。サーバーでは、データは更新されますが、応答は返されません。

Web サービス起動メタデータの定義

Web サービスを起動する前に Oracle Workflow ビジネス・イベント・システムでサービス起動が実行されるため、イベントやイベント・サブスクリプションなどの Web サービス起動メタデータをビジネス・イベント・システムで事前に定義しておく必要があります。

この項では、次のトピックについて説明します。

1. Web サービス起動側ビジネス・イベントの作成、ページ 12-6

サービスに対する要求メッセージ(または Web サービス入力メッセージ)として機能する Web サービス起動側ビジネス・イベントを事前に作成しておく必要があります。

2. 起動側イベントに対するローカル・サブスクリプションおよびエラー・イベント・サブスクリプションの作成、ページ 12-7

起動側イベントの定義後、次の 2 つのサブスクリプションを作成する必要があります。

- 「Web サービスの起動」処理タイプを使用したローカル・サブスクリプションの作成、ページ 12-7

このイベント・サブスクリプションを使用すると、トリガー・イベントが発生したときに、このサブスクリプションの処理項目により、このサブスクリプションの一部として定義された Web サービスが起動されます。

- 「ワークフローの起動」処理タイプを使用したエラー・サブスクリプションの作成、ページ 12-12

このエラー・サブスクリプションを使用すると、ビジネス・イベント・システムでのエラー処理が有効になり、サブスクリプションの実行時にエラー状態が SYSADMIN ユーザーに伝達されます。

3. 受入イベントおよびイベント・サブスクリプションの作成(オプション)、ページ 12-14

このステップは、Oracle E-Business Suite への伝達またはコールバックのために、Web サービスによる出力または応答メッセージが存在する場合にのみ必要です。受入イベントを設定した場合は、その受入イベントに対する外部サブスクリプションを作成して、

Web サービス応答メッセージが渡されるようにする必要があります。

Web サービスで応答が不要な場合は、受入イベントもイベント・サブスクリプションも作成する必要はありません。

ステップ 1: Web サービス起動側ビジネス・イベントの作成

ビジネス・イベントとは、インターネットまたはイントラネット・アプリケーションやプログラム内における、システム内の他のオブジェクトや外部エージェントに影響を及ぼす可能性がある事象です。たとえば、発注の作成は、購買アプリケーションにおけるビジネス・イベントの一例です。

Web サービス起動側ビジネス・イベントを定義するには、Oracle Workflow ビジネス・イベント・システムを使用します。

起動側イベントは、メッセージ・パターンにおける要求メッセージ(または Web サービス入力メッセージ)として、サービスに照会を送信するために使用されます。

ビジネス・イベント・システムから Web サービスを起動するには、起動側ビジネス・イベントを作成してから、後で適切な処理タイプを使用して起動側イベントを購読する必要があります。

注意: このリリースの Web サービス起動フレームワークでは、次のタイプのサービス起動がサポートされています。

- **一方向(要求のみ)サービス:** 消費者またはクライアントがサービスにメッセージを送信し、サービスは応答を返す必要がありません。
- **同期要求応答サービス・タイプ:** 操作を続行するには応答が必要です。

起動側イベントで応答を求める場合は、Web サービスが正常に起動された後に Oracle E- Business Suite に伝達またはコールバックされるように、受入ビジネス・イベントを定義する必要があります。「受入イベントおよびイベント・サブスクリプションの作成(オプション)」(ページ 12-14)を参照してください。

ビジネス・イベントの詳細は、*Oracle Workflow 開発者ガイド*の「イベント」を参照してください。

起動側イベントを作成するには:

1. 「ワークフロー管理者 Web(新規)」職責を持つユーザーとして Oracle E-Business Suite にログインします。「イベント」ページがまだ表示されていない場合は、「ビジネス・イベント」リンクを選択し、水平ナビゲーションで「イベント」を選択します。
2. 「イベント」ページで、「**イベントの作成**」をクリックして「イベントの作成」ページを開きます。
3. 「イベントの作成」ページで、次の情報を入力します。

- 名前: イベント名を入力します
(oracle.apps.xxx.user.webservice.invoke など)
- 表示名: イベントの表示名を入力します
(oracle.apps.xxx.user.webservice.invoke など)
- 説明: イベントの説明を入力します
- ステータス: 使用可能
- 関数の生成: PL/SQL ベースのイベントについて、イベントの発生元アプリケーションからイベント・データが提供されない場合は、その生成機能を指定します
- Java 生成機能: Java ベースのイベントについて、イベントの発生元アプリケーションからイベント・データが提供されない場合は、その生成機能を指定します
- 所有者名: イベントを所有するプログラムまたはアプリケーション名を指定します
(「Oracle Workflow」など)
- 所有者タグ: イベントを所有するプログラムまたはアプリケーション ID を指定します
(「FND」など)

4. 「適用」をクリックして作業内容を保存します。

このページを開いたまま、受入イベントを作成します。

ビジネス・イベントの作成方法の詳細は、*Oracle Workflow 開発者ガイド*を参照してください。

ステップ 2: 起動側イベントに対するローカル・サブスクリプションおよびエラー・イベント・サブスクリプションの作成

- 「Web サービスの起動」処理タイプを使用したローカル・サブスクリプションの作成、
ページ 12-7

このイベント・サブスクリプションを使用すると、トリガー・イベントが発生したときに、このサブスクリプションの処理項目により、起動イベントで作成した Web サービスが起動されます。

- 「ワークフローの起動」処理タイプを使用したエラー・サブスクリプションの作成、ページ
12-12

このエラー・サブスクリプションを使用すると、ビジネス・イベント・システムでのエラー処理が有効になり、サブスクリプションの実行時にエラー状態が SYSADMIN ユーザーに伝達されます。

Web サービスの定義、エラー詳細およびイベント詳細を含むワークフロー通知が SYSADMIN に送信されるため、SYSADMIN ユーザーは必要に応じてエラーを処理できます。

「Web サービスの起動」処理タイプを使用したローカル・サブスクリプションの作成

起動側イベントを購読するには、「Web サービスの起動」処理タイプを使用したサブスクリプションを作成し、トリガー・イベントが発生したときに、このサブスクリプションの処理項目により、Web サービスが起動されるようにする必要があります。この場合、「イベント・サブスクリプ

ションの作成 - Web サービスの起動」ウィザードで任意のタイプの Web サービス(ネイティブ Web サービスや BPEL プロセスなど)を表す WSDL URL を入力する必要があります。ウィザードで入力した WSDL 情報は、解析によってサービス・メタデータに分類され、後で選択できるようになります。

注意: BPEL プロセスはそれ自身が Web サービスとして動作し、WSDL および SOAP を通じてクライアント・インタフェースを定義してサポートできます。BPEL プロセスの WSDL URL はパートナ・リンクを使用して作成でき、このリンクを介して Oracle SOA Suite に対して要求を発行し、Web サービスに接続させることができます。

トリガー・イベントが発生すると、ビジネス・イベント・システムにより、シード Java 関数を使用してサブスクリプションが実行され、BPEL プロセスが起動されます。

適切なサービス・メタデータを選択すると、選択したデータが次のサブスクリプション・パラメータとして格納され、「Web サービス詳細」リージョンに表示されます。

- SERVICE_WSDL_URL
- SERVICE_NAME
- SERVICE_PORT
- SERVICE_PORTTYPE
- SERVICE_OPERATION

注意: SOAP アドレスは、イベント・サブスクリプションの実行時に取得されるこの SERVICE_WSDL_URL パラメータから導出されますが、実行時におけるサービス起動において SERVICE_SOAP_ADDRESS というイベント・パラメータを渡すことによって上書きできます。

シード Java ルール関数 `oracle.apps.fnd.wf.bes.WebServiceInvokerSubscription` では、サービス起動時にこれらのサブスクリプション・パラメータが使用されます。

注意: Oracle E-Business Suite Integrated SOA Gateway では、開発者は、Java コーディング標準を使用して起動側サブスクリプションのシード・ルール関数を拡張し、より専門的なサービス起動処理を行うことができます。シード Java ルール関数のカスタマイズの詳細は、「シード・ルール関数の拡張」(ページ 12-46)を参照してください。

Web サービス・セキュリティの構成

起動対象の Web サービスでユーザー名/パスワードによる認証が行われる場合のために、サービス起動フレームワークは、Web サービス起動時における UsernameToken ベースの WS セキュリティ・ヘッダーの使用にも対応しています。

重要: サービス起動時にこの UsernameToken ベースの WS セキュリティ・ヘッダーが実装されるのは、Web サービス要求を処理する Web サービス・プロバイダでこのセキュリティ・ヘッダーが必要になる場合のみです。

「イベント・サブスクリプションの作成 - Web サービスの起動」ウィザードで必要な情報を入力すると、「Web サービス・セキュリティ」リージョンが表示され、ユーザー名およびパスワード情報を適宜指定または更新できるようになります。この情報は Vault に安全に格納されます。

Oracle Workflow では、カスタマイズ・レベルに基づいてビジネス・イベントやサブスクリプションに対して様々なレベルの更新が許容されることに注意してください。「Web サービスの起動」イベント・サブスクリプションのカスタマイズ・レベルが「コア」または「制限」であり、ユーザー名がサブスクリプション所有者によって指定されている場合、そのユーザー名は更新できません。ユーザー名がまだ指定されていない場合は、必要に応じて更新できます。パスワードについては、カスタマイズ・レベルに関係なく、必要に応じていつでも更新できます。

カスタマイズ・レベルを指定してセキュリティ・フィールドを構成する方法の詳細は、次を参照してください。

- カスタマイズ・レベルを指定したセキュリティ・パスワードの構成、ページ 12-22
- カスタマイズ・レベルを指定したセキュリティの構成の例、ページ 12-24

Web サービス・セキュリティの詳細は、「WS セキュリティのサポート」(ページ 12-21)を参照してください。

追加のサブスクリプション・パラメータの設定

「Web サービスの起動」サブスクリプション・ページで解析および格納されたサブスクリプション・パラメータとは別に、追加のサブスクリプション・パラメータを指定することによって、次の情報も取得できます。これらのパラメータは、Web サービス起動に対してメッセージ処理を有効にするためにシード Java ルール関数によって使用されます。

- WS セキュリティ・ヘッダー

リプレイ攻撃によるセキュリティ・ヘッダー<wsse:Security>の再利用を防ぐために、オプションで次のパラメータを使用してヘッダーに有効期限を設定できます。

- WFBES_SOAP_EXPIRY_DURATION

デフォルトでは、セキュリティ・ヘッダーの有効期限は、<wsu:Timestamp>要素で作成後 60 秒に設定されています。このパラメータでは、ヘッダーに別の有効期限(秒単位)を設定することもできます。

- メッセージ変換

起動側イベントの XML ペイロード(Web サービス入力メッセージとして使用される)をフォームに変換してから、そのフォームを入力メッセージ・スキーマとともにコンパイルする必要がある場合は、シード Java ルール関数により、Web サービスを起動する前に、ペイロードの XSL 変換を実行できます。同様に、Web サービス出力メッセージを所定のフォームに変換してから、Oracle E-Business Suite で処理する必要がある場合は、シード Java ルール関数により、Oracle E-Business Suite にコールバックする前に、応答の XSL 変換を実行できます。

- WFBES_OUT_XSL_FILENAME
- WFBES_IN_XSL_FILENAME

Web サービスの入力メッセージおよび出力メッセージで XSL 変換が必要な場合は、イベント・ペイロードがイベント呼出し時に渡されるか、イベント呼出し後に生成機能によって生成された後に、シード Java ルール関数により、これらのサブスクリプション・パラメータに基づいて XSL ファイル名が取得されます。実行時に、同名のイベント・パラメータが渡された場合は、イベント・パラメータによってサブスクリプション・パラメータが上書きされます。

これらの変換パラメータの詳細は、「Web サービス入力メッセージ・パートについて」(ページ 12-16)を参照してください。

- コールバック: Web サービス応答での Oracle E-Business Suite へのコールバック
 - WFBES_CALLBACK_EVENT
 - WFBES_CALLBACK_AGENT

Web サービスの出力または応答(同期要求応答)メッセージを処理するには、コールバック・メカニズムを使用してビジネス・イベントをインバウンド・ワークフロー・エージェン트에エンキューすることによって、そのイベントに基づく応答が Oracle E-Business Suite に返されます。新規または待機ワークフロー・プロセスを開始または実行できるようになります。

これらのコールバック・パラメータの詳細は、「Web サービス応答での Oracle E-Business Suite へのコールバック」(ページ 12-27)を参照してください。

「Web サービスの起動」処理タイプを使用したローカル・イベント・サブスクリプションの作成
「Web サービスの起動」処理タイプを使用したローカル・イベント・サブスクリプションを作成するには:

1. 「ワークフロー管理者 Web(新規)」職責を持つユーザーとして Oracle E-Business Suite にログインします。「ビジネス・イベント」リンクを選択し、水平ナビゲーションで「サブスクリプション」を選択します。
2. 「イベント・サブスクリプション」ページで、「サブスクリプションの作成」をクリックして「イベント・サブスクリプションの作成」ページを開きます。
3. 「イベント・サブスクリプションの作成」ページで、次の情報を入力します。
 - サブスクライバ: ローカル・システムを選択します
 - ソース・タイプ: ローカル

- イベント・フィルタ: 作成したばかりのイベント名 (oracle.apps.xxx.user.webservice.invoke など)を選択します
- フェーズ: 90

Java からイベントが呼び出された場合は、フェーズ番号に基づいて、イベントがすぐに起動されるか、WF_JAVA_DEFERRED キューにエンキューされるかが決まります。

注意: PL/SQL から呼び出された起動側イベントは、データベースではサブスクリプションで使用される Java ルール関数を実行できないため、フェーズに関係なく、常に、WF_JAVA_DEFERRED キューで遅延実行されます。

- フェーズが 100 以上であれば、イベントは WF_JAVA_DEFERRED キューにエンキューされ、後で作業手配されます。
 - フェーズが 100 未満であれば、イベントはトリガー・イベント発生後すぐに Java ビジネス・イベント・システムに作業手配されます。
 - ステータス: 使用可能
 - ルール・データ: メッセージ
 - 処理タイプ: Web サービスの起動
 - エラー時: 停止およびロールバック
4. 「次へ」をクリックします。これにより、「イベント・サブスクリプションの作成 - Web サービスの起動」ウィザードが開き、WSDL URL を入力できます。入力した WSDL URL は、解析によってサービス・メタデータに分類され、後で選択できるようになります。
1. 起動対象の Web サービスの WSDL URL 情報を入力します。「次へ」をクリックして WSDL を解析し、すべてのサービスを表示します。
 2. ドロップダウン・リストから適切なサービス名を選択します。「次へ」をクリックして選択したサービスのすべてのポートを表示します
 3. 適切なサービス・ポートを選択し、「次へ」をクリックして選択したポートのすべての操作を表示します。
 4. 適切なサービス操作を選択し、「次へ」をクリックします。これにより、「イベント・サブスクリプションの作成 - Web サービスの起動」ウィザードの最後のページが開きます。
 5. 選択したすべてのサービス・メタデータ情報が「Web サービス詳細」リージョンに自動的に表示されます。

6. 「Web サービス・セキュリティ」リージョンで、ユーザー名およびパスワード情報を適宜入力します。

Oracle Workflow では、カスタマイズ・レベルに基づいてビジネス・イベントやサブスクリプションに対して様々なレベルの更新が許容されることに注意してください。

「Web サービスの起動」イベント・サブスクリプションのカスタマイズ・レベルが「コア」または「制限」であり、ユーザー名がサブスクリプション所有者によって指定されている場合、そのユーザー名は更新できません。ユーザー名がまだ指定されていない場合は、必要に応じて更新できます。パスワードについては、カスタマイズ・レベルに関係なく、必要に応じていつでも更新できます。

詳細は、「カスタマイズ・レベルを指定したセキュリティ・パスワードの構成」(ページ 12-22)を参照してください。

7. 「Web サービス起動側」リージョンで、デフォルトの Java ルール関数名 `oracle.apps.fnd.wf.bes.WebServiceInvokerSubscription` が自動的に移入されます。

重要: シード・ルール関数の機能を拡張した場合は、ここにカスタム関数名を手動で入力してください。

8. 「ドキュメント」リージョンで、サブスクリプションを所有するアプリケーション名またはプログラム名(「Oracle Workflow」など)を「所有者名」フィールドに入力します。プログラム ID (「FND」など)を「所有者タグ」フィールドに入力します。「適用」をクリックします。

詳細は、*Oracle Workflow 開発者ガイド*の「イベント・サブスクリプションの定義」を参照してください。

「ワークフローの起動」処理タイプを使用したエラー・サブスクリプションの作成

サービス起動時におけるエラー処理機能を有効にするには、起動側ビジネス・イベントに対するエラー・サブスクリプションを作成する必要があります。

このエラー処理を購読すると、起動時にエラーが発生した場合に、エラー・プロセスによって SYSADMIN ユーザーにワークフロー通知が送信されるようになります。この情報には、SYSADMIN ユーザーがエラーを簡単に特定できるように、Web サービス定義、イベント詳細およびエラー詳細が含まれています。通知には、SYSADMIN ユーザーがエラーに応答するためのオプションも用意されています。SYSADMIN ユーザーは、エラーの原因となった根本的問題を解決してから再び Web サービスを起動したり、エラーが発生したイベントを必要に応じて中止したり、エラー通知を別のユーザーに適宜再割当てできます。

Web サービス起動時のエラー管理の詳細は、「エラーの管理」(ページ 12-35)を参照してください。

「ワークフローの起動」処理タイプを使用したエラー・サブスクリプションを作成するには:

1. 「ワークフロー管理者 Web(新規)」職責を持つユーザーとして Oracle E-Business Suite にログインします。「ビジネス・イベント」リンクを選択し、水平ナビゲーションで「サブスクリプション」を選択します。

2. 「イベント・サブスクリプション」ページで、「サブスクリプションの作成」をクリックして「イベント・サブスクリプションの作成」ページを開きます。

3. 「イベント・サブスクリプションの作成」ページで、次の情報を入力します。

- サブスクライバ: ローカル・システムを選択します
- ソース・タイプ: エラー
- イベント・フィルタ: 作成したばかりのイベント名
(oracle.apps.xxx.user.webservice.invoke など)を選択します
- フェーズ: 任意のフェーズ番号を指定できます。
- ステータス: 使用可能

注意: カスタマイズ・レベルが「制限」に設定されたシード・イベントについては、イベントとサブスクリプションの更新時にステータスしか更新できません。カスタマイズ・レベルが「コア」に設定された製品固有のシード・イベントについては、どのプロパティも更新できません。サブスクリプション定義の表示のみが可能です。

カスタマイズ・レベルの使用の詳細は、「イベント・サブスクリプション・ユーザー・インタフェースを使用した Web サービス・セキュリティの構成」(ページ 12-22)を参照してください。

- ルール・データ: キー
- 処理タイプ: ワークフローの起動
- エラー時: 停止およびロールバック

4. 「次へ」をクリックして「イベント・サブスクリプションの作成 - ワークフローの起動」ページを開きます。

5. 「処理」リージョンで、次の情報を入力します。

- ワークフロー・タイプ: WFERROUR
- ワークフロー・プロセス: DEFAULT_EVENT_ERROR2
- 優先度: 通常

6. 「ドキュメント」リージョンで、イベント・サブスクリプションを所有するアプリケーションまたはプログラム名(「Oracle Workflow」など)を「所有者名」フィールドに入力し、アプリケーションまたはプログラム ID (「FND」など)を「所有者タグ」フィールドに入力します。

7. 「適用」をクリックします。

ステップ 3: 受入イベントおよびサブスクリプションの作成(オプション)

受入イベントは、Web サービスが正常に起動された後に出力または応答メッセージを返す必要がある場合に、Oracle E-Business Suite への伝達またはコールバックの手段として使用できます。ただし、受入イベントとそのイベントに対する外部サブスクリプションを作成する必要があるかどうかは、次の基準によって決まります。

- 使用するメッセージ・パターン
- イベントの呼出し元(Java または PL/SQL レイヤー)
- イベント・サブスクリプションのフェーズ番号

同期要求応答の Web サービス起動の場合

- Web サービス起動側イベントが中間層の Java コードから呼び出され、起動側サブスクリプションがサブスクリプション・フェーズ 100 未満の同期型サブスクリプションである場合は、イベントが呼び出されるとすぐに Web サービスが起動されます。起動が成功した場合は、コール側アプリケーションから応答を読み取ることができます。

`BusinessEvent.raise()` をコールした後、

`BusinessEvent.getResponseData()` メソッドを使用することによって、即座に応答を取得できます。

このケースでは、コールバック・イベントを使用して Oracle E-Business Suite に応答を返す必要はありません。したがって、受入イベントとそのイベントに対するサブスクリプションを作成する必要はありません。

- Web サービス起動側イベントが Java コードから呼び出され、サブスクリプション・フェーズが 100 以上であるか、イベントが PL/SQL から呼び出された場合は、イベント・メッセージは `WF_JAVA_DEFERRED` キューにエンキューされます。この状況では、Web サービスによる出力または応答メッセージが存在する場合、受入イベントとそのイベントに対する外部サブスクリプションを作成する必要があります。Oracle E-Business Suite で出力メッセージを受信するためには、コールバック・エージェントによるコールバック・イベントが必要です。

この受入イベントを Oracle E-Business Suite へのコールバックとして使用することで、このイベントの呼出しを通して、関係者に Web サービス応答の存在を認識させることもできます。

参照: Web サービス応答での Oracle E-Business Suite へのコールバック、ページ 12-27。

受入イベントが必要な場合は、受入イベントの作成後に、そのイベントに対する外部イベント・サブスクリプションも作成する必要があります。受入イベントを介して伝達される Web サービス応答メッセージは、常にインバウンド・ワークフロー・エージェントにエンキューされます。インバウンド・ワークフロー・エージェント内のイベントを処理するには、外部サブスクリプションが必要です。

要求のみの Web サービスの場合

応答が不要な要求のみの Web サービスに対しては、受入イベントを作成する必要はありません。

受入イベントを作成するには:

1. 「イベント」ページで、「**イベントの作成**」をクリックして別の「イベントの作成」ページを開きます。
2. 「イベントの作成」ページで、次の情報を入力します。
 - 名前: イベント名を入力します
(`oracle.apps.xxx.user.webservice.receive` など)
 - 表示名: イベントの表示名を入力します
(`oracle.apps.xxx.user.webservice.receive` など)
 - 説明: イベントの説明を入力します
 - ステータス: 使用可能
 - 所有者名: イベントを所有するアプリケーションまたはプログラム名を入力します
(「Oracle Workflow」など)
 - 所有者タグ: イベントを所有するアプリケーションまたはプログラム ID を入力します
(「FND」など)

3. 「**適用**」をクリックして受入イベントを作成します。

受入イベント・サブスクリプションを作成するには:

1. 「ワークフロー管理者 Web アプリケーション」職責を持つユーザーとして Oracle E-Business Suite にログインします。「ビジネス・イベント」リンクを選択し、水平ナビゲーションで「サブスクリプション」を選択します。
2. 「イベント・サブスクリプション」ページで、「**サブスクリプションの作成**」をクリックして「イベント・サブスクリプションの作成」ページを開きます。
3. 「イベント・サブスクリプションの作成」ページで、次の情報を入力します。
 - サブスクライバ: ローカル・システムを選択します
 - ソース・タイプ: 外部
 - イベント・フィルタ: 作成したばかりの受入イベント名
(`oracle.apps.xxx.user.webservice.receive` など)を選択します
 - フェーズ: 任意のフェーズ番号
 - ステータス: 使用可能
 - ルール・データ: キー

- 処理タイプ: 任意の処理タイプ
 - エラー時: 停止およびロールバック
4. 「次へ」をクリックして「イベント・サブスクリプションの作成 - ワークフローの起動」ページを開きます。
- 表示される「イベント・サブスクリプションの作成」ページのタイプは、「処理タイプ」フィールドで選択した値によって異なることに注意してください。「ワークフローの起動」を選択した場合は、「イベント・サブスクリプションの作成 - ワークフローの起動」ページが表示されます。他の処理タイプを選択した場合は、別のタイプのイベント・サブスクリプションの作成ページが表示されます。サブスクリプション・ページで適切な処理タイプを入力することによって、該当するサブスクリプションの一部として定義されたイベントに対して、ワークフロー・プロセスを起動したり、カスタム・ルール関数を実行できます。
5. 「処理」リージョンで、次の情報を入力します。
- ワークフロー・タイプ: 応答待ちのワークフロー・タイプを入力します
 - ワークフロー・プロセス: 応答待ちのワークフロー・プロセスを入力します
 - 優先度: 通常
6. 「ドキュメント」リージョンで、アプリケーションまたはプログラム名（「Oracle Workflow」など）を「所有者名」フィールドに入力します。アプリケーションまたはプログラム ID（「FND」など）を「所有者タグ」フィールドに入力します。
7. 「適用」をクリックします。

Web サービス入力メッセージ・パートについて

メッセージは 1 つ以上の論理パートで構成されています。各パートは、メッセージの論理的な抽象コンテンツを表します。たとえば、一般的な文書形式の Web サービスの場合、入力メッセージはヘッダー・パートと本文パートで構成されています。

例として、次に示す Oracle E-Business Suite XML ゲートウェイ・サービス (<http://<host>:<port>/webservices/SOAPProvider/xmlgateway/ont poi/?wsdl>) の操作 PROCESSPO について考えます。

```

<definitions targetNamespace="ONT_POI" targetNamespace="http://xmlns.
oracle.com/apps/ont/soapprovider/xmlgateway/ont poi/">
<type>
<schema elementFormDefault="qualified"
targetNamespace="http://xmlns.oracle.com/apps/ont/soapprovider/xmlgatew
ay/ont poi/">
<include schemaLocation="http://<host>:
<port>/webservices/SOAProvider/xmlgateway/ont
poi/PROCESS_PO_007.xsd"/>
</schema>
...
<message name="PROCESSPO_Input_Msg">
  <part name="header" element="tns:SOAHeader"/>
  <part name="body" element="tns1:PROCESS_PO_007"/>
</message>
...
<binding name="ONT_POI_Binding" type="tns:ONT_POI_PortType">
<soap: binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
<operation name="PROCESSPO">
<soap:operation soapAction="http://<host>:
<port>/webservices/SOAProvider/xmlgateway/ont poi/">
<input>
<soap:header message="tns:PROCESSPO_Input_Msg" part="header" use="
literal"/>
<soap:body parts="body" use="literal"/>
</input>
</operation>
</binding>
...
</definitions>

```

操作 PROCESSPO では、次の 2 つのパートで構成される入力メッセージ
PROCESSPO_Input_Msg が必要になります。

- 本文: SOAP 本文として設定される PROCESS_PO_007 タイプの値。この値はビジネス・イベント・ペイロードとして送信されます。
- ヘッダー: SOAP ヘッダーで送信される SOAHeader タイプの値。この値は Web サービスでの承認に必要です。

Web サービス操作の入力メッセージをよりよく理解できるように、この項では、次のトピックを取り上げます。

- SOAP 本文としてのイベント・ペイロード、ページ 12-17
- その他の Web サービス入力メッセージ・パート、ページ 12-20

SOAP 本文としてのイベント・ペイロード

イベント名とイベント・キーに加え、イベントで発生したことを表すために必要な詳細情報のことをイベント・データと呼びます。たとえば、発注イベントのイベント・データには、品目番号、摘要、原価などがあります。

イベントの作成時には、生成機能を使用するかどうかを問わず、PL/SQL と Java のいずれのイベントに対してもイベント・データを指定できます。イベントの発生元アプリケーションからイベント・データが提供されない場合は、イベントの作成時に生成機能を使用できます。生成機能を使用すると、イベント呼出し時に、イベント名、イベント・キーおよびオプションのパラメータ・リストに基づいて完全なイベント・データが生成されます。一方、イベントの発生元アプリケーションからイベント・データが提供される場合は、関数の生成フィールドを指定する

必要はありません。言い換えると、次のいずれかの方法でイベント・ペイロードを渡すことができます。

- イベント呼出し中に、生成機能を使用してイベント・データまたはペイロードを渡します。
- 生成機能を使用しないで、イベント自体とともにイベント・データまたはペイロードを渡します。

注意: 生成機能では標準の PL/SQL または Java API に従う必要があります。
*Oracle Workflow 開発者ガイド*および *Oracle Workflow API リファレンス*を参照してください。

イベント・データは、イベント呼出し時に、XML 文書として構造化し、SOAP 本文として渡すことができます。この SOAP 本文は、ビジネス・イベント・ペイロードを介してシード Java ルール関数に受け入れられます。SOAP 本文は、整形 XML 要素で表され、SOAP エンベロープに埋め込まれます。

- `BusinessEvent.setData(String)`
- `WF_EVENT.Raise(?. p_event_data => ?.);`

XSL 変換をサポートするためのメッセージ変換パラメータ

起動側イベントの XML ペイロード(Web サービス入力メッセージとして使用される)をフォームに変換してから、そのフォームを入力メッセージ・スキーマとともにコンパイルする必要がある場合は、シード Java ルール関数により、Web サービスを起動する前に、ペイロードの XSL 変換を実行できます。同様に、Web サービス出力メッセージを所定のフォームに変換してから、Oracle E-Business Suite で処理する必要がある場合は、シード Java ルール関数により、Oracle E-Business Suite にコールバックする前に、応答の XSL 変換を実行できます。

注意: 入力メッセージは、SOAP 要求で Web サービスに渡される XML ペイロードです。出力メッセージは、Web サービスが正常に起動された後に応答として返される XML 文書です。

同期要求応答操作で、出力(応答)メッセージとして XML 文書が生成され、その XML 文書を Oracle E-Business Suite が認識しやすいフォームに変換する必要がある場合は、出力メッセージに対して XSL 変換が実行されます。

注意: XSL ファイル名は、<File Name>:

<Application Short Name>:<Version>の形式で指定されます。

たとえば、「PO_XSL_1_1_2.xsl:FND:1.1」のようになります。

「イベント・サブスクリプションの作成 - Web サービスの起動」ウィザードで Web サービス起動側イベントに対するサブスクリプションを作成する際には、XSL ファイル名が次のサブスクリプション・パラメータとしてシード Java ルール関数に渡されます。

- **WFBES_OUT_XSL_FILENAME:** 出力(応答)メッセージに対して変換を実行する XSL ファイル

例: WFBES_OUT_XSL_FILENAME=PO_XSL_OUT_2.xsl:FND:1.1

- **WFBES_IN_XSL_FILENAME:** 入力メッセージに対して変換を実行する XSL ファイル

例: WFBES_IN_XSL_FILENAME=PO_XSL_IN_2.xsl:FND:1.1

実行時には、同じパラメータをイベント・パラメータとして使用して XSL ファイル名が渡されます。解析および格納されたサブスクリプション・パラメータと同名のイベント・パラメータが渡された場合は、イベント・パラメータの値によってサブスクリプション・パラメータの値が上書きされます。たとえば、イベント・パラメータは次のように渡されます。

- `BusinessEvent.setStringProperty("WFBES_OUT_XSL_FILENAME", "PO_XSL_OUT_2.xsl:FND:1.1");`
- `BusinessEvent.setStringProperty("WFBES_IN_XSL_FILENAME", "PO_XSL_IN_2.xsl:FND:1.1");`

WFBES_OUT_XSL_FILENAME が null である場合、アウトバウンド変換は実行されません。WFBES_IN_XSL_FILENAME が null である場合、インバウンド変換は実行されません。

Oracle E-Business Suite への XSL ファイルのロード

入力メッセージおよび出力メッセージに対する XSL 変換は、シード Java ルール関数により、XML ゲートウェイ API の `ECX_STANDARD.perform_xslt_transformation` を使用して実行されます。したがって、`oracle.apps.ecx.loader.LoadXSLTToClob` ロードャーを使用して、入力メッセージおよび出力メッセージに対する XSL 変換用の XSL ファイルが Oracle XML Gateway にロードされます。

注意: XSL 変換用の PL/SQL API の詳細は、*Oracle XML Gateway ユーザーズ・ガイド*の「実行エンジン API」を参照してください。

したがって、サービス起動時に次のステップを使用して、XSL 変換を実行します。

1. Oracle XML Gateway で `oracle.apps.ecx.loader.LoadXSLTToClob` ロードャーを使用して、Oracle E-Business Suite に XSL ファイルをアップロードします。
2. 入力メッセージおよび出力メッセージのどちらに対して XSL 変換を実行するかに応じて、適切な XSL ファイル名(`PO_XSL_IN_2.xsl:FND:1.1` など)をイベント・パラメータまたはサブスクリプション・パラメータ(`WFBES_IN_XSL_FILENAME` および `WFBES_OUT_XSL_FILENAME`)で指定します。

たとえば、次のように Oracle E-Business Suite に XSL ファイルをアップロードします。

```
java oracle.apps.ecx.loader.LoadXSLTToClob apps password
ap601sdb:4115:owf12dev PO_XSL_IN_2.xsl FND 1.1
```

詳細は、*Oracle XML Gateway ユーザーズ・ガイド*の「XSLT スタイル・シートのロードおよび削除」を参照してください。

その他の Web サービス入力メッセージ・パート

サービス起動フレームワークでは、SOAP 本文パートをイベント・ペイロードとして渡すだけでなく、Web サービス操作の入力メッセージに定義されたその他のパートの値も渡すことができます。その場合、次の形式のビジネス・イベント・パラメータを使用します。

WFBES_INPUT_<partname>

<partname>は、WSDL での入力メッセージ定義に含まれるパート名と同じです。

たとえば、前述の例のヘッダー・パートは、起動側イベントの呼出し時に、パラメータ **WFBES_INPUT_header** としてビジネス・イベントに渡されます。次のコード・スニペットは、Web サービスでの承認に必要なユーザー名、職責、職責アプリケーションおよび NLS 言語要素を渡すために使用されるヘッダー・パートを示しています。

```
String headerPartMsg = "<SOAHeader
xmlns:=\"http://xmlns.oracle.com/xdb/SYSTEM\" \" \" +
\"env:mustUnderstand=\"0\"
xmlns:env=\"http://schemas.xmlsoap.org/soap/envelope/\"> \n\" +
\" <MESSAGE_TYPE>XML<MESSAGE_TYPE>\n\" +
\" <MESSAGE_STANDARD>OAG<MESSAGE_STANDARD>\n\" +
\" <TRANSACTION_TYPE>PO<TRANSACTION_TYPE>\n\" +
\" <TRANSACTION_SUBTYPE>PROCESS<TRANSACTION_SUBTYPE>\n\" +
\" <DOCUMENT_NUMBER>123<DOCUMENT_NUMBER>\n\" +
\" <PARTY_SITE_ID>4444<PARTY_SITE_ID>\n\" +
\"<SOAHeader>\n\";
businessEvent.setStringProperty("WFBES_INPUT_header", headerPartMsg);
```

注意: この WFBES_INPUT_<partname>パラメータは、実行時におけるイベント呼出し時にのみ渡すことができ、イベント・サブスクリプションでは渡すことができないことに注意してください。Java コード用のインタフェース `oracle.apps.fnd.wf.bes.InvokerConstants` にはいくつかの定数が定義されています。

Web サービス入力メッセージ定義に複数のパートが含まれている場合は、SOAP 本文として送信されるパートの値がイベント・ペイロードとして渡されます。その他のパートの値はいずれもイベント・パラメータ(パラメータ名形式は **WFBES_INPUT_<partname>**)として渡されます。Web サービスを起動するために、特定の入力メッセージ・パートの値がオプションであっても、イベント・ペイロードのどのパートを SOAP 本文として設定する必要があるかを起動側サブスクリプションが認識できるように、null 値のパラメータを渡す必要があります。

たとえば、Web サービスの入力メッセージ・パート `myheader` がオプションであり、有効な値がなくても起動が成功する場合でも、次のように入力のイベント・パラメータを null 値に設定する必要があります。

```
businessEvent.setStringProperty("WFBES_INPUT_myheader", null);
```

WS セキュリティのサポート

Web サービス・セキュリティ(WS セキュリティ)は、Web サービスにセキュリティを適用するための手段を提供する通信プロトコルです。これには、メッセージ整合性や単一メッセージ認証を通じて保護品質を提供する SOAP メッセージング拡張が記述されています。また、SOAP メッセージにセキュリティ・トークンを添付して、セキュリティ機能を強化する方法についても記述されています。

サービス起動フレームワークでは、メッセージにセキュリティ・トークンを関連付けることで、Oracle E-Business Suite からの Web サービス要求とサービス起動を認証できるように、汎用メカニズムの中で WS セキュリティをサポートしています。

この目的を達成するために、サービス起動フレームワークでは、WS セキュリティのサポートに UsernameToken ベースのセキュリティを使用しています。次の各項では、UsernameToken ベースのセキュリティ、およびイベント・サブスクリプション・ユーザー・インタフェースを使用したセキュリティ構成について説明します。

- UsernameToken ベースのセキュリティ、ページ 12-21
- イベント・サブスクリプション・ユーザー・インタフェースを使用した Web サービス・セキュリティの構成、ページ 12-22
 - カスタマイズ・レベルを指定したセキュリティ・パスワードの構成、ページ 12-22
 - セキュリティ・ヘッダーの有効期限パラメータの指定、ページ 12-26

UsernameToken ベースのセキュリティ

このセキュリティ・メカニズムでは、Web サービスを起動するユーザーを認証するために、SOAP 要求の SOAP ヘッダーで Web サービス・プロバイダに送信されたユーザー名とオプションのパスワードを渡します。

ここで説明するユーザー名/パスワード情報は、Oracle E-Business Suite のユーザー名/パスワードの概念であることに注意してください。

起動対象の Web サービスでユーザー名/パスワードによる認証が行われる場合のために、サービス起動フレームワークは、Web サービス起動時における UsernameToken ベースの WS セキュリティ・ヘッダーの使用にも対応しています。

注意: Web サービスの起動を求める SOAP 要求には、ユーザー名とプレーン・テキスト・パスワードで構成されるセキュリティ・ヘッダーが含まれている必要があります。SOAP 要求の一部として実行時に受信したパスワードは、Oracle E-Business Suite に格納された暗号化パスワードに対して検証されます。検証後に、SOAP 要求からのプレーン・テキスト・パスワードは破棄されます。

ユーザー名はクリア・テキストです。**パスワード**は UsernameToken プロファイルの中で最も機密性の高い部分です。

サービス起動フレームワークでは、サービス起動時にユーザー名とオプション・パスワード (PasswordText タイプ)による UsernameToken ベースの WS セキュリティを適用できます。

たとえば、UsernameToken を使用した WS セキュリティ・ヘッダーは次のようになります。

```
<wsse:Security>
...
<wsse:UsernameToken wsu:Id="UsernameToken-1"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">
  <wsse:Username>myUser</wsse:Username>
  <wsse:Password
Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username- token-profile-1.0#PasswordText">password</wsse:Password>
  <wsse:Nonce
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soap-message-security-
1.0#Base64Binary">RDyVo/jbXJdSKuVEPrQW6Q==</wsse:Nonce>
  <wsu:Created>2013-09-02T04:56:48.597Z</wsu:Created>
</wsse:UsernameToken>
</wsse:Security>
```

- PasswordText パスワード・タイプは、クリア・テキストで記述されたパスワードです。もう 1 つのパスワード・タイプとして、UTF8 エンコードのパスワードを base64 でエンコードした SHA-1 ハッシュ値を扱う「PasswordDigest」がありますが、このタイプのパスワードはこのリリースではサポートされていません。
- <Nonce>: この要素はパスワードを識別する一意のランダム文字列です。リプレイ攻撃による UsernameToken セキュリティの再利用を防ぐのに役立ちます。
- <Created>: この要素はセキュリティの作成時間を示します。

イベント・サブスクリプション・ユーザー・インタフェースを使用した Web サービス・セキュリティの構成

サービス起動フレームワークでは、UsernameToken ベースのセキュリティを容易に保守できるように、設計時ユーザー・インタフェースを使用してセキュリティ・パスワードを構成できます。

「イベント・サブスクリプションの作成 - Web サービスの起動」ウィザードで Web サービスの詳細を入力すると、「Web サービス・セキュリティ」リージョンが表示され、ユーザー名およびパスワード情報を適宜指定または更新できるようになります。この情報は Vault に安全に格納されます。

カスタマイズ・レベルを指定したセキュリティ・パスワードの構成

Oracle Workflow では、ビジネス・イベントに対して様々なレベルの更新が許容されます。各イベントおよびサブスクリプションには、イベント定義の更新の可否を決めるカスタマイズ・レベルが割り当てられます。カスタマイズ・レベルを使用することで、Oracle E-Business Suite のシード・データを保護したり、アップグレード時に既存のカスタマイズを維持できます。

イベントおよびサブスクリプションのカスタマイズ・レベルは次のいずれかになります。

- コア - イベント定義およびサブスクリプション定義の変更が禁止されます。このレベルは、Oracle E-Business Suite によってシードされたイベントに対してのみ使用されます。
- 制限 - イベント・ステータスを「使用可能」または「使用不可」に更新することは可能ですが、イベント定義のその他の変更は禁止されます。このレベルは、Oracle E-Business Suite によってシードされたイベントに対してのみ使用されます。
- ユーザー - イベント定義およびサブスクリプション定義に含まれるすべてのプロパティの更新が可能です。このレベルは、独自に定義するイベントに対して自動的に設定されます。

インスタンス間のセキュリティ情報の構成

インスタンス間でのイベント・サブスクリプション定義の移動を考慮して Web サービス・セキュリティを構成する際には、セキュリティ情報の入力または更新の可否は、次に説明するカスタマイズ・レベルによって決まります。

- **カスタマイズ・レベル - 「ユーザー」**

カスタマイズ・レベルが「ユーザー」に設定された起動側サブスクリプションがターゲット環境で作成された場合は、その環境と定義がアップロードされた環境であれば定義全体の編集が可能です。

ある環境で WS セキュリティのユーザー名を作成し、別の環境に移動することは可能ですが、各ターゲット環境でパスワードを構成しないかぎり、そのユーザーを使用して Web サービスを起動することはできません。

- **カスタマイズ・レベル - 「制限」または「コア」**

カスタマイズ・レベルが「制限」または「コア」に設定された起動側サブスクリプションがターゲット環境にアップロードされた場合は、次のオプションがあります。

- ユーザー名が構成済である場合

Web サービスに対してユーザー名が構成済である場合、ワークフロー管理者は、Oracle Workflow ビジネス・イベント・マネージャおよび起動側サブスクリプションにアクセスしたり、ターゲット環境で当該ユーザーのパスワードを更新できます。そのためには、「ワークフロー管理者 Web(新規)」職責で Oracle E-Business Suite にログインする必要があります。ナビゲータから「ビジネス・イベント」を選択し、水平ナビゲーションで「サブスクリプション」を選択します。

起動側イベント・サブスクリプションを検索および特定し、パスワードを更新します。

ユーザー名は更新できません。

- ユーザー名が未定義である場合

ワークフロー管理者は、Workflow ビジネス・イベント・マネージャにアクセスして、ユーザー名とパスワードの両方を構成できます。

注意: システム側では、Vault にパスワードを格納するためのモジュールとキーの値がターゲット環境で導出されます。

- モジュール - Vault 用のモジュール名はビジネス・イベントから導出され、30 文字に制限されます。
- キー - Vault 用のキー値はビジネス・イベントとユーザー名から導出されます。

起動側サブスクリプションに対して次の情報がまだ存在しなければ、サブスクリプション定義の一部として内部的に格納されます。

- WFBES_SOAP_USERNAME=<entered username>
- WFBES_SOAP_PASSWORD_MOD=<derived module name>
- WFBES_SOAP_PASSWORD_KEY=<derived key name>

異なるサブスクリプション・カスタマイズ・レベルを指定した WS セキュリティの構成の例

シナリオ 1:

カスタマイズ・レベルを「ユーザー」とした新規ビジネス・イベントおよびサブスクリプションをソース環境で定義し、WS セキュリティを要求する Web サービスの起動時にユーザー名とパスワードの両方を手動で入力させます。前に定義したイベントおよびサブスクリプションをターゲット環境に移動し、必要に応じて WS セキュリティを構成します。

ソリューション:

このシナリオでは、「ユーザー名」フィールドと「パスワード」フィールドの両方をターゲット環境で編集できます。ユーザー名の値は自動的に移入され、パスワードの値は空白です。必要に応じて、新規ユーザー名(オプション)とそれに対応するパスワードを更新できます。

次のステップを使用して、ターゲット・インスタンスで WS セキュリティを構成します。

1. 次の各トピックで説明するすべてのステップを実行して、セキュリティ・ユーザー名およびパスワードを使用する新規イベントおよびサブスクリプションを定義します。
 - Web サービス起動側ビジネス・イベントの作成、ページ 12-6
 - 「Web サービスの起動」処理タイプを使用したローカル・サブスクリプションの作成、ページ 12-7
2. ワークフローXML ローダーを使用してイベントおよびサブスクリプションをダウンロードし、それらをターゲット環境にアップロードします。

注意: ワークフローXML ローダーは、ビジネス・イベント・システム・オブジェクトの XML 定義をデータベースとフラット・ファイルの間でアップロードおよびダウンロードできるコマンドライン・ユーティリティです。ワークフローXML ローダーを使用したイベントのダウンロードおよびアップロードについては、次のを参照してください。

- ビジネス・イベントの特定およびダウンロード、ページ 11-15
- データベースへの注釈付きファイルのアップロード、ページ 11-21

3. ソース・インスタンスで前に定義した起動側ビジネス・イベント

(`oracle.apps.xxx.user.webservice.invoke` など)を検索および特定し、結果表で「サブスクリプション」アイコンをクリックします。

4. サブスクリプションの「更新」アイコンをクリックします。カスタマイズ・レベルが「ユーザー」に設定されているため、すべてのフィールドを更新できます。「次へ」をクリックして「サブスクリプションの更新 - Web サービスの起動」ページの最後まで進みます。

「Web サービス・セキュリティ」リージョンで、「ユーザー名」と「パスワード」の両方のフィールドを編集できます。

- 「ユーザー名」フィールドの値(`weblogic` など)は、ステップ 1 でソース環境に定義したユーザー名に基づいて自動的に移入されます。
- パスワードの値は空白です。

必要に応じてユーザー名を更新し、Web サービスに対するパスワード情報を入力できます。「適用」をクリックします。

シナリオ 2:

すべての Oracle E-Business Suite 製品において、サービス起動に対するシード・イベントおよびサブスクリプションが用意されており、それぞれカスタマイズ・レベルが「制限」または「コア」に設定されています。WS セキュリティを要求する Web サービスの起動に必要なユーザー名は、製品固有のシード・イベントに対するサブスクリプションの作成時に構成されている場合とそうでない場合があります。

Oracle E-Business Suite リリース 12.2 のターゲット・インスタンスでシード・イベントおよびサブスクリプションを使用する際には、サービスの起動に使用するユーザー名(サブスクリプション所有者によってまだ提供されていない場合)とそのユーザーのパスワードを入力することによって、WS セキュリティを構成します。

ソリューション:

Oracle E-Business Suite リリース 12.2 のターゲット・インスタンスで、「ワークフロー・システム管理者」ロールを持つユーザー(`sysadmin` など)としてログオンします。ユーザー名がサブスクリプション所有者によってすでに提供されている場合、「ユーザー名」フィールドは更新できません。サービスの起動に使用するユーザーに対応するパスワードは常に入力できます。

次のステップを使用して、ターゲット・インスタンスで WS セキュリティを構成します。

1. Oracle E-Business Suite 12.2 のターゲット・インスタンスにログオンします。製品固有のシード・イベントを検索および特定し、結果表で「サブスクリプション」アイコンをクリックします。

- サブスクリプションの「更新」アイコンをクリックして、サブスクリプションの詳細をロードします。カスタマイズ・レベルが「制限」に設定されているため、「ステータス」フィールド以外はすべて無効化されています。

「次へ」をクリックして「サブスクリプションの更新 - Web サービスの起動」ページの最後まで進みます。「Web サービス・セキュリティ」リージョンで、次のセキュリティ情報を入力します。

- ユーザー名: シード・サブスクリプション定義にユーザー名情報が含まれていない場合は、その情報を入力します。

サブスクリプション所有者によってユーザー名がシード定義の一部として入力されている場合は、そのユーザー名の値がこのフィールドに表示され、編集できません。ユーザーはパスワードのみを入力する必要があります。

- パスワード: WS セキュリティ用のパスワード情報を入力します。
- パスワードの繰返し: 「パスワード」フィールドに入力したものと同一パスワードを入力します。

「適用」をクリックします。WS セキュリティを構成すると、Web サービスを起動できるようになります。

イベントに対するカスタマイズ・レベルの使用の詳細は、*Oracle Workflow 開発者ガイド*の「イベントのカスタマイズ・レベルおよびライセンス・ステータスのレビュー」および「ビジネス・イベントの管理」を参照してください。

セキュリティ・ヘッダーの有効期限パラメータの指定

起動側イベントに対するサブスクリプションを作成する際には、「Web サービス起動側パラメータ」リージョンに次のパラメータを追加して、セキュリティ・ヘッダーの有効期限を設定できます。そうすれば、リプレイによるヘッダーの再利用を防ぐのに役立ちます。

- WFBES_SOAP_EXPIRY_DURATION

デフォルトでは、ヘッダーの有効期限は、<wsu:Timestamp>要素で<wsu:Created>と<wsu:Expires>を使用して、作成後 60 秒に設定されています。

WFBES_SOAP_EXPIRY_DURATION パラメータで別の時間を指定した場合、その時間によってヘッダーのデフォルトの有効期限(60 秒)が上書きされます。

```
<wsse:Security soapenv:mustUnderstand="1"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">
  <wsu:Timestamp wsu:Id="Timestamp-2"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">
    <wsu:Created>2013-09-02T04:56:59.592Z</wsu:Created>
    <wsu:Expires>2013-09-02T04:57:59.592Z</wsu:Expires>
  </wsu:Timestamp>
  <wsse:UsernameToken>
    ...
  </wsse:UsernameToken>
</wsse:Security>
```

このリージョンに追加したその他のサブスクリプション・パラメータと同様に、別の有効期限がイベント・パラメータとして渡された場合は、イベント・パラメータによってサブスクリプション・パラメータが上書きされます。

Web サービス応答での Oracle E-Business Suite へのコールバック

サービス起動フレームワークでは、同期要求応答サービス操作をサポートするために、Web サービスによる出力または応答メッセージが存在する場合は、Oracle Workflow のコールバック・メカニズムを使用して、応答メッセージがビジネス・イベント・システムを介して Oracle E-Business Suite に返されます。

注意: 同期要求応答メッセージは、Web サービス操作における共通のメッセージ交換パターンです。この交換パターンでは、クライアントがサービス・プロバイダに質問をし、その回答を受け取ってから処理を続けることになります。この操作パターンの詳細は、「メッセージ・パターンについて」(ページ 12-3)を参照してください。

このコールバック機能に起動側イベントのイベント・キーを渡すと、それに基づいて、応答用のコールバック・イベントが指定のインバウンド・エージェント(コールバック・エージェント)にエンキューされます。また、ワークフロー・プロセスにおいて「呼出し」イベント・アクティビティを使用して Web サービスを起動し、「受入」イベント・アクティビティを使用して Web サービス応答を待つ場合は、起動側イベント・キーを起動側または待機ワークフロー・プロセスの項目キーと同じにして、コールバックの実行時に、待機ワークフロー・プロセスが WF_ENGINE.EVENT API によって正しく識別されるようにする必要があります。

コールバック・イベントとエージェントの両方を使用することによって、Web サービス起動を Oracle E-Business Suite 内の待機ワークフロー・プロセスまたはその他のモジュールと再結合できます。Web サービス起動では、次のコールバック・サブスクリプションまたはイベント・パラメータを使用します。

- WFBES_CALLBACK_EVENT

このパラメータでは、Web サービスの完了時に呼び出されるように、サービス出力メッセージをペイロードとした有効なビジネス・イベントを指定できます。

たとえば、次のようにできます。

```
WFBES_CALLBACK_EVENT=oracle.apps.wf.myservice.callback
```

- WFBES_CALLBACK_AGENT

このパラメータでは、サービス応答メッセージをペイロードとしたイベントのエンキュー先として、有効なビジネス・イベント・システム・エージェントを指定できます。

重要: このパラメータは、WFBES_CALLBACK_EVENT が null でない場合にのみ機能します。null の場合は、出力メッセージが失われ、コールバックは実行されません。

たとえば、次のように、Web サービス・メッセージのデフォルトのインバウンド・エージェント(またはその他のインバウンド・キュー)を指定できます。

```
WFBES_CALLBACK_AGENT=WF_WS_JMS_IN
```

注意: カスタム・エージェントを定義した場合は、カスタム・エージェント名をパラメータ値として指定することもできます。

Web サービスの出力メッセージは `WFBES_CALLBACK_AGENT` に指定するインバウンド・エージェントにエンキューされるため、コールバック/受入ビジネス・イベント・メッセージを処理するには、インバウンド・エージェントに対するワークフロー・エージェント・リスナーを設定する必要があります(未設定の場合)。

注意: 応答メッセージがコールバック・エージェントにエンキューされるときには、コールバック・イベントを関連 ID として使用できます。これにより、管理者は、コールバック・イベントを処理するための、コールバック・エージェントに対する専用エージェント・リスナーを容易に作成できます。

たとえば、サービス起動のコールバック・イベントが `oracle.apps.wf.myservice.callback` で、コールバック・エージェントが `WF_WS_JMS_IN` であるとします。サービスが正常に起動されてこのイベントが `WF_WS_JMS_IN` にエンキューされる時、イベント `oracle.apps.wf.myservice.callback` を `WF_WS_JMS_IN` で関連 ID として使用すると、該当するイベントを処理するためのエージェント・リスナーを容易に作成できます。

実行時に、解析および格納されたサブスクリプション・パラメータと同名のイベント・パラメータが渡された場合は、イベント・パラメータの値がサブスクリプション・パラメータよりも優先されます。たとえば、イベント・パラメータは次のように渡されます。

- `BusinessEvent.setStringProperty("WFBES_CALLBACK_EVENT", "oracle.apps.wf.myservice.callback");`
- `BusinessEvent.setStringProperty("WFBES_CALLBACK_AGENT", "WF_WS_JMS_IN");`

サービス起動時にコールバック機能を使用するには、受入イベントを作成し、そのイベントを購読する必要があります。参照: 受入イベントおよびイベント・サブスクリプションの作成(オプション)、ページ 12-14。

Web サービスの起動方法をよりよく理解できるように、「ワークフロー・プロセスからの Web サービス起動の例」(ページ 12-32)を参照してください

Web サービスの起動

Oracle Workflow ビジネス・イベント・システムは、PL/SQL と Java の両方のレイヤーからイベントを呼び出すことができるワークフロー・コンポーネントです。したがって、Oracle E-Business Suite からのサービス起動は、PL/SQL レイヤーまたは Java レイヤーから行うことができます。

PL/SQL からのサービス起動

1. アプリケーションにより、PL/SQL API `WF_EVENT.Raise` を使用してビジネス・イベントが呼び出されます。

イベント・データは、`WF_EVENT.Raise` API のコールの中でイベント・マネージャに渡すことができます。または、イベント・マネージャでは、サブスクリプションにイベント・データまたはペイロードが必要な場合に、イベントの生成機能呼び出すことで、イベント・データまたはメッセージ・ペイロードを取得できます。

注意: `WF_EVENT.Raise` API の詳細は、*Oracle Workflow API リファレンス* を参照してください。

2. Oracle Workflow ビジネス・イベント・システム(BES)により、イベントのサブスクリプションに Java ルール関数
`oracle.apps.fnd.wf.bes.WebServiceInvokerSubscription` が存在することが識別されます。
3. ビジネス・イベント・システムにより、イベント・メッセージが `WF_JAVA_DEFERRED` キューにエンキューされます。その後、Java 遅延エージェント・リスナーにより、サブスクリプションがデキューおよび実行され、その Java ルール関数によって Web サービスが起動されます。
4. コールバック・イベントおよびエージェントのパラメータが指定されている場合は、コールバック情報に基づいて、Web サービス応答が Oracle E-Business Suite に返されます。コンカレント・マネージャ(CM)層で実行される Java 遅延エージェント・リスナーのプロセスにより、Web サービスが起動されます。

Java からのサービス起動

1. Java アプリケーションにより、「OA フレームワーク」ページのコントローラ/AMImpl、またはコンカレント・マネージャ層で実行される Java コードのいずれかで、Java メソッド
`oracle.apps.fnd.wf.bes.BusinessEvent.raise` を使用して、ビジネス・イベントが呼び出されます。
2. Java でのイベント呼出し時にはサブスクリプションのシード Java ルール関数
`oracle.apps.fnd.wf.bes.WebServiceInvokerSubscription` が利用可能であるため、サブスクリプションのフェーズに応じて、このルール関数がインライン実行されるか、遅延実行されるかが決定されます。
 - フェーズが 100 以上の起動側サブスクリプションが作成された場合は、イベントは `WF_JAVA_DEFERRED` キューにエンキューされます。
 - フェーズが 100 未満の起動側サブスクリプションが作成された場合は、イベントはインラインで作業手配されます。

イベントが「OA フレームワーク」ページから呼び出された場合は、OACORE WebLogic Server 内で作業手配ロジックが実行されます。

注意: Web サービス起動側イベントが中間層の Java コードから呼び出され、起動側サブスクリプションがサブスクリプション・フェーズ 100 未満の同期型サブスクリプションである場合は、イベントが呼び出されるとすぐに Web サービスが

起動されます。

起動が成功した場合は、コール側アプリケーションから応答を読み取ることができます。`BusinessEvent.getResponseData()` メソッドを使用することによって、即座に応答を取得できます。

インラインでの Web サービスの起動時に問題が発生した場合は、`oracle.apps.fnd.wf.bes.BusinessEvent.raise` によって `oracle.apps.fnd.wf.bes.BusinessEventException` がスローされます。

`BusinessEventException` オブジェクトにより、根本原因となる例外が内部的に `linkedException` オブジェクト内に格納されます。例外の完全な詳細を確認するには、`BusinessEventException.getLinkedException()` から例外スタック・トレースを出力します。

イベントが Java コードから呼び出され、サブスクリプション・フェーズが 100 以上であるか、イベントが PL/SQL から呼び出された場合は、イベント・メッセージは `WF_JAVA_DEFERRED` キューにエンキューされます。Web サービスによる出力または応答メッセージが存在する場合に、Oracle E-Business Suite でその出力メッセージを受信するためには、コールバック・エージェントによるコールバック・イベントが必要です。

次のサンプル Java コードでは、同じセッション内で Web サービスの起動と応答の読取りを行うビジネス・イベントが呼び出されます。

```

package oracle.apps.fnd.wf.bes;
import java.sql.Connection;
import oracle.apps.fnd.common.AppsLog;
import oracle.apps.fnd.common.Log;
import oracle.apps.fnd.wf.bes.InvokerConstants;
import oracle.apps.fnd.wf.common.WorkflowContext;
public class InvokeWebService {
    static Log mLog;
    static WorkflowContext mCtx;
    public InvokeWebService() {
    }
    public static Connection getConnection(String dbcFile) {
        Connection conn = null;
        System.setProperty("dbcfiler", dbcFile);
        WorkflowContext mCtx = new WorkflowContext();

        mLog = mCtx.getLog();
        mLog.setLevel(Log.STATEMENT);
        ((AppsLog)mLog).reInitialize();
        mLog.setModule("%");

        return mCtx.getJDBCConnection();
    }
    public static void main(String[] args)
    {
        BusinessEvent event;
        Connection conn;
        conn = getConnection(args[0]);
        try {
            // Proxyt host and port requires to be set in Java options
            System.setProperty("http.proxyHost", args[1]);
            System.setProperty("http.proxyPort", args[2]);

            event = new BusinessEvent
                ("oracle.apps.wf.IrepService.invoke", "eventKey1");
            // Input XML message for Web Service
            String input = null;
            input =
"<IntegrationRepositoryService_GetInterfaceFunctionByName
xmlns:=\"http://xmlns.oracle.com/apps/fnd/rep/ws\"> \n"+
<fullName>SERVICEBEAN:/oracle/apps/fnd/rep/ws/IntegrationRepos
itoryService:getInterfaceFunctionByNameSERVICEBEAN:/oracle/apps/fnd/
rep/ws/IntegrationRepositoryService:
getInterfaceFunctionByName</fullName>\n"+
<IntegrationRepositoryService_GetInterfaceFunctionByName>;
            event.setData(input);

            String headerPartMsg = "<SOAHeader
xmlns:=\"http://xmlns.oracle.com/xdb/SYSTEM\" \" \" +
\"env:mustUnderstand=\"0\"
xmlns:env=\"http://schemas.xmlsoap.org/soap/envelope/\"> \n\" +
\" <MESSAGE_TYPE>XML<MESSAGE_TYPE>\n\" +
\" <MESSAGE_STANDARD>OAG<MESSAGE_STANDARD>\n\" +
\" <TRANSACTION_TYPE>PO<TRANSACTION_TYPE>\n\" +
\" <TRANSACTION_SUBTYPE>PROCESS<TRANSACTION_SUBTYPE>\n\" +
\" <PARTY_SITE_ID>>123<PARTY_SITE_ID>>\n\" +
\" <DOCUMENT_NUMBER>4444<DOCUMENT_NUMBER>\n\" +
\" <SOAHeader>\n\"
businessEvent.setStringProperty("WFBES_INPUT_header",
headerPartMsg);
            event.raise(conn);
            conn.commit();

            Object resp = event.getResponseData();
            if (resp != null) {

```

```

String respStr = resp.toString();
    // Process web service response here
    }
    else {
        // Either web service invocation failed and no exception was
        thrown
        // or the web service is one-way or asynchronous and
        did not return
        // a valid response
        System.out.println("No response received");
    }
}
catch (BusinessEventException e) {
    // Use appropriate logging mechanism as per your coding standards
    // instead of System.out.println
    System.out.println("Exception occurred " + e.
getLinkedException().getMessage());
    // Print the complete exception stack to log file for
    troubleshooting
    // Most importantly, if an exception occurred, do not proceed to
    process the
    // response
    e.getLinkedException().printStackTrace();
}
catch (Exception e) {
    // Use appropriate logging mechanism as per your coding standards
    // instead of System.out.println
    System.out.println("Exception occurred " + e.getMessage());
    // Print the complete exception stack to log file for
    troubleshooting
    // Most importantly, if an exception occurred, do not proceed to
    process the
    // response
    e.printStackTrace();
}
}
}

```

重要: サービス起動フレームワークを使用して Web サービスを起動する際には、JDBC 接続を渡す必要がある

oracle.apps.fnd.wf.bes.BusinessEvent.raise(Connection) メソッドを使用して起動側ビジネス・イベントが呼び出されます。

JDBC 接続を取得するには、必ず、現在のシナリオに使用できるアプリケーション・コンテキスト・オブジェクトを使用してください。たとえば、サービス起動を「OA フレームワーク」ページから行う場合は、OAPageContext オブジェクトから JDBC 接続を取得します。コンカレント・プログラムから行う場合は、CpContext オブジェクトから JDBC 接続を取得します。これらの状況で WorkflowContext を作成しないでください。そうしないと、重複するアプリケーション・コンテキストが不必要に作成されます。新規 WorkflowContext は、他の手段ではまだ JDBC 接続を使用できない場合にのみ作成してください。

ワークフロー・プロセスからの Web サービス起動の例

この例では、次のノードまたはアクティビティを含むワークフロー・プロセスを起動することによって Web サービスを起動します。

- Web サービスを起動するための起動側ビジネス・イベント。

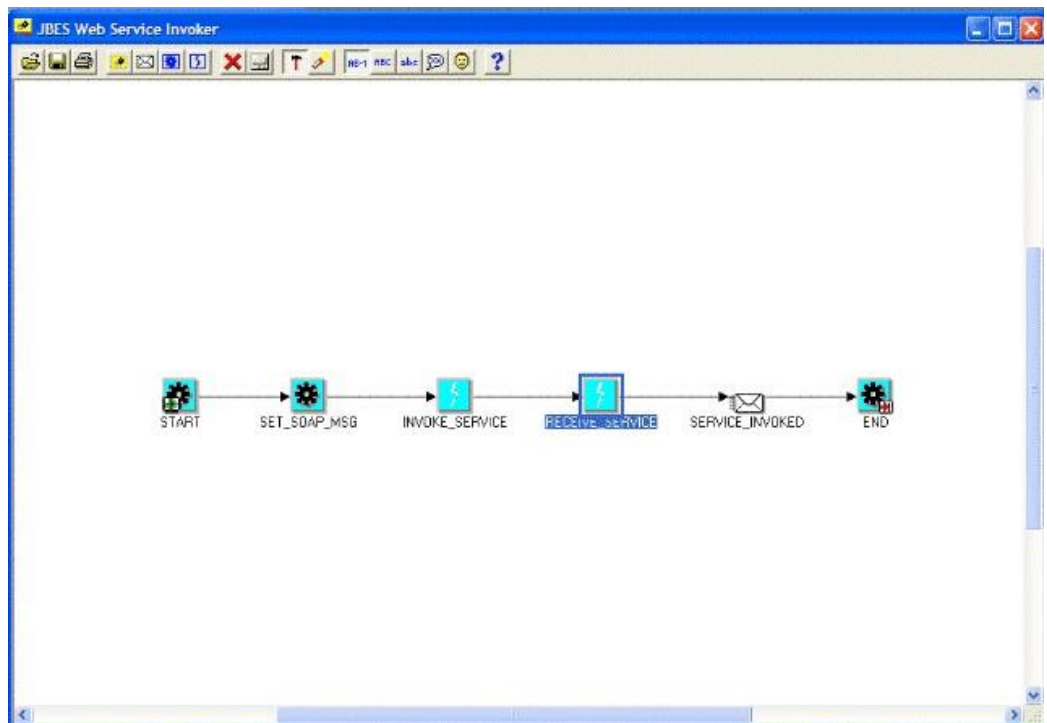
たとえば、`INVOKE_SERVICE` は、イベント処理「呼出し」を使用したイベント・アクティビティです。

- 応答または Web サービス出力メッセージを受信するための受入ビジネス・イベント。たとえば、`RECEIVE_SERVICE` は、イベント処理「受入」を使用したイベント・アクティビティです。
- プロセスの一環として XML メッセージの処理、Web サービス起動応答のユーザー通知、通常のトランザクション処理などに使用するその他のアクティビティ。

たとえば、`SERVICE_INVOKED` は、Web サービスが正常に起動されたときに通知メッセージを送信する通知アクティビティです。

次のワークフロー・プロセス・ダイアグラムは、サービス起動プロセス・フローを示しています。

Web サービス起動のワークフロー・プロセス・ダイアグラム



サービス起動メタデータの定義

コールバック機能を使用してサービス起動メタデータを定義するには、次に示すイベントおよびサブスクリプションを設定する必要があります。

1. ワークフロー・ダイアグラムに含まれる起動側イベント(`INVOKE_SERVICE` など)。

このアクティビティを使用して、イベントの XML ペイロードを SOAP 本文として渡したり、Web サービスの起動に必要なその他のイベント・パラメータを渡します。

参照: Web サービス起動側ビジネス・イベントの作成、ページ 12-6。

2. 起動側イベントに対するローカル・サブスクリプションおよびエラー・イベント・サブスクリプション。参照: 起動側イベントに対するローカル・サブスクリプションおよびエラー・イベント・サブスクリプションの作成、ページ 12-7。
3. 受入イベント(ワークフロー・ダイアグラムに含まれる RECEIVE_SERVICE など)とそのイベントに対する外部サブスクリプション。

重要: 受入イベントは、起動側イベントに使用するものと同じイベント・キーを使用して呼び出されます。待機ワークフロー・プロセスの項目キーと起動側イベントのイベント・キーが同じであることが重要です。

コールバック・イベントおよびエージェントのパラメータが設定されている場合は、このアクティビティを使用して、Web サービスの正常起動後に受入イベントが発生するのを待ちます。

参照: 受入イベントおよびイベント・サブスクリプションの作成(オプション)、ページ 12-14。

ワークフロー・エージェント・リスナーのステータスの確認

インバウンド・エージェントからの Web サービス応答メッセージを処理するには、そのエージェントに対するワークフロー・エージェント・リスナーが実行されているかどうかを確認する必要があります。

確認には次のステップを使用します。

1. ワークフロー管理者 Web アプリケーション職責を持つユーザーとして Oracle E-Business Suite にログインします。
2. ナビゲーション・メニューから、「Oracle Applications Manager」→「ワークフロー・マネージャ」を選択します。
3. 「エージェント・リスナー」ステータス・アイコンをクリックして、「サービス・コンポーネント」ページを開きます。
4. コールバック・エージェント・リスナーに使用するワークフロー・エージェント・リスナーを探します。たとえば、Web サービス応答メッセージの処理に使用するワークフロー・インバウンド JMS エージェント・リスナーを探し、そのリスナーが稼働中であることを確認します。

確認後、Oracle Workflow からワークフロー・プロセスを起動すると、Web サービスが起動し、コールバック応答が返されます。ワークフロー・プロセス・ダイアグラムに含まれる各アクティビティの進捗状況をレビューすることによって、プロセスを検証することもできます。

自動化されたワークフロー・プロセスに通知アクティビティが含まれている場合は、このワークフロー・プロセスから Web サービスが正常に起動されたときに、ワークフロー通知メッセージが届きます。

ワークフローの作成および起動方法の詳細は、*Oracle Workflow 開発者ガイド*を参照してください。

エラーの管理

ビジネス・イベント・サブスクリプションの実行時にエラーが発生した場合、サービス起動フレームワークでは、ビジネス・イベント・システムにおけるエラー処理と同様の方法でエラーが管理されます。サービス起動でフォルト・メッセージが返された場合は、エラー処理をトリガーするために、イベントがエラー・キューにエンキューされます。起動プロセスで発生した例外がサービスの使用不能に起因する場合は、サービス・フォルトをログに記録し、エラー・サブスクリプションを起動する必要があります。

エラー・キューにエンキューされるイベントの実行時例外を効果的に処理するために、サービス起動フレームワークでは、次のイベント・エラー・プロセスを通じて、サービス起動時に明確にエラー処理がトリガーされます。

- **DEFAULT_EVENT_ERROR2:** デフォルト・イベント・エラー・プロセス(再試行回数 1)

注意: DEFAULT_EVENT_ERROR2 エラー・ワークフロー・プロセスは WFERROR 項目タイプの下位に作成されます。

たとえば、ワークフローJava 遅延エージェント・リスナーにより Web サービス起動のイベント・サブスクリプションが実行されたときに実行時例外が発生した場合、イベントは WF_JAVA_ERROR キューにエンキューされます。イベントにエラー・ワークフロー・プロセス WFERROR:DEFAULT_EVENT_ERROR2 を起動するためのエラー・サブスクリプションが定義されている場合は、ワークフローJava エラー・エージェント・リスナーによりそのエラー・サブスクリプションが実行され、それを通じて、Web サービス定義、エラー詳細およびイベント詳細を含む通知が SYSADMIN に送信されます。Oracle Workflow のデフォルト・イベント・エラー・ハンドラには、報告されたエラーが修正されたことを確認してから、Web サービス起動プロセスを再試行できるオプションが用意されているため、SYSADMIN は、必要に応じて通知から再び Web サービスを起動できます。

ただし、同期サブスクリプション(フェーズ 100 未満)による起動側イベントの呼出しを通じて Web サービスを起動したときに実行時例外が発生した場合は、コール側アプリケーションに例外がスローされます。この場合、コール側アプリケーションによって例外が管理されます。

サービス起動時におけるエラー処理の有効化

サービス起動時におけるエラー処理機能を有効にするには、次の値を使用してエラー・サブスクリプションを作成する必要があります。

- 「エラー」ソース・タイプ
- 「ワークフローの起動」処理タイプ
- 「WFERROR:DEFAULT_EVENT_ERROR2」ワークフロー・プロセス

「イベント・サブスクリプションの作成」ページにアクセスするには、「ワークフロー管理者 Web アプリケーション」職責を持つユーザーとして Oracle E-Business Suite にログインします。ナビゲーション・メニューから「ビジネス・イベント」を選択し、「サブスクリプション」サブタブを選択します。

「イベント・サブスクリプション」ページで、「サブスクリプションの作成」をクリックします。

サービス起動に対するエラー・サブスクリプションの作成方法の詳細は、「「ワークフローの起動」処理タイプを使用したエラー・サブスクリプションの作成」(ページ 12-12)を参照してください。

Web サービス起動のテスト

サービス起動フレームワークでは、Oracle Workflow の「ビジネス・イベントのテスト」ページを使用して、Java または PL/SQL のいずれかのレイヤーからテスト・イベントを呼び出し、そのイベントに対する同期/非同期サブスクリプションを実行することによって、ビジネス・イベント・システムの基本操作をチェックします。このテスト機能を使用すると、コンカレント・マネージャ層や OACORE WebLogic Server から Web サービスを正常に起動できるかどうかを簡単に検証できます。

Web サービス起動のテストには次のいずれかの方法を使用できます。

- 「ビジネス・イベントのテスト」ページを使用してイベントを手動で呼び出す、ページ 12-36
- コマンドラインを使用してイベントを呼び出す、ページ 12-40

「ビジネス・イベントのテスト」ページの使用

「ビジネス・イベントのテスト」ページを使用すると、PL/SQL API または Java からイベントを呼び出してテストできます。

- 「Java で呼出し」オプションを使用して起動側イベントを呼び出す場合、サブスクリプション・フェーズが 100 未満であれば、OACORE WebLogic Server から Web サービスが起動されます。

Web サービスが正常に起動されると、「ビジネス・イベントのテスト」ページがリロードされ、「XML コンテンツ」フィールドの直後に XML 応答リージョンが表示されます。

同期サブスクリプションを使用して Web サービスを起動したときに実行時例外が発生した場合は、「ビジネス・イベントのテスト」ページに例外メッセージが表示されます。

- 「PLSQL で呼出し」オプションを使用して起動側イベントを呼び出す場合、コンカレント・マネージャ層から Web サービスが起動されます。呼び出したイベントは、WF_JAVA_DEFERRED にエンキューされてから、ワークフロー Java 遅延エージェント・リスナーによって作業手配されます。

シード Java ルール関数により、コールバック・イベントおよびエージェントを使用して、応答または Web サービス出力メッセージがビジネス・イベント・システムを介して Oracle E-Business Suite に返されます。

注意: サブスクリプションの作業手配とコンカレント・マネージャ層からの Web サービスの起動はワークフロー Java 遅延エージェント・リスナーによって行われるため、ワークフロー Java 遅延エージェント・リスナーが稼働中であることを確認してください。

検証するには、Oracle Applications Manager にログオンし、「ワークフロー・マネージャ」リンクを選択します。「エージェント・リスナー」を選択し、「ワークフ

ローJava 遅延エージェント・リスナー」を検索してそのステータスを確認します。

サービス起動のテスト

「ワークフロー管理者 Web(新規)」職責を持つユーザーとして Oracle E-Business Suite にログインした後、ナビゲーション・メニューから「ビジネス・イベント」を選択します。テスト対象のイベントを検索します。検索結果表で、呼出し対象のイベントの横にある「テスト」アイコンをクリックします。これにより、「ビジネス・イベントのテスト」ページが開き、一意のイベント・キーを使用してイベントを呼び出すことができます。起動側イベント・サブスクリプションのイベント・パラメータ、および入力メッセージ・スキーマに準拠した有効な XML メッセージを入力します。該当する場合は、応答 XML メッセージも「ビジネス・イベントのテスト」ページに表示されます。

入力したデータはすべて「ビジネス・イベントのテスト」ページに保持されることに注意してください。したがって、別のイベントを呼び出す必要がある場合は、「消去」をクリックして、入力済のデータをすべて消去する必要があります。

「ビジネス・イベントのテスト」ページからイベントを呼び出して Web サービスを起動する場合は、次のパラメータを指定できます。

- メッセージ変換: Web サービスの入力メッセージおよび出力メッセージに対する XSL 変換です。
 - WFBES_OUT_XSL_FILENAME
 - WFBES_IN_XSL_FILENAME
- 入力メッセージ・パートの値: アプリケーション・コンテキストを SOAP エンベロープに埋め込むために必要なパートの値を渡します。
 - WFBES_INPUT_<partname>

注意: WFBES_INPUT_<partname>パラメータは、実行時におけるイベント呼出し時にのみ渡すことができます。

- 実行時の SOAP サービス・アドレス: 実行時に渡すと、それによってサブスクリプション・パラメータ SERVICE_WSDL_URL から導出された SOAP サービス・エンドポイントが上書きされます。
 - SERVICE_SOAP_ADDRESS
- コールバック: Web サービス応答での Oracle E-Business Suite へのコールバックです。
 - WFBES_CALLBACK_EVENT
 - WFBES_CALLBACK_AGENT
- SOAP 本文:

- XML 入力メッセージ(必須)
- WS セキュリティ: UsernameToken ヘッダーを SOAP 要求に追加するために必要な情報です。

Web サービスのセキュリティ情報は、WSDL の起動ウィザードで必要な情報を入力した後に、イベント・サブスクリプション・ページの「Web サービス・セキュリティ」リージョンに入力します。参照: 「Web サービスの起動」処理タイプを使用したローカル・サブスクリプションの作成、ページ 12-7。

注意: ここで説明するように、セキュリティ情報はイベント・サブスクリプション・ユーザー・インタフェースから入力するように変更され、それにより、Oracle E-Business Suite リリース 12.1 で使用していたセキュリティ・パラメータが置き換えられています。

これらの WS セキュリティ・パラメータ(WFBES_SOAP_USERNAME、WFBES_SOAP_PASSWORD_MOD および WFBES_SOAP_PASSWORD_KEY)は、サービス起動フレームワークによって内部的に保持されるようになりました。

これらのパラメータの詳細は、次を参照してください。

- Web サービス入力メッセージ・パートについて、ページ 12-16
- WS セキュリティのサポート、ページ 12-21
- Web サービス応答での Oracle E-Business Suite へのコールバック、ページ 12-27

コールバックを要する起動のテスト

Oracle E-Business Suite へのコールバックを伴う起動をテストする場合は、次のパラメータおよび値を入力します。

- WFBES_CALLBACK_EVENT: 受入イベント
- WFBES_CALLBACK_AGENT: WF_WS_JMS_IN (またはその他のインバウンド・キューを値として指定)

「ビジネス・イベントのテスト」ページからテストする場合は、XML メッセージを事前に作成して「XML コンテンツ」フィールドに入力することから、入力 XML メッセージにエラーがあるときにエラー通知からそのエラーを修正してプロセスを再試行することはできません。

イベント起動をテストするには:

1. 「ワークフロー管理者 Web(新規)」職責を持つユーザーとして Oracle E-Business Suite にログインし、「ビジネス・イベント」を選択します。
2. テストを実行するビジネス・イベント
(oracle.apps.xxx.user.webservice.invoke など)を検索し、「進む」をクリックします。
3. 結果表から呼出し対象のビジネス・イベントを選択し、「テスト」アイコンをクリックして「ビジネス・イベントのテスト」ページを開きます。

4. 「イベント・キー」フィールドに一意のイベント・キーを入力し、「送信日」フィールドは空白のままにします。
5. パラメータの入力リージョンで、適切なパラメータを入力します。
6. 「イベント・データ」リージョンで、次の情報を入力します。

- アップロード・オプション: XML の記述
- XML コンテンツ: 入力メッセージとして適切な XML 情報を入力します。たとえば、次のように入力します。

```
<Process xmlns="http://xmlns.oracle.com/MyFirstSOAComposite_jws/UseMovieService/InvokeMovie">
  <zipCode>32822</zipCode>
  <radius>5</radius>
</Process>
```

7. 「Java で呼出し」をクリックして、OACORE WebLogic Server からイベントを呼び出します。

Web サービスが正常に起動されたら、イベント

(oracle.apps.xxx.user.webservice.invoke)が正しく呼び出されたことを示す確認メッセージが「ビジネス・イベントのテスト」ページの上部に表示されます。

このテスト・ページがリロードされ、「XML コンテンツ」フィールドの直後に XML 応答リージョンが表示されます。

8. 「PLSQL で呼出し」をクリックして、コンカレント・マネージャ層からイベントを呼び出します。
9. エラーが発生した場合は、「イベント・エラー詳細」リージョンが表示され、エラー詳細を確認できます。

「ビジネス・イベントのテスト」ページでのエラー詳細の確認

サービス起動のテスト中にエラーが発生した場合は、イベントの作業手配中にエラーが発生したことを示すエラー・メッセージが表示され、その詳細情報が「イベント・エラー詳細」リージョンに表示されます。

「イベント・エラー詳細」リージョンでは、同じページからエラー情報をすばやく確認でき、デバッグが容易になります。

オプションで、サービス起動で発生したことを順番に記録した詳細なログ・メッセージを表示するには、起動のテストを開始する前に、診断およびロギング機能を有効にしてテスト・ページの画面上に直接ログが表示されるようにします。ロギング機能を有効にする手順は、「OACORE WebLogic Server での起動失敗のトラブルシューティング」(ページ 12-41)を参照してください。

ビジネス・イベントのテストの詳細は、*Oracle Workflow 開発者ガイド*の「テスト・イベントを呼び出すには」を参照してください。

コマンドラインの使用

コマンドライン(API ベースのテスト方法)を使用して、PL/SQL または Java ベースのイベントを呼び出すこともできます。

- PL/SQL ベースのイベントの場合、PL/SQL `WF_EVENT.Raise` API を使用して、コンカレント・マネージャ層の JVM からの Web サービス起動をテストします。サブスクリプションの作業手配とコンカレント・マネージャ層からの Web サービスの起動はワークフローJava 遅延エージェント・リスナーによって行われます。

注意: サブスクリプションの作業手配とコンカレント・マネージャ層からの Web サービスの起動はワークフローJava 遅延エージェント・リスナーによって行われるため、ワークフローJava 遅延エージェント・リスナーが稼働中であることを確認してください。

検証するには、Oracle Applications Manager にログオンし、「ワークフロー・マネージャ」リンクを選択します。「エージェント・リスナー」を選択し、「ワークフローJava 遅延エージェント・リスナー」を検索してそのステータスを確認します。

- Java ベースの Web イベントの場合、Java メソッド `oracle.apps.fnd.wf.bes.BusinessEvent.raise` を使用して、Web サービス起動をテストします。

たとえば、クラスパスを `$AF_CLASSPATH` に設定したテスト・クラス

`oracle.apps.fnd.wf.bes.WFInvokerTestCase` を使用できます。

```
java oracle.apps.fnd.wf.bes.WSInvokerTestCase <DBC file> <proxy  
host> <proxy port>
```

Web サービス起動失敗のトラブルシューティング

Web サービスは次のいずれかの層から起動できます。

- **OACORE WebLogic Server:** 「OA フレームワーク」ページからの同期イベント・サブスクリプション(フェーズ 100 未満)を使用した Web サービス起動は、OACORE WebLogic Server 内から実行されます。
- **コンカレント・マネージャ(CM)層の JVM:** 次の Web サービス起動は、CM 層の JVM から実行されます。
 - ワークフロー・エージェント・リスナー・サービス内で実行される Java 遅延エージェント・リスナーによるもの:
 - PL/SQL からの同期または非同期イベント・サブスクリプションによる起動
 - Java/OA フレームワークからの同期イベント・サブスクリプションによる起動

- Java コンカレント・プログラムによるもの
 - Java コンカレント・プログラム内から直接実行される起動。
- **スタンドアロン JVM:** OACORE または CM 外で実行される Java プロセスからの同期イベント・サブスクリプションによる Web サービス起動は、その JVM 内から実行されます。

ほとんどの場合、Web サービスはファイアウォールの外側に存在し、実行ホストは、SOAP 要求の送信時に、WSDL または Web サービス・エンドポイントに直接アクセスすることはできません。Web サービスの起動元となる各層でプロキシ・パラメータを適切に設定および構成しないと、サブスクリプションの実行時に WSDL ファイルが解析および使用されず、Web サービスが正常に起動しなくなります。

各レイヤーでプロキシ・ホストおよびポートを適切に設定する方法の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「設定タスク」に記載の詳細情報を参照してください。

実行時に Web サービス起動が失敗した場合は、例外がスローされ、起動側イベントが WF_ERROR キューにエンキューされます。Web サービスは前述のレイヤーのいずれかから起動できるため、次に、Web サービスの起動元となるレイヤーごとに、起動失敗をトラブルシューティングして解決する方法を説明します。

- OACORE WebLogic Server での起動失敗のトラブルシューティング、ページ 12-41
- コンカレント・マネージャ(CM)層の JVM での起動失敗のトラブルシューティング、ページ 12-44
- スタンドアロン JVM での起動失敗のトラブルシューティング、ページ 12-45

OACORE WebLogic Server での起動失敗のトラブルシューティング

「OA フレームワーク」ページ内からの Web サービス起動のテスト実行全体を通してデバッグやトラブルシューティングが容易になるように、画面上ロギング・メカニズムを使用する必要があります。

画面上ロギングの有効化

画面上ロギング機能を有効にして、「ビジネス・イベントのテスト」ページの下部にログを直接表示できます。これらのログにより、コードを実行して Web サービスを起動する際の処理の詳細が示されます。

イベントの処理時およびサービスの起動時にフォルトまたは実行時例外が発生した場合は、その内容が画面上ロギングによってすばやく公開されます。

画面上ロギングを有効にするには、次の 2 つのステップを実行します。

1. 「FND: 診断」プロファイル・オプションの設定、ページ 12-42

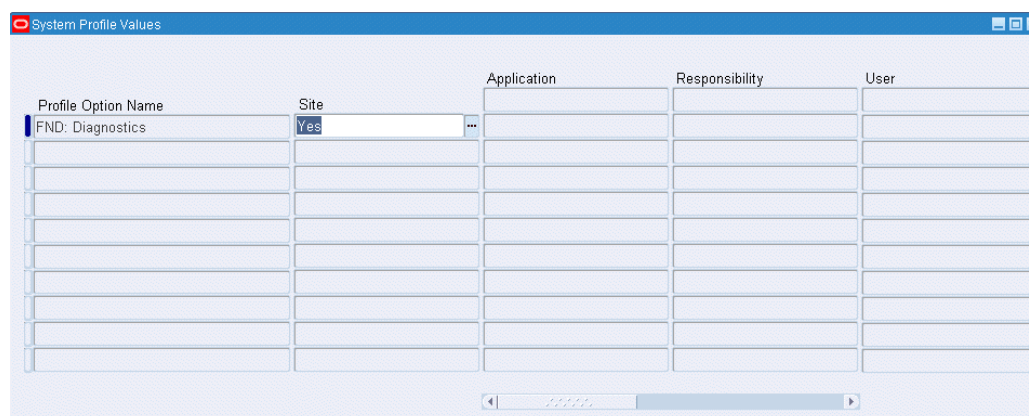
2. 画面上ログインの表示、ページ 12-42

「FND: 診断」プロファイル・オプションの設定

「ビジネス・イベントのテスト」ページを使用する前に、該当するレベルで「FND: 診断」プロファイル・オプションを「Yes」に設定して、HTML ベースのアプリケーション・ページのグローバル・メニューで「診断」リンクを有効にします。

注意: 「診断」リンクを使用すると、データベース・トレース、プロファイリングおよび画面上ログインを有効にすることができ、それにより、HTML ベースのアプリケーション・ページから実行されたトランザクションのトラブルシューティングが容易になります。

「FND: 診断」プロファイル・オプションの設定



The screenshot shows a window titled 'System Profile Values' with a table containing the following columns: Profile Option Name, Site, Application, Responsibility, and User. The first row is highlighted, showing 'FND: Diagnostics' in the Profile Option Name column and 'Yes' in the Site column. The other columns are empty.

Profile Option Name	Site	Application	Responsibility	User
FND: Diagnostics	Yes			

診断機能を使用して画面上ログインを有効にすると、「ビジネス・イベントのテスト」ページから起動側イベントを呼び出すことで Web サービスを起動したときに、WebServiceInvokeSubscription のログ・メッセージを追跡しやすくなります。

画面上ログインの表示

「FND: 診断」プロファイル・オプションを「Yes」に設定すると、HTML ページの右上隅に「診断」リンクが表示されます。

「診断」リンクを選択して適切な情報を入力すると、画面上ログイン機能が有効になります。目的のイベントを探してその起動をテストすると、関連するログ・メッセージがテスト・ページの下部に直接表示され、デバッグやトラブルシューティングが必要になったときに作業が容易になります。

重要: 「FND: 診断」プロファイル・オプションを「Yes」に設定しなければ、「診断」リンクはグローバル・メニューに表示されず、選択できません。参照: 「FND: 診断」プロファイル・オプションの設定、ページ 12-42。

「ビジネス・イベントのテスト」ページでのサービス起動のテスト時に画面上ログを表示するには:

1. 「ワークフロー管理者 Web(新規)」職責を持つユーザーとして Oracle E-Business Suite にログインします。「ビジネス・イベント」リンクを選択して、テストを実行する起動側ビジネス・イベント(oracle.apps.xxx.user.webservice.invoke など)を探し、「進む」をクリックして検索を実行します。
2. 検索結果表で、呼出し対象のビジネス・イベントを選択し、「テスト」アイコンをクリックして「ビジネス・イベントのテスト」ページを開きます。
3. ページの右上隅にある「診断」リンクをクリックします。
4. 次の情報を入力して、画面上ログを有効にします。
 - 診断: 画面上ログの表示
 - ログ・レベル: 文 (1)
 - モジュール: %
5. 「進む」をクリックします。画面上ロギングが有効になります。
6. 「ビジネス・イベントのテスト」ページにナビゲートし、イベントを呼び出して起動テストを実行します。

画面上ログ・メッセージのレビュー

画面上ロギング機能を有効にすると、テスト中に、関連するログ・メッセージが「ビジネス・イベントのテスト」ページの下部に表示されるようになります。これにより、コードによる Web サービス起動処理全体の詳細情報が示されます。

たとえば、同じページに表示される WebServiceInvokerSubscription ログ・メッセージをレビューして、サービス実行のステータス、例外やフォルト(発生した場合)およびコールバックの成否を確認できます。

次のログ例は、WFBES_CALLBACK_EVENT パラメータの値が受入イベントに設定され、WFBES_CALLBACK_AGENT パラメータの値が WF_WS_JMS_IN に設定されている場合に、コールバック応答メッセージが WF_WS_JMS_IN インバウンド・キューにエンキューされ、サービス実行が完了したことを示しています。

WebServiceInvokerSubscription のログ

```
5351]:PROCEDURE:[fnd.wf.bes.QueueHandlerInvoker]:enqueue() BEGIN
5351]:PROCEDURE:[fnd.wf.bes.PLSQLQueueHandler]:enqueue() :BEGIN (BusinessEvent{name=Receive event, key=002, priority=1, corr
5351]:EXCEPTION:[fnd.wf.bes.PLSQLQueueHandler]:prepareEnqueueStatement() : Successfully prepared enqueue statement. Call=(
5749]:EXCEPTION:[fnd.wf.bes.PLSQLQueueHandler]:enqueue() : Enqueued the following BusinessEvent -> BusinessEvent(name=Receive
5750]:EXCEPTION:[fnd.wf.bes.QueueHandlerInvoker]:enqueue() Enqueued the following BusinessEvent -> BusinessEvent(name=Receive
5750]:STATEMENT:[fnd.wf.bes.WebServiceInvokerSubscription]:performCallback():Enqueued response to WF_WS_JMS_IN
5750]:PROCEDURE:[fnd.wf.bes.WebServiceInvokerSubscription]:performCallback():END
5750]:PROCEDURE:[fnd.wf.bes.WebServiceInvokerSubscription]:postInvokeService():END
5750]:STATEMENT:[fnd.wf.bes.WebServiceInvokerSubscription]:onBusinessEvent():Service invocation complete
5750]:PROCEDURE:[fnd.wf.bes.WebServiceInvokerSubscription]:onBusinessEvent():END
```

ロギング機能を有効にする方法の詳細は、「画面上ロギングの有効化」(ページ 12-41)を参照してください。

コンカレント・マネージャ(CM)層の JVM での起動失敗のトラブルシューティング

コンカレント・マネージャ(CM)層の JVM での Web サービス起動失敗をトラブルシューティングするには、すべての Web サービス起動側イベントに対するエラー・サブスクリプションを作成し、ワークフローJava 遅延エージェント・リスナーからの起動実行時における例外の全詳細が取得されるようにする必要があります。

エラー・サブスクリプション

いずれの Web サービス起動側イベントに対しても、エラー・サブスクリプションを通じてビジネス・イベント・システムでのエラー処理を有効にし、サブスクリプションの実行時にエラー状態が SYSADMIN ユーザーに伝達されるようにする必要があります。Web サービスの定義、エラー詳細およびイベント詳細を含むワークフロー通知が SYSADMIN に送信されるため、SYSADMIN ユーザーは必要に応じてエラーを処理できます。

たとえば、起動時にエラーが発生し、イベントが WF_JAVA_ERROR キューにエンキューされ、そのイベントにエラー・ワークフロー・プロセス

WFERROR:DEFAULT_EVENT_ERROR2 を起動するためのエラー・サブスクリプションが定義されている場合は、ワークフローJava エラー・エージェント・リスナーによりそのエラー・サブスクリプションが実行され、それを通じて、Web サービス定義、エラー詳細およびイベント詳細を含む通知が SYSADMIN に送信されます。

詳細は、「エラーの管理」(ページ 12-35)を参照してください。

ワークフローJava 遅延エージェント・リスナーに対するログgingsの有効化

Oracle Workflow のデフォルト・イベント・エラー・ハンドラには、報告されたエラーが修正されたことを確認してから、Web サービス起動プロセスを再試行できるオプションが用意されているため、SYSADMIN は、必要に応じて通知から再び Web サービスを起動できます。さらに、例外を引き起こしたステップを詳細に分析する必要がある場合は、ワークフローJava 遅延エージェント・リスナー・ログgings・メカニズムを使用してワークフローJava 遅延エージェント・リスナーに対して文レベルのログを設定してから、失敗した Web サービス起動を再試行して、例外の原因となったステップの詳細を取得します。

詳細は、*Oracle Workflow 管理者ガイド*の「Java エージェント・リスナー」を参照してください。

Java コンカレント・プログラムに対するログgingsの有効化

Java コンカレント・プログラム内からサービス起動フレームワークを使用して Web サービスを起動する場合にログgingsを有効にするには、ビジネス・イベントの呼出しに使用される JDBC 接続の取得方法に基づき、適切なアプリケーション・コンテキストに対して次のプロファイル・オプション値を設定します。

- 「FND: デバッグ・ログ使用可能」(AFLOG_ENABLED)プロファイルの値を「Yes」に設定
- 「FND: デバッグ・ログ・レベル」(AFLOG_LEVEL)プロファイルの値を「文」に設定
- 「FND: デバッグ・ログ・モジュール」(AFLOG_MODULE)プロファイルの値を「%fnd.wf.bes%, %fnd.sif%」に設定

このプロファイルの値は、「fnd.wf.bes%, fnd.sif%」のように、カンマ区切りのモジュール名として設定でき、引用符を除く部分が 1 つの文字列として扱われます。すべ

てのモジュールに対してログを有効にする場合は、これを%に設定してください。

JDBC 接続を次に示すコンカレント・プログラム・コンテキスト自体から直接取得してから、Java コンカレント・プログラム要求が送信されたアプリケーション・コンテキスト(ユーザーおよび職責)を使用することをお勧めします。コンカレント・プログラム・コンテキストからの JDBC 接続を使用すると、該当するコンカレント・プログラム要求およびサービス起動フレームワークの実行に関するすべてのログをその要求 ID で簡単に分類できるようになります。コンカレント・プログラムの実行が完了したら、その要求 ID を使用して、コンポーネント「コンカレント・プログラム」のすべてのログ・メッセージを検索します。

```
Connection conn = cpContext.getJDBCConnection();  
event.raise(conn)
```

また一方で、非推奨のアプローチではあるものの、次のように個別の `WorkflowContext` を作成し、そこから JDBC 接続を取得することもできます。

```
WorkflowContext wfCtx = new WorkflowContext();  
Connection conn = wfCtx.getJDBCConnection();  
event.raise(conn)
```

ロギング・プロファイル・オプションの設定には、次のコンテキストを使用します。

- ユーザー: SYSADMIN
- 職責: システム管理者(SYSTEM_ADMINISTRATOR)

Java コンカレント・プログラムは、要求が送信されたアプリケーション・コンテキストで実行されますが、プログラム内で新規 `WorkflowContext` を作成した場合は、その JDBC 接続に対して SYSADMIN および SYSTEM_ADMINISTRATOR のアプリケーション・コンテキストがハードコードされます。したがって、このアプローチは推奨されていません。新規 `WorkflowContext` は、他の手段ではまだ JDBC 接続を使用できない場合にのみ作成してください。

2 つの異なるアプリケーション・コンテキスト(コンカレント・プログラム・コンテキスト用に作成済みのものと `WorkflowContext` 用に新規に作成したもの)が存在するため、コンカレント・プログラム要求およびサービス起動フレームワークの実行に関するログは、2 つのコンテキストに分けて格納されます。

コンカレント・プログラムの実行が完了した後で、モジュール `fnd.wf.bes%` および `fnd.sif%` のログ(サービス起動フレームワークの実行のみのログ)を取得します。また、要求 ID を使用して、コンカレント・プログラムの残りの実行に関するログを取得することもできます。

ログを取得するには、「システム管理者」職責で Oracle E-Business Suite にログオンし、ナビゲータから「**Oracle Applications Manager**」→「**ログ**」を選択して該当するログを検索します。

スタンドアロン JVM での起動失敗のトラブルシューティング

OACORE または CM 外で実行される Java プロセスから、`BusinessEvent.raise` メソッドを呼び出して「Web サービスの起動」同期サブスクリプションを通じて起動側イベントを呼び出すことで、Web サービスを起動する場合は、次のような状況が発生します。

- 起動が成功した場合は、メソッドによって応答メッセージが返されます。
- 実行時例外が発生した場合は、メソッドによって `BusinessEventException` がスローされます。これを使用して、完全なスタック・トレースを使用できます。

詳細は、「Web サービスの起動」(ページ 12-28)の「Java からのサービス起動」の項に記載のサンプル Java コードを参照してください。

シード Java ルール関数の拡張

Oracle E-Business Suite Integrated SOA Gateway では、開発者は、Java コーディング標準を使用して起動側サブスクリプションのシード・ルール関数

`oracle.apps.fnd.wf.bes.WebServiceInvokerSubscription` を拡張し、より専門的な処理を行うことができます。

開発者は、シード・ルール関数を拡張して、次のメソッドをオーバーライドできます。

- `preInvokeService`
- `postInvokeService`
- `addWSSecurityHeader`
- `setInputParts`
- `addCustomSOAPHeaders`

これらのメソッドの詳細は、*Oracle Workflow API リファレンス*を参照してください。次のステップを使用して、シード・ルール関数を拡張します。

1. `oracle.apps.fnd.wf.bes.WebServiceInvokerSubscription` を使用して、メソッドを拡張します。
2. コンパイルしたカスタム・クラス・ファイルを次の場所にアップロードします
`$JAVA_TOP/oracle/apps/fnd/wf/bes/`
3. `oacore` および `oafm` サーバーをバウンスします。
4. サブスクリプションを作成する際にはカスタム・ルール関数 `oracle.apps.fnd.wf.bes.xxxx` を使用します。
`xxxx` は拡張したカスタム・クラス名にすることに注意してください。たとえば、`oracle.apps.fnd.wf.bes.CustomWebServiceInvoker` のようにします。

`preInvokeService`

このメソッドは、Web サービス起動前の事前処理に使用します。

```
protected String preInvokeService(Subscription eo,
                                   BusinessEvent event,
                                   WorkflowContext context)
throws BusinessException;
```

Web サービス入力メッセージまたは要求メッセージは、`event.getData()` をコールすることによって取得します。これは、起動側イベントの呼出し時に渡されるか、ビジネス・イベントの生成機能によって生成されるビジネス・イベント・ペイロードです。

このメソッドでは、必要に応じて要求データに対して追加処理を実行できます。シード Java ルール関数によるデフォルト実装では、入力ペイロード・メッセージがある場合に、`WFBES_IN_XSL_FILENAME` に指定する XSL ファイルを使用して XSL 変換を実行します。

postInvokeService

```
protected void postInvokeService(Subscription eo,
                                   BusinessEvent event,
                                   WorkflowContext context,
                                   String requestData,
                                   String responseData)
throws BusinessException;
```

操作が同期要求応答である場合は、応答がパラメータ `responseData` で返されます。

このメソッドでは、必要に応じて応答に対して追加処理を実行し、アプリケーションの状態を更新できます。シード Java ルール関数によるデフォルト実装では、次のタスクを実行します。

- 応答または Web サービス出力メッセージに対する XSL 変換 - `WFBES_OUT_XSL_FILENAME` に基づく
- Workflow ビジネス・イベント・システムへのコールバック - `WFBES_CALLBACK_EVENT` および `WFBES_CALLBACK_AGENT` パラメータ値に基づく

addWSSecurityHeader

```
protected void addWSSecurityHeader(ArrayList headersList) throws
Exception;
```

このメソッドでは、WS セキュリティに準拠したヘッダーを SOAP 要求に追加します。シード Java ルール関数によるデフォルト実装では、イベント・パラメータ

`WFBES_SOAP_USERNAME`、`WFBES_SOAP_PASSWORD_MOD` および

`WFBES_SOAP_PASSWORD_KEY` に基づいて *UsernameToken* 要素をセキュリティ・ヘッダーに追加し、`WFBES_SOAP_EXPIRY_DURATION` パラメータに基づいて *Timestamp* 要素でそのヘッダーの有効期限を設定します。

このメソッドをオーバーライドして、任意の WS セキュリティ・ヘッダーを追加したり、カスタム・ロジックによってユーザー名とパスワードを取得して *UsernameToken* 要素を構築することもできます。ArrayList には整形形式 XML 要素を追加する必要があります。

次のコード・スニペットは、SOAP ヘッダーに追加された WS セキュリティを示しています。

```

try {
    DocumentBuilderFactory factory = DocumentBuilderFactory.
newInstance();
    factory.setNamespaceAware(true);
    DocumentBuilder bldr = factory.newDocumentBuilder();
    Document doc = bldr.newDocument();

    Element sec = doc.createElement("wsse:Security");
    Attr attr = doc.createAttribute("xmlns:wsse");
    attr.setValue("http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wsswssecurity-secext-1.0.xsd");
    sec.setAttributeNode(attr);
    doc.appendChild(sec);

    Element unt = doc.createElement("wsse:UsernameToken");
    sec.appendChild(unt);
    .... build XML message ....
}
catch (Exception e) {
}
headersList.add(doc.getDocumentElement());

```

setInputParts

```

protected void setInputParts(String[] partNames, Hashtable<String,
Element> partValues) throws Exception;

```

注意: 以前のリリースで使用していたメソッド

`setInputParts(WSIFMessage, Input, String)` throws `Exception`; は、このリリースではサポートされていません。このメソッドを実装する `WebServiceInvokerSubscription` のサブクラスがある場合は、ここで説明する新規メソッドを使用するように変更してください。開発者に必要な作業は、入力パート値に対して `org.w3c.dom.Element` オブジェクトを作成し、それを `Collections` オブジェクトに設定することだけです。

`WebServiceInvokerSubscription` のサブクラスには、操作のすべての入力パートの値を設定するための `setInputParts` メソッドをオプションで実装できます。このサブクラスは、Java ルール関数として「Web サービスの起動」イベント・サブスクリプションに対して使用されます。

このメソッドには、特定の起動インスタンスによって起動される操作の入力パート名のリストをパラメータ `partNames` で `java.lang.String` の配列として渡します。このメソッドを実装すると、パート名ごとに、タイプ `org.w3c.dom.Element` の自己完結型 XML 要素を `partValues` `java.util.Hashtable` パラメータにキーとして設定できるようになります。

イベント・パラメータ、イベント・ペイロードおよび `setInputParts` メソッドの拡張を組み合わせて、サービス起動の入力パート値を渡すことができます。たとえば、次の組合せの 1 つ以上を使用できます。

- `setInputParts` メソッドの実装からすべての入力パート値を設定し、イベント・ペイロードを `null` にできます。
- すべての入力パート値をビジネス・イベント・パラメータまたはペイロードとしてサービス起動フレームワークに渡すことができます。通常、イベント・ペイロードでは、`<soap:Body>` のように最大のパート値が伝送されます。

- 上の 2 つのアプローチを組み合わせます。
- 一部のパート値をイベント・パラメータまたはペイロードとして渡し、さらに setInputParts メソッドからも設定した場合は、setInputParts メソッドから設定した値が優先されます。

Web サービス操作に対するタイプ `org.w3c.dom.Element` の入力パート値は、次の 2 つの方法のいずれかで設定できます。

- パートごとに、XML 要素の値を名前 `WFBES_INPUT_<partname>` のイベント・パラメータとして渡します。
- 起動対象の Web サービス操作の各入力パートに対応する XML 要素を生成して設定するように、メソッド `setInputParts(String[], Hashtable (<String, Element>) throws Exception;` を拡張できます。

次のコード・スニペットは、このメソッドを使用して、ハードコードされた XML 要素文字列から `Element` オブジェクトを作成することによって入力パート `header` および `body` の値を設定する方法を示しています。

```

final protected void setInputParts(String[] partNames, Hashtable<String,
Element>
partValues) throws Exception {
    String METHOD_NAME = CLASS_PREFIX+"setInputParts(String[],
Hashtable<String,Element>";
    writeLog(METHOD_NAME, "BEGIN", Log.PROCEDURE);
    String value = "<SOAHeader xmlns:ns1=\"http://xmlns.oracle.
com/apps/fnd/soapprovider/plsql/fnd_user_pkg/\>
    <Responsibility>SYSTEM_ADMINISTRATOR<Responsibility>
    <RespApplication>SYSADMIN<RespApplication>
    <SecurityGroup><SecurityGroup>
    <NLSLanguage><NLSLanguage>
    <Org_Id><Org_Id>
<SOAHeader>";
    partValues.put("header", getDocumentElement(value));
    value = "<InputParameters
xmlns=\"http://xmlns.oracle.
com/apps/fnd/soapprovider/plsql/fnd_user_pkg/testusername/\>
><X_USER_NAME>SYSADMIN</X_USER_NAME></InputParameters>";
    partValues.put("body", getDocumentElement(value));
    value = "<GetTheatersAndMovies
xmlns=\"http://www.example.com/whatsshowing\"
><zipCode>32822</zipCode><radius>10</radius></GetTheatersAndMovies>";
    partValues.put("parameters", getDocumentElement(value));
    writeLog(METHOD_NAME, "END", Log.PROCEDURE);
}

/**
 * This function is called to convert String to XML Element
 * @param data
 * @return
 * @throws Exception
 */

public Element getDocumentElement(String data)
throws Exception {
    String METHOD_NAME = "getDocumentElement(String)";
    Element ret = null;
    writeLog(CLASS_PREFIX + METHOD_NAME, "BEGIN", Log.PROCEDURE);
    DOMParser parser = new DOMParser();
    parser.parse(new StringReader(data));
    Document doc = parser.getDocument();
    if(doc != null) {
        ret = doc.getDocumentElement();
    }
    writeLog(CLASS_PREFIX + METHOD_NAME, "END", Log.PROCEDURE);
    return ret;
}

```

addCustomSOAPHeaders

```

protected void addCustomSOAPHeaders(ArrayList<Element> customHeaders)
throws Exception;

```

注意: 以前のリリースで使用していたメソッド `addSOAPHeaders` は、このリリースではサポートされていません。このメソッドを実装する

`WebServiceInvokerSubscription` のサブクラスがある場合は、ここで説明する `addCustomSOAPHeaders` メソッドを使用するように変更してください。開発者に必要な作業は、カスタム SOAP ヘッダーに対して `org.w3c.dom.Element` オブジェクトを作成し、それを `Collections` オブジェクトに設定することだけです。

WebServiceInvokerSubscription のサブクラスには、SOAP 要求にカスタム SOAP ヘッダーを設定するための addCustomSOAPHeaders メソッドをオプションで実装できます。このサブクラスは、Java ルール関数として「Web サービスの起動」イベント・サブスクリプションに対して使用されます。

このメソッドを実装すると、タイプ org.w3c.dom.Element の自己完結型 XML 要素を任意の数だけ customHeadersjava.util.ArrayList パラメータに設定できるようになります。SOAP ヘッダーにはすべての XML 要素が追加されます。

このメソッドは、起動対象の WSDL 操作の入力メッセージに定義されていないカスタム SOAP ヘッダーを SOAP 要求に追加するのに役立ちます。WSDL 操作の入力メッセージに定義されている特定の入力パートに値を設定する場合は、setInputParts メソッドを使用します。

次のコード・スニペットは、このメソッドを使用して、SOAP 要求にカスタム SOAP ヘッダーを設定する方法を示しています。

```
final protected void addCustomSOAPHeaders(ArrayList customHeaders)
throws Exception {
    String custHdr = mEvent.getStringProperty("XXX_CUSTOM_HEADER");
    try {
        DocumentBuilderFactory factory = DocumentBuilderFactory.
newInstance();
        factory.setNamespaceAware(true);
        DocumentBuilder bldr = factory.newDocumentBuilder();
        Document doc = bldr.newDocument();
        doc = bldr.parse(new ByteArrayInputStream(custHdr.getBytes()));
        customHeaders.add((Element)doc.getFirstChild());
    }
    catch (Exception e) {
        throw e;
    }
}
```

サンプル・コード

次のコードは、addCustomSOAPHeaders メソッドを拡張して、WSDL に定義されていないヘッダー要素を SOAP ヘッダーに追加する方法を示しています。

```

final protected void addCustomSOAPHeaders(ArrayList customHeaders)
    throws Exception {
    String METHOD_NAME = CLASS_PREFIX+"addCustomSOAPHeaders
(ArrayList)";
    writeLog(CLASS_PREFIX + METHOD_NAME, "BEGIN", Log.PROCEDURE);

    // Add my own Custom header
    writeLog(METHOD_NAME, "Adding Custom header", Log.STATEMENT);
    addMyCustomHeader(customHeaders);

    // Add more headers if required to the ArrayList
    System.out.println("Adding Custom Headers in the sub-class");
    writeLog(METHOD_NAME, "END", Log.PROCEDURE);
}

private void addMyCustomHeader(ArrayList headersList)
    throws Exception {

    String METHOD_NAME = CLASS_PREFIX+"addMyCustomHeader
(ArrayList)";

    writeLog(METHOD_NAME, "BEGIN", Log.PROCEDURE);
    // Adding special SOAP Header to the request. This is required
only
    // if the WSDL's SOAP binding does not mandate the header but
it is
    // still required by the service. The XML element should be
self-sufficient
    // with all namespace declarations local to this element
    // In this case, the custom header is passed as an Event
Parameter and
    // set to the request to avoid hard-coding the header element
in code.
    // Any element can be passed at the time of raising the invoker
business event
    String custHdr = mEvent.getStringProperty("XXX_CUSTOM_HEADER");

    if (custHdr != null && !"".equals(custHdr)) {
        try {
            DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
            factory.setNamespaceAware(true);
            DocumentBuilder bldr = factory.newDocumentBuilder();
            Document doc = bldr.newDocument();

            doc = bldr.parse(new ByteArrayInputStream(custHdr.
getBytes()));

            // Add the element to the Headers list
            headersList.add((Element)doc.getFirstChild());
            writeLog(CLASS_PREFIX + METHOD_NAME, "Successfully
added custom header 1", Log.STATEMENT);
        }
        catch (Exception e) {
            throw new BusinessEventException("Exception when
creating header element - "+e.getMessage());
        }
    }

    String custHdr2 = mEvent.getStringProperty
("XXX_CUSTOM_HEADER2");

    if (custHdr2 != null && !"".equals(custHdr2)) {
        try {

```

```

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
    factory.setNamespaceAware(true);
    DocumentBuilder bldr = factory.newDocumentBuilder();
    Document doc = bldr.newDocument();

    doc = bldr.parse(new ByteArrayInputStream(custHdr2.
getBytes()));

    // Add the element to the Headers list
    headersList.add((Element)doc.getFirstChild());
    writeLog(CLASS_PREFIX + METHOD_NAME, "Successfully
added custom header 2", Log.STATEMENT);
    }
    catch (Exception e) {
        throw new BusinessException("Exception when
creating header element - "+e.getMessage());
    }
    }

    writeLog(METHOD_NAME, "END", Log.PROCEDURE);
}

```

その他の起動使用の考慮事項

Web サービスを起動するためのサービス起動フレームワークを実装する際には、いくつかの制限事項を考慮する必要があります。

- **WFBES_INPUT_<partname>**パラメータは、実行時におけるイベント呼出し時にのみ渡すことができます。
- サービス起動フレームワークでは、文書ベースの **Web** サービス起動のみがサポートされています。
- イベント・サブスクリプションの 1 対 1 の関係のサポート

Web サービスを正常に起動するには、各イベントにそれぞれ 1 つのみのサブスクリプション(「Web サービスの起動」処理タイプを使用)を関連付ける必要があります。イベント・サブスクリプションの 1 対 1 の関係は、特に同期要求応答のサービス起動で重要になります。

サービス起動フレームワークの実装時における考慮事項の詳細は、*Oracle E-Business Suite Integrated SOA Gateway インプリメンテーション・ガイド*の「実装に関する制限事項および考慮事項」を参照してください。

統合リポジトリ・アノテーションの標準

一般的なガイドライン

Oracle Integration Repository は、Oracle E-Business Suite 全体のアプリケーションで公開されている数多くの統合インタフェース・エンドポイントを含む、一元化されたリポジトリです。統合リポジトリの内容は、注釈付きソース・コード・ファイルを解析することによって設定されます。ソース・コード・ファイルは、統合リポジトリのメタデータの真のソースとなるため、規定の標準化された方法で注釈付けすることが非常に重要です。

この項では、使用するソース・コード・ファイルのタイプに関係なく、統合リポジトリ・アノテーションについて一般的に把握する必要がある内容について説明します。

アノテーションの構文

アノテーションは、アノテーション・タイプと、ゼロ以上のメンバーと値のペアを含むモディファイアです。メンバーと値の各ペアによって、値と、アノテーション・タイプの他のメンバーが関連付けられます。

アノテーションの構文は、Javadoc 構文に似ています。

```
@Namespace:TypeName keyString
```

```
@Namespace:TypeName freeString
```

```
@Namespace:TypeName keyString keyString keyString
```

```
@Namespace:TypeName keyString freeString
```

```
@Namespace:TypeName {inline annotation} {inline annotation}
```

要素の定義

Namespace は、使用しているアノテーションのグループを示します。大文字小文字が区別されます。現在使用しているアノテーションは、rep ネームスペースにあります。将来のアノテーションは、異なるネームスペースに導入される可能性があります。

TypeName は、アノテーション・タイプ名を示します。大文字小文字が区別されます。

製品チーム間で一貫性を確保するために、常に小文字のタイプ名を使用してください。

`keyString` は、アノテーションの後の最初の単語です。これはスペースを除外した文字列全体です。

`freeString` は、`keyString` の後の文字列です。これにはスペースやインライン・アノテーションを含めることができます。これは、次のアノテーションの先頭またはドキュメント・コメントの末尾で終了します。

形式要件

ソース・コード・ファイルでは、リポジトリ・アノテーションは Javadoc スタイルのコメントのブロックとして含められます。

次の一般的な手順を使用します。(Java での作業において、すでにファイルに強力な Javadoc コメントが存在する場合は、多くの場合、適切な `@rep:タグ` を追加することのみが必要になります。)

- 統合リポジトリに公開するインタフェースを選択します。*public*、*private* または *internal* と同時に、*active*、*obsolete*、*deprecated* または *planned* インタフェースに注釈付けできることに留意してください。

公開として注釈付けするインタフェースのみが、外部の統合リポジトリ UI に表示され、非公開と内部のインタフェースは、内部向けの Oracle UI に表示されます。したがって、以前に顧客マニュアルで公開としてドキュメント化されたすべてのインタフェースは、ソース・ファイルのアノテーションで公開として定義する必要があります。

- ソース・ファイルで、次の条件ルールに従って、アノテーション・ブロックの先頭を設定します。
 - Java の場合、スラッシュ、アスタリスク、アスタリスクの文字(`/*`)を挿入します。
 - Java 以外のファイルの場合、スラッシュ、アスタリスク、シャープの文字(`/*#`)を挿入します。
- テキスト説明を入力します。標準的な英語の完全な文を使用します。
- 該当する場合は、`@param` や `@return` などのプレーンな Javadoc タグを追加します。
- 次に、`@rep:scope` や `@rep:product` などの `@rep:タグ` を追加します。
- オプションで、`@rep:comment` のアノテーションを使用して、公開不可のコメントを追加します。(リマインダーやメモなどのために使用します。パーサーではこのアノテーションをスキップします。)
- アスタリスク、スラッシュ(`*/`)でアノテーション・ブロックを終了します。

次の例を参照してください。ソース・ファイルが PL/SQL またはその他の Java 以外のテクノロジーの場合、最初の行は、かわりにスラッシュ、アスタリスク、シャープの文字(`/*#`)にできます。

```

/**
 * This is the first sentence of a description of a sample
 * interface.This description can span multiple lines.
 * Be careful for public interfaces, where the description is
 * displayed externally in the Integration Repository UI.
 * It should be reviewed for content as well as spelling and
 * grammar errors.Additionally, the first sentence of
 * the description should be a concise summary of the
 * interface or method, as the repository UI will display
 * the first sentence by itself.
 *
 * @param <param name> <parameter description>
 * @rep:paraminfo {@rep:innertype <typeName>} {@rep:precision <value>}
 {@rep:required}
 * @rep:scope <public | internal | private>
 * @rep:product <product short code>
 * @rep:displayname Sample Interface
 */

```

アノテーションの構文チェッカと iLDT ジェネレータ

構文チェッカは、次のディレクトリにあります。

```
$IAS_ORACLE_HOME/perl/bin/perl $FND_TOP/bin/irep_parser.pl
```

チェッカの詳細は、-h フラグを使用して確認できます。

クラス・レベルとメソッド・レベル

ここでは、アノテーションの要件を分類することを目的として、緩やかな定義でクラスとメソッドという用語を使用します。インタフェースのアノテーションのコンテキストでは、PL/SQL パッケージはクラスと考えられ、PL/SQL ファンクションまたはプロシージャはメソッドと考えられます。一部のテクノロジーでは、クラス・レベルとメソッド・レベルで、異なるアノテーションの要件があります。詳細は、次の「必須」と「オプション」のアノテーションのリストを参照してください。

コンカレント・プログラムの考慮事項

コンカレント・プログラム(CP)の実装の基礎となるテクノロジーがインタフェース・タイプ(PL/SQL や Java CP など)でもある場合、何を注釈付けする必要があるかについて、混乱が生じる場合があります。

コンカレント・プログラムがリポジトリによって公開されるようにする場合は、コンカレント・プログラムを注釈付けする必要があります。基礎となるテクノロジーをリポジトリのコンカレント・プログラムから個別に公開する場合以外は、基礎となるテクノロジー(PL/SQL ファイルなど)を注釈付けしないでください。

この章では、アノテーションの標準について説明します。

- Java アノテーション、ページ A-4
- PL/SQL アノテーション、ページ A-17
- コンカレント・プログラム・アノテーション、ページ A-23
- XML ゲートウェイ・アノテーション、ページ A-25
- ビジネス・イベント・アノテーション、ページ A-35

- ビジネス・エンティティ・アノテーション、ページ A-41
- コンポジット・サービス - BPEL アノテーション、ページ A-115
- アノテーション用語集、ページ A-120

Java アノテーション

ユーザーは、クラスまたはメソッドの宣言の直前の Javadoc コメントにアノテーションを配置します。

必須のクラス・レベルのアノテーション

- 説明文で始める必要あり、ページ A-120
- rep:scope、ページ A-123
- rep:product、ページ A-124
- rep:implementation、ページ A-125

Java ビジネス・サービス・オブジェクトの場合にのみ必須で、プレーンな Java または SDO の場合は不要です。

- rep:displayname、ページ A-125
- rep:service、ページ A-142
- rep:servicedoc、ページ A-143

オプションのクラス・レベルのアノテーション

- link、ページ A-129
- see、ページ A-130
- rep:lifecycle、ページ A-127
- rep:ihelp、ページ A-131
- rep:category、ページ A-132

基礎となるすべてのメソッドが同じビジネス・エンティティを持つ場合にかぎり、クラス・レベルで BUSINESS_ENTITY を使用します。このような場合、メソッド・レベルでアノテーションを繰り返す必要はありません。

クラス・レベルの IREP_CLASS_SUBTYPE JAVA_BEAN_SERVICES を使用して、Java API がサービス可能であることを示します。詳細は、「Java Bean サービスに対するアノテーション」(ページ A-6)を参照してください。

クラス・レベルの IREP_CLASS_SUBTYPE AM_SERVICES を使用して、Java API のアプリケーション・モジュール・クラスがサービス可能であることを示します。「アプリケーション・モジュール・サービスに対するアノテーション」(ページ A-6)を参照してください。

- rep:compatibility、ページ A-128
- rep:standard、ページ A-135
- rep:metalink、ページ A-132
- rep:synchronicity、ページ A-144

必須のメソッド・レベルのアノテーション

- 説明文で始める必要あり、ページ A-120
- param、ページ A-138

適用可能であり、その他のタグ(@see や@rep:metalink など)にパラメータの説明が含まれていない場合にのみ使用します。

- return、ページ A-139 (該当する場合)
- rep:paraminfo、ページ A-139

REST サービスとしての Java API の場合は、パラメータ・レベルのアノテーション

@rep:paraminfo {@rep:required} {@rep: **key_param**}を使用します。参照: Java Bean サービスに対するアノテーション、ページ A-6 およびアプリケーション・モジュール・サービスに対するアノテーション、ページ A-6。

- rep:displayname、ページ A-125
- rep:businessevent、ページ A-141 (イベントが呼び出された場合)

オプションのメソッド・レベルのアノテーション

- link、ページ A-129
- see、ページ A-130
- rep:scope、ページ A-123
- rep:lifecycle、ページ A-127
- rep:compatibility、ページ A-128
- rep:category、ページ A-132

クラス・メソッドが異種ビジネス・エンティティを持つ場合にかぎり、メソッド・レベルで BUSINESS_ENTITY を使用します。

- rep:ihelp、ページ A-131
- rep:metalink、ページ A-132
- rep:appscontext、ページ A-145
- rep:primaryinstance、ページ A-146
- rep:httpverb、ページ A-136

このアノテーションを使用して、Java メソッドに適した HTTP 動詞を指定します。参照：
Java Bean サービスに対するアノテーション、ページ A-6 およびアプリケーション・モ
ジュール・サービスに対するアノテーション、ページ A-6。

- rep:synchronicity、ページ A-144

Java Bean サービスに対するアノテーション

統合リポジトリに登録されたすべての Java API を REST サービスとして公開できるとはかぎり
ません。シリアル化可能な Java Bean、または String や Int などの単純なデータ型の
Java API パラメータのみを Java Bean サービスとして公開できます。

既存の Java 固有のアノテーションに加えて、次のオプションのアノテーションを .Java ファイル
に追加して、REST サービスとしての Java API を注釈付けします。

- @rep:category IREP_CLASS_SUBTYPE JAVA_BEAN_SERVICES このクラス・
レベルのアノテーションは、Java API をサービス可能インタフェースとしてマークしま
す。 .Java ファイルにのみ適用可能です。

詳細は、A-132 ページの rep:category を参照してください。

- @rep:httpverb <comma separated list of HTTP VERBS ? GET,
POST>

このメソッド・レベルのアノテーションは、メソッドまたは操作に適した HTTP 動詞を明示
的に識別します。

詳細は、rep:httpverb (ページ A-136) を参照してください。

- @rep:paraminfo {@rep:required} {@rep:key_param}

このパラメータ・レベルのアノテーションは、パス変数をマークします。

@rep:key_param は、既存の @rep:paraminfo のアノテーションに追加されたイ
ンライン・アノテーションです。

詳細は、rep:paraminfo (ページ A-139) を参照してください。

アプリケーション・モジュール・サービスに対するアノテーション

Java Bean サービスと同様に、システム統合開発者は、アプリケーション・モジュール・サービ
スの .java ファイルであるアプリケーション・モジュール実装 Java クラスを注釈付けするた
めに、次のオプションのアノテーションを追加する必要があります。

- `@rep:category IREP_CLASS_SUBTYPE AM_SERVICES`

このクラス・レベルのアノテーションは、アプリケーション・モジュールをサービス可能インタフェースとしてマークします。`.Java` ファイルにのみ適用可能です。

詳細は、A-132 ページの `rep:category` を参照してください。

- `@rep:httpverb <comma separated list of HTTP VERBS ? GET, POST>`

`Java Bean` サービスで前述したように、このメソッド・レベルのアノテーションは、メソッドまたは操作に適した `HTTP` 動詞を明示的に識別します。

参照: `rep:httpverb`、ページ A-136。

- `@rep:paraminfo {@rep:required} {@rep:key_param}`

`Java Bean` サービスのアノテーションと同様に、このパラメータ・レベルのアノテーションは、パス変数をマークします。

参照: `rep:paraminfo`、ページ A-139。

システム統合開発者がアプリケーション・モジュール・サービスに対するアノテーションを完成させたら、統合リポジトリ・パーサー(`IREP` パーサー)を使用して、注釈付けされたインタフェース定義を検証する必要があります。検証中にエラーが発生しなければ、統合リポジトリ・ローダー・ファイル(`iLDT`)を生成できます。その後、統合リポジトリ管理者は、`FNDLOAD` を使用して統合リポジトリに `iLDT` ファイルをアップロードできます。

テンプレート

アプリケーション・モジュール・サービスを注釈付けするときは、次のテンプレートを使用できます。

Interface Template:

```
/**
 * < Interface description
 * ...
 * >
 *
 * @rep:scope <public>
 * @rep:product <product code>
 * @rep:displayname <Interface display name>
 * @rep:lifecycle <active|deprecated|obsolete|planned>
 * @rep:category IREP_CLASS_SUBTYPE AM_SERVICES
 * @rep:category BUSINESS_ENTITY <business_entity_code>
 * <sequenceNumber>
 */
```

Methods Template:

```
/**
 * < Method description
 * ...
 * >
 *
 * @param <paramName> < Parameter description
 * ...>
 * @rep:paraminfo {@rep:innertype <typeName>} {@rep:precision
 * <value>}
 * {@rep:required} {@rep:key_param}
 *
 *
 * @return < Parameter description
 * ...>
 * @rep:paraminfo {@rep:innertype <typeName>} {@rep:precision
 * <value>}
 * {@rep:required}
 *
 *
 * @rep:scope <public|private|internal>
 * @rep:displayname <Interface display name>
 * @rep:httpverb <GET|POST|GET,POST>
 * @rep:lifecycle <active|deprecated|obsolete|planned>
 * @rep:category BUSINESS_ENTITY <business_entity_code>
 * <sequenceNumber>
 */
```

Java Bean サービスを注釈付けするときは、次のテンプレートを使用できます。

Interface Template:

```
/**
 * < Interface description
 * ...
 * >
 *
 * @rep:scope <public>
 * @rep:displayname <Interface display name>
 * @rep:product <product code>
 * @rep:lifecycle <active|deprecated|obsolete|planned>
 * @rep:category BUSINESS_ENTITY <business_entity_code>
<sequenceNumber>
 * @rep:category IREP_CLASS_SUBTYPE JAVA_BEAN_SERVICES
 */
```

Methods Template:

```
/**
 * < Method description
 * ...
 * >
 *
 * @param <paramName> < Parameter description
 * ...>
 * @rep:paraminfo {@rep:innertype <typeName>} {@rep:precision <value>}
{@rep:required} {@rep:key_param}
 *
 *
 * @return < Parameter description
 * ...>
 * @rep:paraminfo {@rep:innertype <typeName>} {@rep:precision <value>}
{@rep:required}
 *
 *
 * @rep:scope <public|private|internal>
 * @rep:displayname <Interface display name>
 * @rep:httpverb <GET|POST|GET,POST>
 * @rep:lifecycle <active|deprecated|obsolete|planned>
 * @rep:category BUSINESS_ENTITY <business_entity_code>
<sequenceNumber>
 * @rep:businessevent <businessEventName>
 */
```

ビジネス・サービス・オブジェクトを注釈付けするときは、次のテンプレートを使用できます。

```

Interface Template:

/**
 * < Interface description
 * ...
 * >
 *
 * @rep:scope <public|private|internal>
 * @rep:displayname <Interface display name>
 * @rep:lifecycle <active|deprecated|obsolete|planned>
 * @rep:product <product code>
 * @rep:compatibility <S|N>
 * @rep:implementation <full implementation class name>
 * @rep:category <lookupType> <lookupCode> <sequenceNumber>
 */

Methods Template:

/**
 * < Method description
 * ...
 * >
 *
 * @param <paramName> < Parameter description
 * ...>
 * @rep:paraminfo {@rep:innertype <typeName>} {@rep:precision <value>}
{@rep:required}
 *
 * @return < Parameter description
 * ...>
 * @rep:paraminfo {@rep:innertype <typeName>} {@rep:precision <value>}
{@rep:required}
 *
 * @rep:scope <public|private|internal>
 * @rep:displayname <Interface display name>
 * @rep:lifecycle <active|deprecated|obsolete|planned>
 * @rep:compatibility <S|N>
 * @rep:category <lookupType> <lookupCode> <sequenceNumber>
 * @rep:businessevent <businessEventName>
 */

```

例

次に、注釈付けされたワークフロー・ワークリストのアプリケーション・モジュール・サービスの例を示します。

クラス・レベル:

```
/**
 * This is a Workflow Worklist Application Module Implementation class
 * which provides APIs to set preferred lists, get user worklist,
 * get lists and search the worklist based on certain filter criteria
 * like viewId, status, block size and block sequence.
 * @rep:scope public
 * @rep:product FND
 * @rep:displayname Workflow Worklist
 * @rep:category IREP_CLASS_SUBTYPE AM_SERVICES
 * @rep:category BUSINESS_ENTITY WF_WORKLIST
 */

public class WFWorklistServiceAMImpl extends OAApplicationModuleImpl {
    ...
}
```

メソッド・レベル:

```
/**
 * This is the method for getting worklist summary for a user
 * @param blockSize Block Size specifies the number of records to
be fetched, cannot be null
 * @paraminfo {@rep:required}
 * @param blockSequence Block Sequence, cannot be null, value >=1
 * @paraminfo {@rep:required}
 * @return Array of notifications
 * @rep:displayname Get HomePage Worklist
 * @rep:httpverb get, post
 * @rep:category BUSINESS_ENTITY WF_WORKLIST
 */
public Output[] getHomePGWorklist(String blockSize, String
blockSequence) throws Exception {
    ...
}
```

注意: パラメータ@param および@paraminfo のアノテーションは、メソッドまたはプロシージャの署名での定義と同じ順序にする必要があります。

次に、注釈付けされた従業員情報サービスの例を示します。

```

package oracle.apps.per.sample.service;

...
/**
 * A sample class to demonstrate how Java API can use the ISG REST
 * framework.This class provides
 * methods to retrieve list of direct reports, all reports of a
 * person.It also has methods to
 * retrieve personal details and accrual balance of a person.
 * @rep:scope public
 * @rep:product PER
 * @rep:displayname Employee Information
 * @rep:category IREP_CLASS_SUBTYPE JAVA_BEAN_SERVICES
 */
public class EmployeeInfo {

    public EmployeeInfo() {
        super();
    }

    /**
     * This method returns a list of direct reports of the requesting
     * user.
     *
     * @return List of person records who are direct reports
     * @rep:paraminfo {@rep:innertype oracle.apps.per.sample.beans.
     * PersonBean}
     * @rep:scope public
     * @rep:displayname Get Direct Reports
     * @rep:httpverb get
     * @rep:category BUSINESS_ENTITY sample
     */
    // Demonstration of list return type
    public List<PersonBean> getDirectReports() throws PerServiceException {

        ...
    }

    /**
     * This method returns an array of all reports of the requesting
     * user.
     *
     * @return Array of person records who are reporting into the
     * requesting user's organization hierarchy
     * @rep:scope public
     * @rep:displayname Get All Reports
     * @rep:httpverb get
     * @rep:category BUSINESS_ENTITY sample
     */
    // Demonstration of array return type
    public PersonBean[] getAllReports() throws PerServiceException {

        ...
    }

    /**
     * This method returns the person details for a specific person id.
     * Throws error if the person
     * is not in requesting user's org hierarchy.
     *
     * @return Details of a person in the logged on user's org hierarchy.
     * @param personId Person Identifier
     * @rep:paraminfo {@rep:required} {@rep:key_param}
     * @rep:scope public
     * @rep:displayname Get Person Details
     * @rep:httpverb get

```



```
* @rep:category BUSINESS_ENTITY sample
  */
  // Demonstration of simple navigation using path param
  public PersonBean getPersonInfo(int personId) throws
    PerServiceException {

    ...
  }
```

次に、注釈付けされた発注サービスの例を示します。

```

...
package oracle.apps.po.tutorial;

import oracle.jbo.domain.Number;

import oracle.svc.data.DataList;
import oracle.svc.data.DataService;
import oracle.svc.msg.MessageService;

import oracle.apps.fnd.common.VersionInfo;

/**
 * The Purchase Order service lets you to view, update, acknowledge and
 * approve purchase orders.It also lets you receive items, and obtain
 * pricing by line item.
 *
 * @see oracle.apps.fnd.framework.toolbox.tutorial.PurchaseOrderSDO
 * @see oracle.apps.fnd.framework.toolbox.tutorial.
 * PurchaseOrderAcknowledgementsSDO
 * @see oracle.apps.fnd.framework.toolbox.tutorial.
 * PurchaseOrderReceiptsSDO
 *
 * @rep:scope public
 * @rep:displayname Purchase Order Service
 * @rep:implementation oracle.apps.fnd.framework.toolbox.tutorial.
 * server.PurchaseOrderSAMImpl
 * @rep:product PO
 * @rep:category BUSINESS_ENTITY PO_PURCHASE_ORDER
 * @rep:service
 */
public interface PurchaseOrder extends DataService, MessageService
{
    public static final String RCS_ID="$Header$";
    public static final boolean RCS_ID_RECORDED =
VersionInfo.recordClassVersion(RCS_ID, "oracle.apps.fnd.
framework.toolbox.tutorial");

/**
 * Approves a purchase order.
 *
 * @param purchaseOrder purchase order unique identifier
 * @rep:paraminfo {@rep:required}
 *
 * @rep:scope public
 * @rep:displayname Approve Purchase Orders
 * @rep:businessevent oracle.apps.po.approve
 */
public void approvePurchaseOrder(Number poNumber);

/**
 * Acknowledges purchase orders, including whether the terms have
 * been accepted or not. You can also provide updated line
 * item pricing and shipment promise dates with the acknowledgement.
 *
 * @param purchaseOrders list of purchase order objects
 * @rep:paraminfo {@rep:innertype oracle.apps.fnd.framework.toolbox.
 * tutorial.PurchaseOrderAcknowledgementsSDO} {@rep:required}
 *
 * @rep:scope public
 * @rep:displayname Receive Purchase Order Items
 * @rep:businessevent oracle.apps.po.acknowledge
 */
public void acknowledgePurchaseOrders(DataList purchaseOrders);

/**
 * Receives purchase order items.For each given purchase order

```

```

* shipment, indicate the quantity to be received and, optionally,
* the receipt date if today's date is not an acceptable receipt
  date.
*
* @param purchaseOrders list of purchase order objects
* @rep:paraminfo {@rep:innertype oracle.apps.fnd.framework.toolbox.
tutorial.PurchaseOrderReceiptsSDO} {@required}
*
* @rep:scope public
* @rep:displayname Receive Purchase Order Items
* @rep:businessevent oracle.apps.po.receive_item
*/
public void receiveItems(DataList purchaseOrders);

/**
* Gets the price for a purchase order line item.
*
* @param poNumber purchase order unique identifier
* @rep:paraminfo {@required}
* @param lineNumber purchase order line unique identifier
* @rep:paraminfo {@required}
* @return the item price for the given purchase order line
*
* @rep:scope public
* @rep:displayname Get Purchase Order Line Item Price
*/
public Number getItemPrice(Number poNumber,
                           Number lineNumber);

```

次に、注釈付けされた発注の SDO データ・オブジェクトの例を示します。

```

package oracle.apps.po.tutorial;

import oracle.jbo.domain.Number;

import oracle.svc.data.DataObjectImpl;
import oracle.svc.data.DataList;

/**
 * The Purchase Order Data Object holds the purchase order data
 * including
 * nested data objects such as lines and shipments.
 *
 * @see oracle.apps.fnd.framework.toolbox.tutorial.PurchaseOrderLineSDO
 *
 * @rep:scope public
 * @rep:displayname Purchase Order Data Object
 * @rep:product PO
 * @rep:category BUSINESS_ENTITY PO_PURCHASE_ORDER
 * @rep:servicedoc
 */
public class PurchaseOrderSDO extends DataObjectImpl
{
    public PurchaseOrderSDO ()
    {
        super();
    }

    /**
     * Returns the purchase order header id.
     *
     * @return purchase order header id.
     */
    public Number getHeaderId()
    {
        return (Number)getAttribute("HeaderId");
    }

    /**
     * Sets the purchase order header id.
     *
     * @param value purchase order header id.
     * @rep:paraminfo {@rep:precision 5} {@rep:required}
     */
    public void setHeaderId(Number value)
    {
        setAttribute("HeaderId", value);
    }

    /**
     * Returns the purchase order name.
     *
     * @return purchase order name.
     * @rep:paraminfo {rep:precision 80}
     */
    public String getName()
    {
        return (String)getAttribute("Name");
    }

    /**
     * Sets the purchase order header name.
     *
     * @param value purchase order header name.
     * @rep:paraminfo {@rep:precision 80}
     */
    public void setName(String value)

```

```

{
    setAttribute("Name", value);
}

/**
 * Returns the purchase order description.
 *
 * @return purchase order description.
 * @rep:paraminfo {rep:precision 120}
 */
public String getDescription()
{
    return (String)getAttribute("Description");
}

/**
 * Sets the purchase order header description.
 *
 * @param value purchase order header description.
 * @rep:paraminfo {@rep:precision 80}
 */
public void setDescription(String value)
{
    setAttribute("Description", value);
}

/**
 * @return the purchase order lines DataList.
 * @rep:paraminfo {@rep:innertype
    oracle.apps.fnd.framework.toolbox.tutorial.PurchaseOrderLineSDO}
 */
public DataList getLines()
{
    return (DataList)getAttribute("Lines");
}

/**
 * @param list the putrchase order lines DataList.
 * @rep:paraminfo {@rep:innertype
    oracle.apps.fnd.framework.toolbox.tutorial.PurchaseOrderLineSDO}
 */
public void setLines(DataList list)
{
    setAttribute("Lines", list);
}
}

```

PL/SQL アノテーション

.pls ファイルと.pkh ファイルに注釈付けできます。

PL/SQL パッケージの場合、パッケージ仕様のみを注釈付けする必要があります。本文は注釈付けしないでください。

注釈付けを行う前に、/*#で始まるコメントが存在しないことを確認してください。スラッシュ、アスタリスク、シャープの文字はリポジトリ・アノテーションの開始を示すために使用するものとなり、通常のコメントで使用すると、エラーまたは望ましくない動作が発生します。

注釈付けするには、テキスト・エディタ(Emacs や vi など。)を使用して、ファイルを編集します。

パッケージごとに、CREATE OR REPLACE の行の直後からの 2 行目で、アノテーションを開始します。(CREATE OR REPLACE PACKAGE <package_name> AS の後の最初の行は、/* \$Header: \$ */ line にする必要があります。)

必須のクラス・レベルのアノテーション

- 説明文で始める必要あり、ページ A-120
- rep:scope、ページ A-123
- rep:product、ページ A-124
- rep:displayname、ページ A-125
- rep:category、ページ A-132

基礎となるすべてのメソッドが同じビジネス・エンティティを持つ場合にかぎり、クラス・レベルで BUSINESS_ENTITY を使用します。このような場合、メソッド・レベルでアノテーションを繰り返す必要はありません。

- rep:businessevent、ページ A-141 (イベントが呼び出された場合)

オプションのクラス・レベルのアノテーション

- link、ページ A-129
- see、ページ A-130
- rep:lifecycle、ページ A-127
- rep:compatibility、ページ A-128
- rep:ihelp、ページ A-131
- rep:metalink、ページ A-132

必須のメソッド・レベルのアノテーション

- 説明文で始める必要あり、ページ A-120
- param、ページ A-138

適用可能であり、その他のタグ(@see や@rep:metalink など)にパラメータの説明が含まれていない場合にのみ使用します。

- return、ページ A-139 (該当する場合)
- rep:displayname、ページ A-125
- rep:paraminfo、ページ A-139

- rep:businessevent、ページ A-141 (イベントが呼び出された場合)

オプションのメソッド・レベルのアノテーション

- link、ページ A-129
- see、ページ A-130
- rep:scope、ページ A-123
- rep:lifecycle、ページ A-127
- rep:compatibility、ページ A-128
- rep:category、ページ A-132

クラス・メソッドが異種ビジネス・エンティティを持つ場合にかぎり、メソッド・レベルで BUSINESS_ENTITY を使用します。

- rep:ihelp、ページ A-131
- rep:metalink、ページ A-132
- rep:appscontext、ページ A-145
- rep:primaryinstance、ページ A-146

テンプレート

PL/SQL ファイルを注釈付けするときは、次のテンプレートを使用できます。

注意: PL/SQL API のアノテーションは、/**または/*#で開始できます。

```

.
.
.
CREATE OR REPLACE PACKAGE <package name> AS
/* $Header: $ */
/**
 * <Put your long package description here
 * it can span multiple lines>
 * @rep:scope <scope>
 * @rep:product <product or pseudoproduct short code>
 * @rep:lifecycle <lifecycle>
 * @rep:displayname <display name>
 * @rep:compatibility <compatibility code>
 * @rep:businessevent <Business event name>
 * @rep:category BUSINESS_ENTITY <entity name>
 */

.
.
.

/**
 * <Put your long procedure description here
 * it can span multiple lines>
 * @param <param name 1> <param description 1>
 * @param <param name 2> <param description 2>
 * @rep:scope <scope>
 * @rep:product <product or pseudoproduct short code>
 * @rep:lifecycle <lifecycle>
 * @rep:displayname <display name>
 * @rep:compatibility <compatibility code>
 * @rep:businessevent <Business event name>
 */
PROCEDURE <procedure name> ( . . . );

.
.
.

/**
 * <Put your long function description here
 * it can span multiple lines>
 * @param <param name 1> <param description 1>
 * @param <param name 2> <param description 2>
 * @return <return description>
 * @rep:scope <scope>
 * @rep:product <product or pseudoproduct short code>
 * @rep:lifecycle <lifecycle>
 * @rep:displayname <display name>
 * @rep:compatibility <compatibility code>
 * @rep:businessevent <Business event name>
 */
FUNCTION <function name> ( . . . );

.
.
.

END <package name>;
/

commit;
exit;

```


例

参照用として、注釈付けされた PL/SQL ファイルの例を次に示します。

```

set verify off
whenever sqlerror exit failure rollback;
whenever oserror exit failure rollback;

create or replace package WF_ENGINE as

/*#
 * This is the public interface for the Workflow engine.  It allows
 * execution of various WF engine functions.
 * @rep:scope public
 * @rep:product WF
 * @rep:displayname Workflow Engine
 * @rep:lifecycle active
 * @rep:compatibility S
 * @rep:category BUSINESS_ENTITY WF_WORKFLOW_ENGINE
 */

g_nid number;          -- current notification id
g_text varchar2(2000); -- text information

--
-- AddItemAttr (PUBLIC)
-- Add a new unvalidated run-time item attribute.
-- IN:
--   itemtype - item type
--   itemkey - item key
--   aname - attribute name
--   text_value - add text value to it if provided.
--   number_value - add number value to it if provided.
--   date_value - add date value to it if provided.
-- NOTE:
--   The new attribute has no type associated. Get/set usages of the
--   attribute must insure type consistency.
--
/*#
 * Adds Item Attribute
 * @param itemtype item type
 * @param itemkey item key
 * @param aname attribute name
 * @param text_value add text value to it if provided.
 * @param number_value add number value to it if provided.
 * @param date_value add date value to it if provided.
 * @rep:scope public
 * @rep:lifecycle active
 * @rep:displayname Add Item Attribute
 */
procedure AddItemAttr(itemtype in varchar2,
                      itemkey in varchar2, aname in varchar2,
                      text_value in varchar2 default null,
                      number_value in number default null,
                      date_value in date default null);

--
-- AddItemAttrTextArray (PUBLIC)
-- Add an array of new unvalidated run-time item attributes of type
-- text.
-- IN:
--   itemtype - item type
--   itemkey - item key
--   aname - Array of Names
--   avalue - Array of New values for attribute

```

```
-- NOTE:
--   The new attributes have no type associated.      Get/set usages
of these
--   attributes must insure type consistency.
--

END WF_ENGINE;
/

commit;
exit;
```

コンカレント・プログラム・アノテーション

コンカレント・プログラムを注釈付けするには、「システム管理」職責を選択して、OA フレームワーク・ベースの「コンカレント・プログラムの定義」ページ上をクリックします。コンカレント・プログラムを問い合せて、「アノテーション」フィールドに移動します。そこにアノテーションを入力し、コミットして作業内容を保存します。

注釈付けとコミットの後で、FNDLOAD を使用して、コンカレント・プログラムの LDT を再作成する必要があります。

```
$FND_TOP/bin/FNDLOAD <db_connect> 0 Y DOWNLOAD
$FND_TOP/patch/115/import/afcpprog.lct <ldt_file_name>.ldt PROGRAM
APPLICATION_SHORT_NAME="<application_short_name>"
CONCURRENT_PROGRAM_NAME="<cp_short_name>"
```

必須のクラス・レベルのアノテーション

- 説明文で始める必要あり、ページ A-120

アノテーションは、LDT 内のコンカレント・プログラム独自の定義よりも優先されます。いずれか一方が存在する必要があり、そうでない場合は、インタフェースの生成が失敗します。

- rep:scope、ページ A-123
- rep:product、ページ A-124
- rep:displayname、ページ A-125

アノテーションは、LDT 内のコンカレント・プログラム独自の定義よりも優先されます。いずれか一方が存在する必要があり、そうでない場合は、インタフェースの生成が失敗します。

- rep:category、ページ A-132
- rep:businessevent、ページ A-141 (イベントが呼び出された場合)

注意: コンカレント・プログラムには、必須のメソッド・レベルのアノテーションはありません。

オプションのクラス・レベルのアノテーション

- link、ページ A-129
- see、ページ A-130
- rep:lifecycle、ページ A-127
- rep:compatibility、ページ A-128
- rep:ihelp、ページ A-131
- rep:metalink、ページ A-132
- rep:usestable、ページ A-134
- rep:usesmap、ページ A-146

注意: コンカレント・プログラムには、オプションのメソッド・レベルのアノテーションはありません。

テンプレート

コンカレント・プログラムを注釈付けするときは、次のテンプレートを使用できます。

```
/**
 * <Put your long description here
 * it can span multiple lines>
 * @rep:scope <scope>
 * @rep:product <product or pseudoproduct short code>
 * @rep:lifecycle <lifecycle>
 * @rep:category OPEN_INTERFACE <open interface name> <sequence_num>
 * @rep:usestable <table or view name> <sequence_num> <direction>
 * @rep:category BUSINESS_ENTITY <BO type>
 * @rep:category <other category> <other value>
 * @rep:businessevent <name of business event>
 */
```

注意: コンカレント・プログラムのアノテーションは、/**または/*#で開始できます。

例

参照用として、注釈付けされたコンカレント・プログラムの例を次に示します。

```

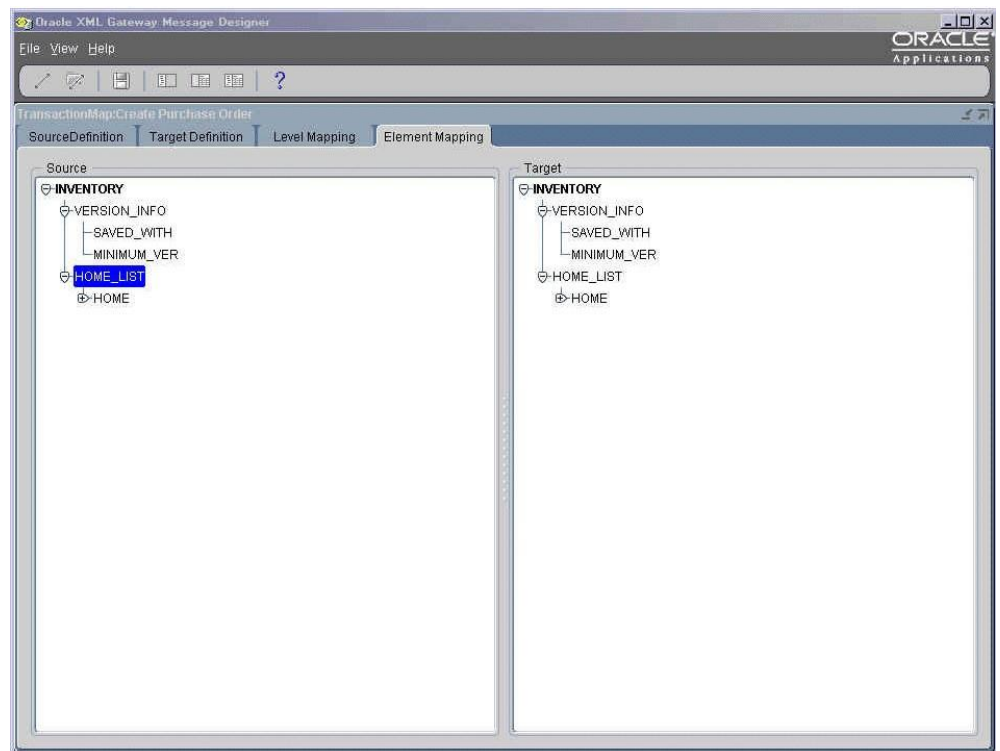
/**
 * Executes the Open Interface for Accounts Payable Invoices. It uses
 * the
 * following tables: AP_INVOICES_INTERFACE,
AP_INVOICE_LINES_INTERFACE.
 * @rep:scope public
 * @rep:product AP
 * @rep:lifecycle active
 * @rep:category OPEN_INTERFACES AP_INVOICES_INTERFACE 1
 * @rep:usestable AP_INVOICES_INTERFACE 2 IN
 * @rep:usestable AP_INVOICE_LINES_INTERFACE 3 IN
 * @rep:category BUSINESS_ENTITY AP_INVOICE
 */

```

XML ゲートウェイ・アノテーション

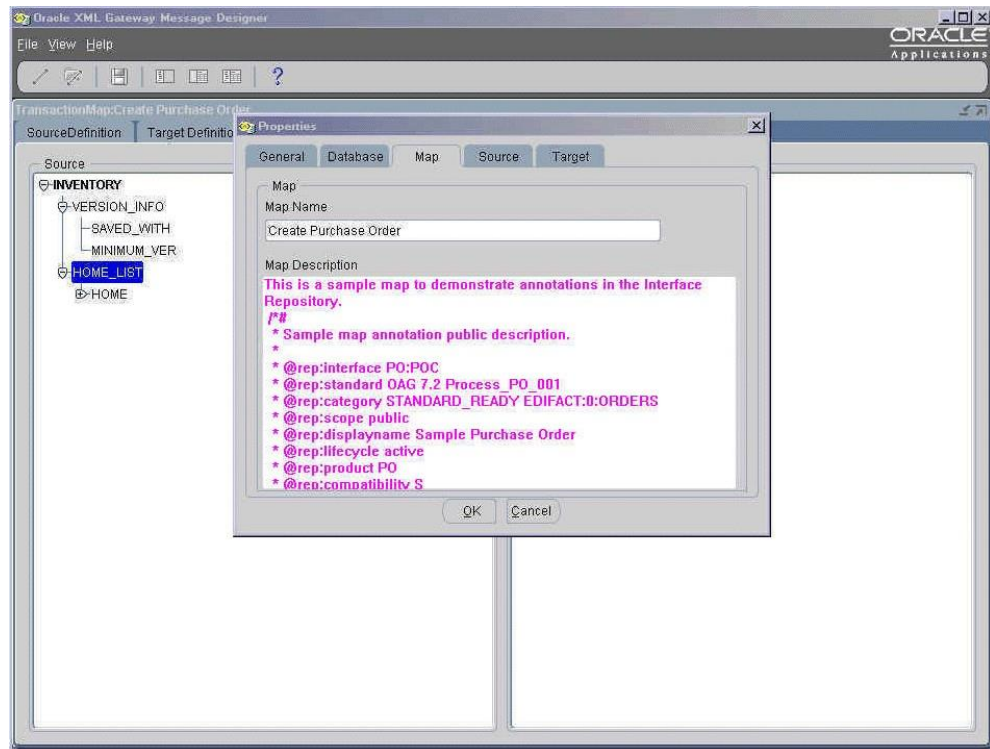
次の手順を使用して、取引情報の XML ゲートウェイ・マップを注釈付けします。

1. ソース・コードから既存のマップをチェックアウトして、Message Designer で開きます。



2. この特定のマップと関連付けられる内部取引タイプ、サブタイプ、標準および方向を確認します。統合リポジトリにロードする XML ゲートウェイに、このエントリが存在する必要があります。

Message Designer ファイル...→「プロパティ」をクリックして、「マップ」タブを選択します。マップの説明フィールドを使用して、既存の説明の後でマップを注釈付けします。
 @rep:interface のアノテーションと<Internal Transaction Type>:<Subtype>を入力して、それに応じて@rep:standard と @rep:direction を入力してください。



(オプション)このマップが、OAG などの特定の標準を完全にサポートするように設計されている場合、@rep:standard をその標準、バージョンおよび仕様名に設定します。ただし、マップが、(標準に対して準備ができていないなど)追加のカスタム変換によって標準をサポートすることを意図して設計されている場合、これを指定するために、rep:category_STANDARD_READY (ページ A-132)のアノテーションを使用します。

- 特定の内部取引タイプと内部取引サブタイプには、(パーティ・タイプに関係なく)特定の標準と方向について、製品チームによってシードされた 1 つのマップのみが必要です。アノテーションに同じタイプを含む追加のマップは、却下されて、エラーとして処理されます。外部取引タイプと外部取引サブタイプに基づく異なるマップが存在する場合がありますが、これらは取引先固有となるため、リポジトリには入力しません。将来のリリースで、XML ゲートウェイ内でこれらのルールがネイティブに強制されます。
- 単一のマップが複数の内部取引タイプと内部取引サブタイプで再使用される場合は、複数のアノテーションのそれぞれを、それ自体のコメント・ブロック内(つまり/*# ... */内)に入力できます。パーサーによって、各アノテーション・セットのエントリが統合リポジトリに作成されます。この機能はサポートされていますが、変更の可能性のあるインターフェースに対応するために、2 つの異なるマップを使用することをお勧めします。次のマップの再使用の例を参照してください。

内部取引 タイプ	内部取引 サブタイプ	D	外部取引 タイプ	外部取引 サブタイプ	STD	パーティ タイプ
AR	請求書	O	請求書	プロセス	OAG	C
AR	貸方	O	請求書	プロセス	OAG	C
AR	借方	O	請求書	プロセス	OAG	C

このシナリオでは、外部の表現が変更されないため、同じマップを再使用できます。ただし、内部処理と承認の考慮事項は、内部取引タイプおよび内部取引サブタイプに基づいて異なる場合があります。この場合、マップには、AR-Invoice、AR-Credit および AR-Debit など、内部取引タイプおよび内部取引サブタイプごとに 1 つの、3 つのアノテーション・ブロックを持つことができます。

- 通常、パラメータは、アウトバウンド・データを生成するための問合せで使用されるキーを指定するために、アウトバウンド・マップで使用されます。インバウンド・マップにはパラメータがありません。

3. 注釈付けされたマップを保存して、ソース管理にチェックインして、通常どおりパッチとしてリリースします。アノテーションは、統合リポジトリ・ローダーの一部として更新されます。

必須のクラス・レベルのアノテーション

- 説明文で始める必要あり、ページ A-120
- rep:scope、ページ A-123
- rep:product、ページ A-124
- rep:displayname、ページ A-125
- rep:category、ページ A-132
- rep:standard、ページ A-135
- rep:interface、ページ A-137

- rep:businessevent、ページ A-141 (イベントが呼び出された場合)
- rep:direction、ページ A-141

注意: XML ゲートウェイには、必須のメソッド・レベルのアノテーションはありません。

オプションのクラス・レベルのアノテーション

- link、ページ A-129
- see、ページ A-130
- param、ページ A-138

適用可能であり、その他のタグ(@see や@rep:metalink など)にパラメータの説明が含まれていない場合にのみ使用します。

- rep:paraminfo、ページ A-139
- rep:lifecycle、ページ A-127
- rep:compatibility、ページ A-128
- rep:ihelp、ページ A-131
- rep:metalink、ページ A-132
- rep:synchronicity、ページ A-144

注意: XML ゲートウェイには、オプションのメソッド・レベルのアノテーションはありません。

テンプレート

XML ゲートウェイを注釈付けするときは、次のテンプレートを使用できます。

Sample Inbound Map Annotation

```
/*#
 *Sample map annotation public description.
 *
 *@rep:interface <transaction_type:sub_type>
 *@rep:standard <OAG|cXML> <7.2|7.3> <specname>
 *@rep:direction IN
 *@rep:scope <public|private|internal>
 *@rep:displayname <Interface display name>
 *@rep:lifecycle <active|deprecated|obsolete|planned>
 *@rep:product <product code>
 *@rep:compatibility <S|N>
 *@rep:category <lookupType> <lookupCode> <sequenceNumber>
 *@rep:category STANDARD_READY <standard:version:specification>
 *@rep:businessevent <businessEventName>
 */
```

Sample Outbound Map Annotation

```
/*#
 *Sample map annotation public description.
 *
 *@param <paramName> <Parameter description>
 *@rep:paraminfo {@rep:required}
 *
 *@rep:interface <transaction_type:sub_type>
 *@rep:standard <OAG|cXML> <7.2|7.3> <specname>
 *@rep:direction OUT
 *@rep:scope <public|private|internal>
 *@rep:displayname <Interface display name>
 *@rep:lifecycle <active|deprecated|obsolete|planned>
 *@rep:product <product code>
 *@rep:compatibility <S|N>
 *@rep:category <lookupType> <lookupCode> <sequenceNumber>
 *@rep:category STANDARD_READY <standard:version:specification>
 *@rep:businessevent <businessEventName>
 */
```

Important Note

A given map should be unique to a given Internal Transaction Type / Subtype, Standard and Direction. This is because the External Transaction Type / Subtype are meant for Trading Partner specific values to be specified in the Trading Partner Details form and the entries in the Integration Repository are NOT Trading Partner specific. Moreover, there should not be a need to change maps on a per Trading Partner basis, and if it does, then those maps should not be part of the Integration Repository entries. Given the current data model however, it is possible that a given map could differ by External Transaction Type / Subtype and even by Trading Partner. Going forward, this would not be allowed for seeded maps and the Integration Repository parser would return an error if it finds multiple maps which point to the same Internal Transaction Type / Subtype.

Additional Notes

* Parameters are typically used in outbound maps for specifying keys used in queries to produce outbound data

例

参照用として、注釈付けされた XML ゲートウェイ・インタフェースの例を次に示します。

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- $Header: MapPrinter.java 115.12 2009/06/13 21:17:58 mtai noship
$
-->
<!-- WARNING: This file should only be edited using Message Designer
-->
<?xGateway mapType="MAP" ?>
<?xGatewayVersion designerVersion="2.6.3.0.0" ?>
<ECX_MAPPINGS>
<MAP_CODE>Create Purchase Order</MAP_CODE>
<DESCRIPTION>This is a sample map to demonstrate annotations in the
Interface Repository.
/*#
 * Sample map annotation public description.
 *
 * @rep:interface PO:POC
 * @rep:standard OAG 7.2 Process_PO_001
 * @rep:direction IN
 * @rep:scope public
 * @rep:displayname Create Purchase Order
 * @rep:lifecycle active
 * @rep:product PO
 * @rep:compatibility S
 * @rep:category BUSINESS_OBJECT PURCHASE_ORDER
 * @rep:businessevent oracle.apps.po.received
 */
/*#
 * Sample map annotation public description for reused transaction
 *
 * @rep:interface PO:POU
 * @rep:standard OAG 7.2 Process_PO_001
 * @rep:direction IN
 * @rep:scope public
 * @rep:displayname Update Purchase Order
 * @rep:lifecycle active
 * @rep:product PO
 * @rep:compatibility S
 * @rep:category BUSINESS_OBJECT PURCHASE_ORDER
 * @rep:businessevent oracle.apps.po.received
 */
</DESCRIPTION>
<OBJECT_ID_SOURCE>1</OBJECT_ID_SOURCE>
<OBJECT_ID_TARGET>2</OBJECT_ID_TARGET>
<ENABLED>Y</ENABLED>
<ECX_MAJOR_VERSION>2</ECX_MAJOR_VERSION>
<ECX_MINOR_VERSION>6</ECX_MINOR_VERSION>
<ECX_OBJECTS>
<OBJECT_ID>1</OBJECT_ID>
<OBJECT_NAME>SRC</OBJECT_NAME>
<OBJECT_TYPE>XML</OBJECT_TYPE>
<OBJECT_DESCRIPTION>Source Definition</OBJECT_DESCRIPTION>
<OBJECT_STANDARDOAG</OBJECT_STANDARDOAG>
<ROOT_ELEMENT>INVENTORY</ROOT_ELEMENT>

<ECX_OBJECT_LEVELS>
<OBJECTLEVEL_ID>0</OBJECTLEVEL_ID>
<OBJECT_ID>1</OBJECT_ID>
<OBJECT_LEVEL>0</OBJECT_LEVEL>
<OBJECT_LEVEL_NAME>INVENTORY</OBJECT_LEVEL_NAME>
<PARENT_LEVEL>0</PARENT_LEVEL>
<ENABLED>Y</ENABLED>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>0</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>0</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>INVENTORY</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>

```

```

<PARENT_ATTRIBUTE_ID>0</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>Y</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>0</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>1</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>VERSION_INFO</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>0</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>0</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>2</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>SAVED_WITH</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>1</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>0</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>3</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>MINIMUM_VER</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>1</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>0</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>4</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>HOME_LIST</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>0</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>

```

```

<OBJECTLEVEL_ID>0</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>5</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>HOME</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>4</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>4</HAS_ATTRIBUTES>
<LEAF_NODE>0</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>0</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>6</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>NAME</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>5</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>2</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>0</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>7</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>LOC</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>5</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>2</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>0</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>8</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>TYPE</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>5</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>2</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>0</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>9</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>IDX</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>5</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>2</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>

```

```

<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
</ECX_OBJECT_LEVELS>
</ECX_OBJECTS>
<ECX_OBJECTS>
<OBJECT_ID>2</OBJECT_ID>
<OBJECT_NAME>TGT</OBJECT_NAME>
<OBJECT_TYPE>XML</OBJECT_TYPE>
<OBJECT_DESCRIPTION>Target Definition</OBJECT_DESCRIPTION>
<OBJECT_STANDARD>OAG</OBJECT_STANDARD>
<ROOT_ELEMENT>INVENTORY</ROOT_ELEMENT>

<ECX_OBJECT_LEVELS>
<OBJECTLEVEL_ID>1</OBJECTLEVEL_ID>
<OBJECT_ID>2</OBJECT_ID>
<OBJECT_LEVEL>0</OBJECT_LEVEL>
<OBJECT_LEVEL_NAME>INVENTORY</OBJECT_LEVEL_NAME>
<PARENT_LEVEL>0</PARENT_LEVEL>
<ENABLED>Y</ENABLED>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>1</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>0</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>INVENTORY</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<PARENT_ATTRIBUTE_ID>0</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>Y</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>1</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>1</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>VERSION_INFO</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>0</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>1</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>2</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>SAVED_WITH</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>1</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>1</OBJECTLEVEL_ID>

```

```

<ATTRIBUTE_ID>3</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>MINIMUM_VER</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>1</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>1</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>4</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>HOME_LIST</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>0</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>1</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>5</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>HOME</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>4</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>4</HAS_ATTRIBUTES>
<LEAF_NODE>0</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>1</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>6</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>NAME</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>5</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>2</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>1</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>7</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>LOC</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>5</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>2</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>

```

```

<IS_MAPPED>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>1</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>8</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>TYPE</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>5</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>2</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>1</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>9</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>IDX</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>5</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>2</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
</ECX_OBJECT_LEVELS>
</ECX_OBJECTS>
</ECX_MAPPINGS>
<SCRIPT SRC="/oracle_smp_chronos/oracle_smp_chronos.js"></SCRIPT>

```

ビジネス・イベント・アノテーション

この項では、ビジネス・イベントの統合リポジトリ・アノテーションについて把握する必要がある内容について説明し、次のトピックが含まれます。

- ビジネス・イベントの注釈付け
- ビジネス・イベントのアノテーション - 構文
- 必須のアノテーション
- オプションのアノテーション
- テンプレート
- 例

ビジネス・イベントの注釈付け

- ビジネス・イベントは、*.wfx ファイルで注釈付けする必要があります。
- イベントのみを注釈付けする必要があります。サブスクリプションを注釈付けする必要はありません。これらは、統合リポジトリで使用できません。

- 注釈付けを行う前に、`/*#`で始まるコメントが存在しないことを確認してください。これらのスラッシュ、アスタリスク、シャープの文字はリポジトリ・アノテーションの開始を示すために使用するものとなり、通常のコメントで使用すると、エラーまたは不明な動作が発生します。
- 注釈付けするには、**Emacs** や **vi** などのテキスト・エディタを使用して、ファイルを編集します。
- **.wfx** ファイルで、ビジネス・イベントの<IREP_ANNOTATION>タグ内にアノテーションを配置します。<IREP_ANNOTATION>タグは、<WF_EVENTS>タグの子ノードです。
- **.wfx** ファイルに複数のビジネス・イベント定義がある場合、それぞれのビジネス・イベント定義を個別に注釈付けする必要があります。つまり、適切なビジネス・イベントの<IREP_ANNOTATION>タグ内にアノテーションを配置する必要があります。
- ビジネス・イベントが呼び出される条件と、ビジネス・イベントを起動する UI 処理を含む、わかりやすい説明を入力します。
- **FND_APPLICATION** で製品コードを定義します。
- イベントに対して、既存のビジネス・エンティティを使用します。既存のビジネス・エンティティのリストは、「ビジネス・エンティティ・アノテーション・ガイドライン」(ページ **A-41**)を参照してください。
- 最終的にイベントの注釈付けも公開も行わないことを決定した場合は、アノテーションのみを削除し、関連付けられたタグは削除しないでください。

 <IREP_ANNOTATION/>タグまたは
 <IREP_ANNOTATION></IREP_ANNOTATION>タグが存在することで、アノテーションについてビジネス・イベントがレビューされており、統合リポジトリに公開する必要がないことがローダーに示されます。次回ユーザーがこれらのイベントをダウンロードすると、ローダーによって空の<IREP_ANNOTATION>タグが挿入されます。
- ビジネス・イベントの<IREP_ANNOTATION>タグ全体を削除して、それをアップロードすると、後続のダウンロードで、ローダーによって、ビジネス・イベントについての部分的に入力されたアノテーション・テンプレートが挿入されます。

ビジネス・イベントのアノテーション - 構文

ビジネス・イベントのアノテーションは、次のとおりです。


```

<IREP_ANNOTATION>
/*#
* This event is raised after the Purchase Order has been pushed
* to Oracle Order management open interface tables.This event
* will start the workflow OEOI/R_OEOI_ORDER_IMPORT to import the
* order.
*@rep:scope public
*@rep:displayname OM Generic Inbound Event
*@rep:product ONT
*@rep:category BUSINESS_ENTITY ONT_SALES_ORDER
*/
</IREP_ANNOTATION>

```

要素の定義の詳細は、アノテーションの一般的なガイドライン(ページ A-1)を参照してください。

必須のアノテーション

次のリンクをたどって、各アノテーションの構文と用途を確認してください。

- 説明文で始める必要あり
- rep:displayname、ページ A-125
- rep:scope、ページ A-123
- rep:product、ページ A-124
- rep:category BUSINESS_ENTITY、A-132 ページ

オプションのアノテーション

- link、ページ A-129
- see、ページ A-130
- rep:lifecycle、ページ A-127
- rep:compatibility、ページ A-128
- rep:ihelp、ページ A-131
- rep:metalink、ページ A-132

テンプレート

.wfx ファイルを注釈付けするときは、このテンプレートを使用できます。

```

.
.
.
<oracle.apps.wf.event.all.sync>
.
.
.
<WF_TABLE_DATA>
  <WF_EVENTS>
    <VERSION>...</VERSION>
    <GUID>....</GUID>
    <NAME>event name</NAME>
    <TYPE>EVENT</TYPE>
    <STATUS>ENABLED</STATUS>
    <GENERATE_FUNCTION/>
    <OWNER_NAME> ...</OWNER_NAME>
    <OWNER_TAG>...</OWNER_TAG>
    <CUSTOMIZATION_LEVEL>...</CUSTOMIZATION_LEVEL>
    <LICENSED_FLAG>..</LICENSED_FLAG>
    <DISPLAY_NAME>...</DISPLAY_NAME>
    <DESCRIPTION> Description for business event </DESCRIPTION>
    <IREP_ANNOTATION>

/*#
 * Put your long package description here; it can span multiple lines.
 *
 * @rep:scope <scope>
 * @rep:displayname <display name>
 * @rep:product <product or pseudoproduct short code>
 * @rep:category BUSINESS_ENTITY <entity name>
 */
</IREP_ANNOTATION>
</WF_EVENTS>
</WF_TABLE_DATA>

.
.
.
<WF_TABLE_DATA>
  <WF_EVENTS>
    <VERSION>...</VERSION>
    <GUID>....</GUID>
    <NAME>event name</NAME>
    <TYPE>EVENT</TYPE>
    <STATUS>ENABLED</STATUS>
    <GENERATE_FUNCTION/>
    <OWNER_NAME> ...</OWNER_NAME>
    <OWNER_TAG>...</OWNER_TAG>
    <CUSTOMIZATION_LEVEL>...</CUSTOMIZATION_LEVEL>
    <LICENSED_FLAG>..</LICENSED_FLAG>
    <DISPLAY_NAME>...</DISPLAY_NAME>
    <DESCRIPTION> Description for business event </DESCRIPTION>
    <IREP_ANNOTATION>

/*#
 * Put your long package description here; it can span multiple lines.
 *
 * @rep:scope <scope>
 * @rep:displayname <display name>
 * @rep:product <product or pseudoproduct short code>
 * @rep:category BUSINESS_ENTITY <entity name>
 */
</IREP_ANNOTATION>
</WF_EVENTS>
</WF_TABLE_DATA>

```

```
.  
. .  
. .  
</oracle.apps.wf.event.all.sync>
```

例

参照用として、注釈付けされた.wfx ファイルの例を次に示します。

```

<?xml version="1.0" encoding="UTF-8" ?>
- <!-- $Header: oeevtname.wfx 120.0 2005/06/01 23:11:59 appldev
  noship
$ -->
- <!-- dbdrv: exec java oracle/apps/fnd/wf WFXLoad.class java
  &phase=daa+38 \ -->
- <!-- dbdrv: checkfile(115.2=120.0):~PROD:~PATH:~FILE \ -->
- <!-- dbdrv: -u &un_apps &pw_apps &jdbc_db_addr &jdbc_protocol US
  \ --
>
- <!-- dbdrv: &fullpath_~PROD_~PATH_~FILE -->
- <oracle.apps.wf.event.all.sync>
- <ExternalElement>
- <OraTranslatibility>
- <XlatElement Name="WF_EVENTS">
- <XlatID>
  <Key>NAME</Key>
</XlatID>
  <XlatElement Name="DISPLAY_NAME" MaxLen="80" Expansion="50" />
- <XlatID>
  <Key Type="CONSTANT">DISPLAY_NAME</Key>
</XlatID>
  <XlatElement Name="DESCRIPTION" MaxLen="2000" Expansion="50" />
- <XlatID>
  <Key Type="CONSTANT">DESCRIPTION</Key>
</XlatID>
</XlatElement>
</OraTranslatibility>
</ExternalElement>
- <WF_TABLE_DATA>
+ <WF_EVENTS>
  <VERSION>1.0</VERSION>
  <GUID>A3BBD9D401776AE4E0340800208ACA52</GUID>
  <NAME>oracle.apps.ont.oi.po_ack.create</NAME>
  <TYPE>EVENT</TYPE>
  <STATUS>ENABLED</STATUS>
  <GENERATE_FUNCTION />
  <OWNER_NAME>Oracle Order Management</OWNER_NAME>
  <OWNER_TAG>ONT</OWNER_TAG>
  <CUSTOMIZATION_LEVEL>L</CUSTOMIZATION_LEVEL>
  <LICENSED_FLAG>Y</LICENSED_FLAG>
  <DISPLAY_NAME>Event for 3A4 Outbound Acknowledgment</DISPLAY_NAME>
  <DESCRIPTION>Event for 3A4 Outbound Acknowledgment</DESCRIPTION>
  <IREP_ANNOTATION>/*# * This event confirms the buyer of the results of
order import.This event will start the workflow
OEQA/R_OEOA_SEND_ACKNOWLEDGMENT.* * @rep:scope public * @rep:
displayname Event for 3A4 Outbound Acknowledgment * @rep:product ONT *
@rep:category BUSINESS_ENTITY ONT_SALES_ORDER */</IREP_ANNOTATION>
</WF_EVENTS>
</WF_TABLE_DATA>
- <WF_TABLE_DATA>
- <WF_EVENTS>
  <VERSION>1.0</VERSION>
  <GUID>9B8BF9DB705D09C9E0340800208ACA52</GUID>
  <NAME>oracle.apps.ont.oi.po_inbound.create</NAME>
  <TYPE>EVENT</TYPE>
  <STATUS>ENABLED</STATUS>
  <GENERATE_FUNCTION />
  <OWNER_NAME>Oracle Order Management</OWNER_NAME>
  <OWNER_TAG>ONT</OWNER_TAG>
  <CUSTOMIZATION_LEVEL>L</CUSTOMIZATION_LEVEL>
  <LICENSED_FLAG>Y</LICENSED_FLAG>
  <DISPLAY_NAME>OM Generic Inbound Event</DISPLAY_NAME>
  <DESCRIPTION>OM Generic Inbound Event</DESCRIPTION>
  <IREP_ANNOTATION>/*# * This event is raised after the Purchase Order

```

```

has been pushed to Oracle Order management open interface tables.This
event will start the workflow OEOI/R_OEOI_ORDER_IMPORT to import the
order.* * @rep:direction OUT * @rep:scope public * @rep:displayname OM
Generic Inbound Event * @rep:lifecycle active * @rep:product ONT *
@rep: compatibility S * @rep:category BUSINESS_ENTITY ONT_SALES_ORDER
*/</IREP_ANNOTATION>
</WF_EVENTS>
</WF_TABLE_DATA>
</oracle.apps.wf.event.all.sync>

```

ビジネス・エンティティ・アノテーション・ガイドライン

ビジネス・エンティティとは、ビジネス活動を実行するもの、またはそこでビジネス活動が実行されるものです。口座番号、従業員、発注書、顧客および領収書はすべて、ビジネス・エンティティの例としてあげられます。

ビジネス・エンティティの重要性とは

ビジネス・エンティティは、統合リポジトリのコンテキストで、非常に有用な検索基準です。統合リポジトリ UI の設計には、ビジネス・エンティティによる参照機能が含まれています。

リポジトリ・アノテーションでのビジネス・エンティティの出現場所

rep:category BUSINESS_ENTITY のアノテーションは、特定のインタフェースとビジネス・エンティティを関連付ける場所です。rep:category アノテーションの一般的な説明は、rep:category (ページ A-132)を参照してください。

注意: エンティティの表示名自体が十分に自己記述している特定のケースでは、それも説明となります。

既存のビジネス・エンティティ

カスタムの統合インタフェースでは、シード済または既存のビジネス・エンティティのみを使用できます。

注意: 統合リポジトリは、現時点で、カスタム製品ファミリやカスタム・ビジネス・エンティティの作成には対応していません。

次の表に、既存のビジネス・エンティティをリストします。

ビジネス・エンティティのリスト

ビジネス・エンティティ・コード	表示名	説明
AHL_DOCUMENT	文書	電子的な文書または文書参照
AHL_ITEM_COMPOSITION	追跡済品目構成	追跡済品目を構成する、品目グループまたは追跡しない品目のリストです。

ビジネス・エンティティ・コード	表示名	説明
AHL_ITEM_GROUP	代替品目グループ	保守の実行時に別の品目と交換できる類似品目のグループ。
AHL_MAINT_OPERATION	保守工程	リソースおよび資材の要件を定義します。これは作業の基本定義です。
AHL_MAINT_REQUIREMENT	保守要件	保守要件の定義です。ルート、品目の適用範囲またはユニット・インスタンスを定義します。また、時間およびカウンタに基づいて頻度を定義します。
AHL_MAINT_ROUTE	保守ルート	工程のセットが含まれ、処分、リソースおよび資材の要件を定義します。
AHL_MAINT_VISIT	保守訪問	タスクのブロックがあるユニットまたは品目インスタンス、保守作業が実行される組織と部署、およびその作業が遂行される時期が関連します。
AHL_MAINT_WORKORDER	保守作業指示	計画のある保守作業指示
AHL_MASTER_CONFIG	マスター構成	マスター構成は電気機械システム組立品の構成をモデル化します。
AHL_OSP_ORDER	外部サービス・オーダー	サード・パーティ組織による部品提供に必要な情報が含まれているオーダー。

ビジネス・エンティティ・コード	表示名	説明
AHL_PROD_CLASS	製品分類	保守および使用に関係するユニットまたは品目のカテゴリ化です。
AHL_UNIT_CONFIG	ユニット構成	ユニット構成は組立済電気機械システムの構成を記述します。
AHL_UNIT_EFFECTIVITY	ユニット保守計画スケジュール	期限のあるユニット保守計画
AHL_UNIT_SCHEDULES	ユニット使用イベント	飛行機のフライトなど、特定の期間での構成済ユニットの使用について記述するイベント。
AME_ACTION	承認処理	承認処理は、承認ルールの条件を満たしている場合に実行される処理を指定します。たとえば、「最初の上司3人までの承認必須」と指定します。
AME_APPROVAL	承認	承認
AME_APPROVER_GROUP	承認管理の承認者グループ	特定のビジネス・プロセス/取引の実行を承認するために割り当てられる承認者の事前定義グループ。
AME_APPROVER_TYPE	承認者タイプ	承認管理で使用できる承認者の分類。たとえば、すべてのHR従業員は承認管理で承認者タイプがPERとして分類されます。

ビジネス・エンティティ・コード	表示名	説明
AME_ATTRIBUTE	承認管理属性	承認が必要な取引のビジネス属性を収集するオブジェクト。たとえば、INVOICE_AMOUNTを、請求書の合計金額を収集する属性にできます。
AME_CONDITION	承認ルール条件	承認ルールを評価する承認管理属性に基づいた条件。属性INVOICE_AMOUNTの条件の例では「INVOICE_AMOUNT > 10,000 USD」のようにします。
AME_CONFIG_VAR	承認構成変数	承認管理における特定の動作を制御する承認構成のセット。
AME_ITEM_CLASS	承認管理項目区分	ヘッダー、明細品目、コスト・センターなどの様々なクラスへの特定の承認管理オブジェクトの分類です。
AME_RULE	承認ビジネス・ルール	承認ビジネス・ルールは一連の条件で構成され、それが満たされた場合にアクションが発生するように指定します(結果として承認者のリストが生成されます)。
AME_TRANSACTION_TYPE	承認取引タイプ	承認ポリシーを構成する承認属性、条件およびルールのセット。
AMS_BUDGETS	マーケティング予算	マーケティング・キャンペーン、イベントおよびその他のマーケティング活動の予算です。

ビジネス・エンティティ・コード	表示名	説明
AMS_CAMPAIGN	マーケティング・キャンペーン	マーケティング・キャンペーン
AMS_EVENT	マーケティング・イベント	マーケティング・イベント
AMS_LEAD	営業引合	営業引合
AMS_LIST	マーケティング・リスト	マーケティング・リスト
AMS_METRIC	マーケティング・メトリック	キャンペーンで生成された応答数など、マーケティング活動の測定値です。
AP_INVOICE	買掛/未払金請求書	買掛/未払金請求書
AP_PAYMENT	仕入先支払	仕入先支払
AP_PAYMENT_ADVICE	支払通知	支払通知
AP_SUPPLIER	仕入先	仕入先
AP_SUPPLIER_CONTACT	仕入先担当	仕入先担当
AP_SUPPLIER_SITE	仕入先サイト	仕入先サイト
AR_ADJUSTMENT	売掛/未収金請求書調整	売掛/未収金請求書調整
AR_BILLS_RECEIVABLE	受取手形	受取手形
AR_CHARGEBACK	チャージバック	チャージバック
AR_CREDIT_MEMO	クレジット・メモ	クレジット・メモ
AR_CREDIT_REQUEST	与信要求	与信要求
AR_DEBIT_MEMO	デビット・メモ	デビット・メモ
AR_DEPOSIT	預金	預金

ビジネス・エンティティ・コード	表示名	説明
AR_INVOICE	売掛/未収金請求書	売掛/未収金請求書
AR_PREPAYMENT	前払	前払
AR_RECEIPT	売掛/未収金領収書	売掛/未収金領収書
AR_REMITTANCE	送金	送金
AR_REVENUE	収益	収益
AR_SALES_CREDIT	販売実績	販売実績
AR_SALES_TAX_RATE	消費税率	消費税率
ASN_OPPORTUNITY	営業商談	営業商談
ASN_SALES_TEAM	営業チーム	商談またはアカウント、あるいは引合の営業チーム
ASO_QUOTE	営業見積(1)	営業見積は、潜在的顧客に提案するソリューションにおける、製品、価格、条件などの詳細な情報を含むビジネス・オブジェクトです(1)。
AS_OPPORTUNITY	営業商談(1)	営業商談(1)
BEN_CWB_3RD_PARTY_STOCK_OPTS	第三者ストック・オプション	第三者ストック・オプション
BEN_CWB_AUDIT	報酬ワークベンチ監査	報酬ワークベンチ・ユーザー・セッション内のすべての変更イベントを記録します。これにはすべての報酬エレメントが含まれます。
BEN_CWB_AWARD	報酬ワークベンチ取得項目	従業員の金銭的な取得項目です。たとえば、昇給、賞与、株など。

ビジネス・エンティティ・コード	表示名	説明
BEN_CWB_BUDGET	報酬ワークベンチ予算計画	基本給および賞与を含む、マネージャに分配可能な金銭や株の予算計画です。
BEN_CWB_PERSON	報酬ワークベンチ個人	報酬ワークベンチ処理に対する特定の日付における HR 個人のスナップショット。
BEN_CWB_PLAN	報酬ワークベンチ・プラン	昇給プラン、賞与プランまたはストック・オプション・プランなどの報酬プラン。
BEN_CWB_TASK	報酬ワークベンチ・タスク	報酬ワークベンチ・プランの管理で実行されるタスクです。予算計画、金額の割当、作業および承認の発行など。
BIS_REPORT	BIS レポート	BIS レポート
BOM_BILL_OF_MATERIAL	部品構成表	このインタフェースは、すべてのタイプの部品構成表を追加、変更および削除します。
BOM_MFG_ROUTING	製品製造工順	組立品の製造に必要な、その部品構成表に従ったステップごとの作業を定義する工順。
BOM_PRODUCT_FAMILY	製品ファミリ	計画目的での製品ファミリ
CAC_APPOINTMENT	約束	所定の日付および期間の約束または会議

ビジネス・エンティティ・コード	表示名	説明
CAC_BUSINESS_OBJECT_META_DATA	ビジネス・オブジェクト・メタ・データ定義	ビジネス・エンティティのメタデータ定義。外部ビジネス・エンティティの動的リンクに使用されます。また、エンティティ詳細の問合せ、動的 LOV 構築およびページ検索にも使用されます。
CAC_CAL_TASK	カレンダー・タスク	時間を制限するタスクまたは ToDo としてユーザーのカレンダーに表示されるタスク。
CAC_NOTE	ノート	様々なビジネス・オブジェクトに関連付けられるノートまたはコメント
CAC_RS_TIME_BOOKING	リソースの時間予約	個人または非個人(会議室など)のリソースの時間予約
CAC_SCHEDULE	スケジュール	スケジュール
CAC_SCHEDULE_TEMPLATE	スケジュール・テンプレート	スケジュール・テンプレート
CAC_SYNC_SERVER	カレンダー同期サーバー	タスク、約束、担当者などのカレンダー・エンティティを外部カレンダーと同期するためのカレンダー・サーバー。
CAC_TASK_TEMPLATE	カレンダー・タスク・テンプレート	カレンダー・タスク・テンプレート
CCT_ADVANCED_TELEPHONY_SDK	拡張テレフォニ SDK	この SDK によりサーバー側統合を使用して Oracle E- Business Suite とのテレフォニの統合が可能です。
CCT_BASIC_TELEPHONY_SDK	基本テレフォニ SDK	この SDK によりクライアント側統合を使用して Oracle E- Business Suite とのテレフォニの統合が可能です。

ビジネス・エンティティ・コード	表示名	説明
CE_BANK_STATEMENT	銀行取引明細書	銀行取引明細書
CE_RECONCILIATION_ITEM	調整項目	調整項目
CHV_PLANNING_SCHEDULE	バイヤー予測	バイヤー予測
CHV_SHIPPING_SCHEDULE	バイヤー出荷要求	バイヤー出荷要求
CLN_TRADING_PARTNER_COLL	コラボレーション取引先	取引先
CLN_TRADING_PARTNER_COLL_EVENT	取引先コラボレーション・イベント	取引先コラボレーション・イベント
CN_COMP_PLANS	インセンティブ報酬計画	インセンティブ報酬計画
CN_INCENTIVES	インセンティブ報酬	営業担当、パートナまたは顧客に対する金銭的または非金銭的な謝礼となる変動報酬またはリベート。
CSD_REPAIR_ESTIMATE	修理見積	修理見積は修理を行うためのコスト合計を表し、資材、作業および費用の詳細が含まれます。
CSD_REPAIR_LOGISTICS	修理ロジスティクス	修理ロジスティクスは、修理中の顧客品目および貸出中の品目の受入と出荷を追跡します。
CSD_REPAIR_ORDER	修理オーダー(1)	修理オーダー(1)
CSF_TASK_DEBRIEF	サービス・タスク報告	資材、作業および費用のサービス・タスク報告

ビジネス・エンティティ・コード	表示名	説明
CSI_COUNTER	カウンタ	マトリックスの様々なタイプを定義および保守するメカニズムを提供します。これらは導入ベース・インスタンスまたはサービス契約詳細などの Oracle E-Business Suite のオブジェクトに関連付け可能です。
CSI_ITEM_INSTANCE	品目インスタンス	導入ベース品目インスタンス
CST_DEPARTMENT_OVERHEAD	製造部門の間接費レート	製造部門の間接費レート
CST_ITEM_COST	在庫品目原価	在庫品目原価
CST_RESOURCE_COST	製造資源単価	製造資源単価
CS_SERVICE_CHARGE	サービス料	サービス料
CS_SERVICE_REQUEST	サービス要求	サービス要求
CZ_CONFIG	構成	構成
CZ_CONFIG_MODEL	構成モデル	構成モデル
CZ_MODEL_PUB	構成モデル公開	構成モデル公開
CZ_RP_FOLDER	コンフィギュレータ・リポジトリ・フォルダ	コンフィギュレータ・リポジトリ・フォルダ
CZ_USER_INTERFACE	構成モデル・ユーザー・インタフェース	構成モデル・ユーザー・インタフェース
DPP_EXECUTION_REQUEST	Execution Integration Request	DPP と他のアプリケーションの統合用エンティティです。AR、AP などの外部アプリケーションとの統合で DPP UI およびコンカレント・プログラムから起動されるイベントで使用されます。

ビジネス・エンティティ・コード	表示名	説明
DPP_TRANSACTION_APPROVAL	Transaction Approval Notification	このエンティティは、DPP 取引の AME 承認に対して定義されています。これは、新規 DPP 取引で「承認の要求」ボタンをクリックした際に UI で参照されます。
DPP_XMLG_OUTBOUND	Outbound preapproval process	価格保護のための Oracle XML Gateway を介して承認前プロセスをトリガーするイベントが使用するエンティティです。
EAM_ASSET_ACTIVITY_ASSOCIATION	保守資産活動関連	保守資産活動関連
EAM_ASSET_ACTIVITY_SUPPRESSION	資産活動抑制関係	資産の予防保守活動が別の活動のパフォーマンスのために抑制されることを示します。
EAM_ASSET_AREA	保守資産エリア	保守資産エリア
EAM_ASSET_ATTRIBUTE_GROUPS	保守資産属性グループ	保守資産属性グループ
EAM_ASSET_ATTRIBUTE_VALUE	保守資産属性値	保守資産属性値
EAM_ASSET_METER	保守資産メーター関連	保守資産メーター関連
EAM_ASSET_NUMBER	保守資産番号	保守資産番号
EAM_ASSET_ROUTE	保守資産ルート	保守資産ルート
EAM_COMPLETE_WO_OPERATION	保守作業完了	保守作業完了
EAM_DEPARTMENT_APPROVER	保守部門承認者	保守部門承認者 - ユーザーまたは職責

ビジネス・エンティティ・コード	表示名	説明
EAM_METER	メーター	メーター
EAM_METER_READING	メーター・リーディング	メーター・リーディング
EAM_PARAMETER	保守設定	保守設定
EAM_PM_SCHEDULE	予防保守スケジュール	予防保守スケジュール
EAM_SET_NAME	保守セット	保守セット
EAM_WORK_ORDER	資産保守作業指示	資産保守作業指示
EAM_WORK_REQUEST	保守作業要求	保守作業要求
ECX_CONFIRM_BOD	XML ゲートウェイ確認メッセージ	XML ゲートウェイ確認メッセージ
ECX_MESSAGE_DELIVERY	XML ゲートウェイ・メッセージ送信	Oracle および非 Oracle のメッセージ・システムで使用され、送信状況を報告します。ステータス情報は XML ゲートウェイ・ログ表に書き込まれて取引送信データを追跡し報告します。
ECX_TRADING_PARTNER	XML ゲートウェイ取引先	ビジネス・メッセージを交換する特定の所在地のビジネス・パートナーを表します。これには、顧客、仕入先、銀行支店または社内事業所があります。
ECX_TRANSFORMATION	XML ゲートウェイ変換	このインタフェースは XML メッセージにスタイル・シートを適用するために使用され、コール環境による追加処理のために変換された XML メッセージを戻します。

ビジネス・エンティティ・コード	表示名	説明
EC_CODE_CONVERSION	コード変換	たとえば通貨コードや単位など、Oracle の内部コードを外部システム・コードに、あるいはその逆に変換します。
EC_EDITION_TRANSACTION_LAYOUT	EDI 取引レイアウト定義レポート	EDI 取引レイアウト定義レポート
EC_INBOUND	インバウンド EDI メッセージ	取引先からシステムに送信された EDI メッセージです。
EC_OUTBOUND	アウトバウンド EDI メッセージ	システムから取引先に送信された EDI メッセージです。
EC_TP_MERGE	取引先のマージ	Trading Community Architecture (TCA)におけるアカウントのマージの結果としての取引先のマージを示します。
EDR_EVIDENCE_STORE	電子レコード証拠ストア	電子レコード証拠ストア
EDR_ISIGN_FILE_UPLOAD	ファイル・アップロード承認要求	ファイル・アップロード承認要求
EGO_ITEM	カタログ品目	品目カタログにリストされた品目。
EGO_USER_DEFINED_ATTR_GROUP	PLM ユーザー定義属性	このインタフェースは、すべてのエンティティのユーザー定義属性を追加、変更、削除および問合せします。
ENG_CHANGE_ORDER	製品変更オーダー	製品または設計変更
FA_ASSET	資産	Oracle Assets に資産を追加するインタフェース。

ビジネス・エンティティ・コード	表示名	説明
FA_CAPITAL_BUDGET	資産購買予算	Oracle Assets に資本資産をアップロードするインタフェース。
FA_LEASE_PAYMENT	リース支払	Oracle Payables にリース支払明細を送信するインタフェース。
FEM_ACCOUNT_FACT	分析銀行口座情報	詳細なレベルの金融口座データ
FEM_BALANCES_FACT	分析残高	元帳入力および元帳収益性処理結果を含みます。
FEM_FACT_REPOSITORY	企業分析ファクト・リポジトリ	複数のディメンションで分類できる数値ファクト(多くの場合測定値と呼ばれます)を含みます。詳細レベルのファクトまたは集計されたファクトのいずれかを含みます。
FEM_STATISTICAL_FACT	分析統計情報	ディメンションの数値的な評価値。これらの評価値は、導出された値や経験的に得られる値の両方の実際の統計値です。
FEM_TRANSACTION_FACT	分析取引情報	金融口座に対するイベントおよび顧客対応の件数を表す値。
FEM_XDIM_ACTIVITY	分析アクティビティ	別のディメンションに対する繰り返し可能なタスクを記述し、処理により定義され、品目に対して実行されます。ビジネス・プロセスおよび個人の処理などが含まれます。

ビジネス・エンティティ・コード	表示名	説明
FEM_XDIM_AUXILIARY	補助分析ディメンション	Enterprise Performance Foundation の「非 Foundation」ディメンションを示します。Foundation とは異なり、これらは付加価値処理の計算エンジンには使用されません。
FEM_XDIM_BUDGET	分析予算	予算および予測を識別します。
FEM_XDIM_CAL_PERIOD	分析カレンダー期間	分析カレンダー期間
FEM_XDIM_CCTR_ORG	分析組織	会社およびコスト・センターで構成される標準分析組織ディメンションを示します。
FEM_XDIM_CHANNEL	分析チャネル	流通および販売のチャネルを識別します。
FEM_XDIM_COMPANY	会社ディメンション	標準分析会社ディメンション
FEM_XDIM_COST_CENTER	コスト・センター・ディメンション	標準分析コスト・センター・ディメンション
FEM_XDIM_COST_OBJECT	分析原価計算対象	原価計算対象はコストを記述した複数ディメンションのエンティティです。
FEM_XDIM_CUSTOMER	分析顧客	分析データとビジネスの関係があるグループまたは個人を識別します。
FEM_XDIM_DATASET	分析データセット	分析データの一般コンテナを識別します。
FEM_XDIM_ENTITY	分析連結エンティティ	Global Consolidation System ユーザーの連結、消去および営業の各エンティティを識別します。

ビジネス・エンティティ・コード	表示名	説明
FEM_XDIM_FINANCIAL_ELEM	分析財務要素	残高、統計およびレポートの各金額タイプのカテゴリを識別します。
FEM_XDIM_GENERIC_FACT_DATA	分析ユーザー定義ファクト・データ	ユーザー定義ディメンションのファクト・データの保存に使用可能な表
FEM_XDIM_GEOGRAPHY	分析地域	地理的な所在地を識別します。
FEM_XDIM_HIERARCHY	分析ディメンション階層	ディメンション・メンバーの組織的な親子関係です。
FEM_XDIM_LEDGER	分析元帳	勘定科目の帳簿を識別します。会計帳簿と類似しています。
FEM_XDIM_LEVEL	分析ディメンション・レベル	ディメンション・メンバーのカテゴリを識別します。
FEM_XDIM_LINE_ITEM	分析明細品目	通常は、勘定科目の拡張として、総勘定元帳の勘定科目を識別します。
FEM_XDIM_NATURAL_ACCOUNT	分析勘定科目	残高が、収益、費用、資本、資産および負債の5つの異なる残高タイプで転記される組織内の勘定科目を識別します。
FEM_XDIM_PRODUCT	分析製品	販売対象の商品またはサービスを識別します。
FEM_XDIM_PROJECT	分析プロジェクト	計画およびその実行を識別します。
FEM_XDIM_SIC	分析標準産業分類	標準産業分類システムのオフィシャル・コードを識別します。

ビジネス・エンティティ・コード	表示名	説明
FEM_XDIM_SIMPLE	値のみのディメンションの分析リスト	属性がなく値リストとしてのみ機能するすべての分析ディメンションのグループ化。
FEM_XDIM_SOURCE_SYSTEM	分析ソース・システム	ファクト・データおよびディメンション・データの原点を識別します。
FEM_XDIM_TASK	分析タスク	個々の操作および部分的な作業を識別します。
FEM_XDIM_USER_DIMENSION	分析ユーザー定義ディメンション	すべてのカスタマイズ可能分析属性ディメンションのグループ化です。
FF_FORMULA_FUNCTION	FastFormula 機能	コア FastFormula 機能に任意拡張機能を提供する外部プロシージャ・コールを表します。
FLM_FLOW_SCHEDULE	フロー・スケジュール	フロー・スケジュール
FND_APPS_CTX	アプリケーション・セキュリティ・コンテキスト	現行ユーザー・セッションを表すアプリケーション・コンテキスト
FND_CP_PROGRAM	コンカレント・プログラム	コンカレント処理システムで実行できる個別の作業単位。通常、コンカレント・プログラムは長期実行のデータ集中タスク(レポートの生成など)です。
FND_CP_REQUEST	コンカレント要求	指定のパラメータ値セット、繰り返すオプション・スケジュールおよびオプションの後処理を含むプログラムを実行するための、コンカレント処理システムへの要求です。

ビジネス・エンティティ・コード	表示名	説明
FND_CP_REQUEST_SET	コンカレント要求セット	印刷オプションとパラメータ値を事前定義して複数のコンカレント・プログラムを実行する便利な方法。 要求セット・グループはセットにより発行されたステージに要求します。
FND_EBS_MOBILE	モバイル最適化 API	モバイル最適化 API は、 Oracle E- Business Suite モバイル・アプリケーション開発用に設計および作成された軽量 API です。
FND_FLEX_KFF	キー・フレックスフィールド	カスタマイズ可能な複数セグメント・フィールド
FND_FORM	アプリケーション・フォーム	フォームは、「ナビゲータ」ウィンドウを使用してそこに移動できる、特別な機能のクラスです。
FND_FUNCTION	アプリケーション機能	機能は、機能セキュリティの提供を目的として、アプリケーションの機能の一部として一意の名前で登録されます。
FND_FUNC_SECURITY	機能セキュリティ	機能セキュリティは、承認済ユーザーに対してアプリケーションの機能を制限します。
FND_GFM	アプリケーション・ファイル	一般ファイル・マネージャは、ファイルをアップロード/ダウンロードする方法を提供し、ファイル属性を操作します。
FND_LDAP_OPERATIONS	LDAP ディレクトリ	Oracle E-Business Suite による統合 OID に対する操作の実行を可能にします。

ビジネス・エンティティ・コード	表示名	説明
FND_MBL_SAMPLE	サンプル・モバイル・インタフェース	Oracle E-Business Suite Mobile Foundation のサンプル・モバイル・アプリで 사용되는サンプル・インタフェース。これらのインタフェースは本番用に設計されたものではなく、デモ用にのみ使用されます。
FND_MENU	アプリケーション・メニュー	「ナビゲータ」に表示される機能および機能のメニューの階層配置。
FND_MESSAGE	アプリケーション・メッセージ・ディクショナリ	Oracle E-Business Suite 全体のメッセージのカatalog/リポジトリを含みます。メッセージ・ディクショナリ機能はアプリケーションからの表示およびロギングに使用されます。
FND_NAVIGATION	アプリケーション・ナビゲーション	アプリケーション内のページ間をナビゲートする標準的な方法
FND_OBJECT_CLASSIFICATION	OATM オブジェクト-表領域分類	このエンティティにはシードされた明示的な OATM オブジェクト-表領域分類が保管され、さらにカスタマイズ可能です。
FND_PROFILE	ユーザー・プロファイル	実行時のアプリケーションの動作方法に影響する、変更可能なオプションのセットです。
FND_RESPONSIBILITY	職責	職責により Oracle E-Business Suite の製品のメニュー構造が定義されます。
FND_SSO_MANAGER	シングル・サイン・オン・マネージャ	シングル・サイン・オンおよび中央のログインに関連する API

ビジネス・エンティティ・コード	表示名	説明
FND_TABLESPACE	Oracle Applications 表領域 モデル表領域	すべての格納関連オブジェクトを分類します。論理表領域は物理表領域と1:1の関係です。
FND_USER	ユーザー	Oracle E-Business Suite のユーザーを表します。
FUN_ARAP_NETTING	買掛/未払金および売掛/未収金のネットティング	ネットティング用の買掛/未払金および売掛/未収金
FUN_IC_TRANSACTION	IC Manual Transaction	会社間取引は、1つの起動側と単一/複数の受入先との間で行われます。
FUN_INTERCOMPANY_BATCH	会社間取引セット	法的エンティティ間の取引を含む会社間バッチです。
FV_BUDGETARY_DISCOUNT	連邦予算割引	予算割引取引を作成します。
FV_BUDGET_JOURNAL	連邦予算実行文書	連邦予算表にインポートされる連邦予算レコードを含みます。
FV_FINANCE_CHARGE	連邦延滞金利	連邦延滞金利
FV_IPAC_DISBURSEMENT	IPAC 支出	IPAC 支出
FV_PRIOR_YEAR_ADJUSTMENT	前年度調整	前年度調整
FV_TREASURY_DISBURSEMENT	財政支出	財政確認、取消および無効な支出取引
FV_YEAR_END_CLOSE	連邦政府年度末クローズ情報	連邦政府年度末クローズ
GHR_DUTY_STATION	US Federal Workplace Duty Station	US Federal Workplace Duty Station

ビジネス・エンティティ・コード	表示名	説明
GHR_EEO_COMPLAINT	US Federal EEO Complaint	US Federal EEO Complaint GHR_POSITION_DESCRIPTION
職階摘要	職階摘要	
GHR_REQ_FOR_PERSONNEL_ACTION	人事処理要求	人事処理要求
GL_ACCOUNTING_SETUP_MANAGER	Accounting Setup Manager	これは、一般会計における元帳と法的エンティティの会計設定です。
GL_ACCOUNT_COMBINATION	General Ledger Code Combination	これは、勘定体系の下で定義された総勘定元帳勘定科目の組み合わせです。
GL_BC_PACKETS	予算資金管理取引パケット	予算資金管理取引パケット
GL_BUDGET_DATA	一般会計予算データ	一般会計予算データ
GL_CHART_OF_ACCOUNTS	勘定体系	勘定体系(COA)
GL_DAILY_RATE	日次通貨換算レート	日次通貨換算レート
GL_INTERCOMPANY_TRANSACTION	会社間取引	会社間取引
GL_JOURNAL	仕訳入力	仕訳入力
GL_PERIOD	General Ledger Accounting Period	これは、会計カレンダーで定義されている会計期間です。
GMD_ACTIVITIES_PUB	製品開発アクティビティ	アクティビティ情報を作成、変更または削除します。

ビジネス・エンティティ・コード	表示名	説明
GMD_FORMULA	プロセス製造算式	プロセス製造算式
GMD_OPERATION	プロセス製造工程	プロセス製造工程
GMD_OUTBOUND_APIS_PUB	プロセス製造品質アウトバウンド取引	サード・パーティ製品に情報をエクスポートする API を含む、パブリック・レベルのプロセス製造品質パッケージです。
GMD_QC_SAMPLES	プロセス製造品質サンプル	プロセス製造品質サンプル
GMD_QC_SPEC	プロセス製造品質仕様	プロセス製造品質仕様
GMD_QC_SPEC_VR	プロセス製造仕様使用ルール	プロセス製造仕様使用ルール
GMD_QC_TESTS_PUB	プロセス製造品質テスト	プロセス製造品質テスト
GMD_RECIPE	プロセス製造レシピ	プロセス製造レシピ
GMD_RECIPE_VALIDITY_RULE	プロセス製造レシピ使用ルール	プロセス製造レシピ使用ルール
GMD_RESULTS_PUB	プロセス製造品質テスト結果	プロセス製造品質テスト結果
GMD_ROUTING	プロセス製造工順	プロセス製造工順
GMD_STATUS_PUB	プロセス製造製品開発ステータス	工順、工程、レシピおよび妥当性ルールステータスを変更します。
GME_BATCH	プロセス製造バッチ	プロセス製造バッチ

ビジネス・エンティティ・コード	表示名	説明
GME_BATCH_STEP	プロセス製造バッチ・ステップ	プロセス製造バッチ・ステップ
GMF_ALLOCATION_DEFINITION	プロセス製造費用配賦定義	品目に間接費用(間接共通費)を割り振る設定データです。
GMF_BURDEN_DETAIL	プロセス製造会計間接費詳細	製造または購入された品目に割り当てられた間接費を示します。
GMF_ITEM_COST	プロセス製造会計品目原価	プロセス製造会計品目原価
GMF_RESOURCE_COST	プロセス製造会計生産資源原価	プロセス製造会計生産資源原価
GMI_ADJUSTMENTS	プロセス製造在庫調整	プロセス製造在庫調整
GMI_API	プロセス製造在庫設定	品目、ロット、ロット換算を作成、変更、削除するプロセス製造在庫取引です。
GMI_ITEM	プロセス製造品目	プロセス製造品目
GMI_ITEM_LOT_UOM_CONV	プロセス製造品目ロット単位換算	プロセス製造品目ロット単位換算
GMI_LOT	プロセス製造ロット	プロセス製造ロット
GMI_OM_ALLOC_API_PUB	プロセス製造受注在庫割当	OPM 受注割当 API は、Order Management の OPM 予約(割当)情報を作成、変更または削除できるビジネス・オブジェクトです。

ビジネス・エンティティ・コード	表示名	説明
GMI_PICK_CONFIRM_PUB	プロセス製造受注在庫 ピック確認	ピック確認 API は、プロセス移動オーダー明細または搬送詳細明細に対し、ピック確認や在庫のステージングを実行するビジネス・オブジェクトです。
GMP_CALENDAR_API	プロセス計画製造カレンダー	製造カレンダーを変更します。
GMP_GENERIC_RESOURCE	一般プロセス製造資源	プロセス製造における製造資源
GMP_PLANT_RESOURCE	プロセス製造工場資源	プロセス製造における工場固有の製造資源
GMP_RSRC_AVL_PKG	プロセス計画生産資源可用性	生産資源の可用性を変更します。
GMS_AWARD	プロジェクト取得予算	プロジェクト取得予算
HR_AUTHORIA_INTEGRATION_MAP	Authoria 統合マップ	Authoria 統合マップ
HR_BUDGET	HR 予算計画	HR 予算計画
HR_BUSINESS_GROUP	ビジネス・グループ	ビジネス・グループ
HR_CALENDAR_EVENT	HR カレンダー・イベント	HR カレンダー・イベント
HR_COST_CENTER	コスト・センター	コスト・センター
HR_EVENT	HR 登録可能イベント	HR 登録可能イベント
HR_HELP_DESK	HR Help Desk Integration	Peoplesoft Help Desk Integration points with the Oracle E-Business Suite HRMS
HR_KI_MAP	知識統合マップ	知識統合マップ

ビジネス・エンティティ・コード	表示名	説明
HR_KI_SYSTEM	知識統合システム	知識統合システム
HR_LEGAL_ENTITY	法的エンティティ	法的エンティティ
HR_LIABILITY_PREMIUM	損害賠償掛金	損害賠償掛金
HR_LOCATION	事業所	事業所
HR_MESSAGE_LINE	HRMS メッセージ行	HRMS メッセージ行
HR_OPERATING_UNIT	営業単位	営業単位
HR_ORGANIZATION	HRMS 組織	HRMS 組織
HR_ORGANIZATION_LINK	組織リンク	組織リンク
HR_PAY_SCALE	支給スケール	支給スケール
HR_PERSON	HR 個人	HR 個人
HR_PERSONAL_DELIVERY_METHOD	個人連絡方法	個人連絡方法
HR_ROLE	HRMS ロール	HRMS ロール
HR_SALARY_BASIS	給与ベース	給与ベース
HR_SELF_SERVICE_TRANSACTION	HR セルフサービス・トランザクション	セルフサービス・トランザクション
HR_SOC_INS_CONTRIBUTIONS	社会保険拠出金	社会保険拠出金
HR_SUPER_CONTRIBUTION	老齢退職拠出金	個人の退職に備えた基金への支払を示します。
HR_USER_HOOK	HRMS ユーザー・フック	HRMS ユーザー・フック
HXC_TIMECARD	タイムカード	タイムカード

ビジネス・エンティティ・コード	表示名	説明
HXC_TIMECARD_RECURRENCE_PERIOD	タイムカード固定処理期間	タイムカード固定処理期間
HXC_TIME_INPUT_SOURCE	勤怠入力ソース	タイムカード・データの 入力方法を示します。
HXC_TIME_RECIPIENT	勤怠受入先アプリケーション	Time and Labor データ を受信して処理するア プリケーション。
HZ_ACCOUNT_CONTACT	顧客アカウント担当者	顧客アカウントの担当者 である個人。
HZ_ADDRESS	取引先コミュニティ所在地	取引先コミュニティ・メン バーの所在地、たとえ ば、顧客またはパートナ の所在地です。
HZ_CLASSIFICATION	取引先コミュニティ分類	ユーザー定義または NAICS、NACE または SIC などの外部基準を 使用したパーティのカテ ゴリ化です。
HZ_CONTACT	取引先コミュニティ担当 者	ある組織の担当者として の個人またはその他の 個人。
HZ_CONTACT_POINT	連絡先	連絡する手段、たとえば 電話や E メール。
HZ_CONTACT_PREFERENCE	連絡作業環境	取引先が希望する連絡 の時間や方法について の情報です。
HZ_CUSTOMER_ACCOUNT	顧客アカウント	採用企業と販売の関係 がある個人または組織。
HZ_EXTERNAL_REFERENCE	取引先コミュニティ外部 参照	取引先コミュニティ・デー タベースと外部ソース・ システムとの間の運用 マッピングの管理。

ビジネス・エンティティ・コード	表示名	説明
HZ_GROUP	取引先コミュニティ・グループ	取引先コミュニティ・グループ
HZ_ORGANIZATION	取引先コミュニティ組織	組織タイプのパーティと財務や与信レポートなどの関連情報です。
HZ_PARTY	パーティ	ビジネス関係を構築できる個人または組織である、取引先コミュニティ・エンティティ。
HZ_PERSON	取引先コミュニティ個人	Person タイプのパーティ、および職業や学歴などの関連情報です。
HZ_RELATIONSHIP	取引先コミュニティ関連	142 つのパーティ間の関連を表したものであり、両者の関係上の各パーティの役割に基づきます。
HZ_RELATIONSHIP_TYPE	取引先コミュニティ関連タイプ	各パーティの関連上の役割のカテゴリ。
IBC_CONTENT_DELIVERY_MANAGER	コンテンツ配信マネージャ	コンテンツ配信マネージャ・クラスは、OCM コンテンツ・リポジトリに収納されたコンテンツ項目を取得する API をアプリケーションに提供します。
IBE_CATALOG_PUNCHOUT	Web ストアー・カタログ・パンチアウト	iStore カタログにある使用可能な品目を調達ユーザーが選択できるようにする処理です。iStore への調達ユーザーのログイン/ログアウトは透過的に行えます。
IBE_CONTENT	Web ストアーのコンテンツ	Web ストアー・ページのコンテンツ
IBE_ITEM	Web ストアー品目	Web ストアー製品品目

ビジネス・エンティティ・コード	表示名	説明
IBE_SALES_ORDER	Web ストアー受注	Web ストアー受注
IBE_SECTION	Web ストアー・セクション	Web コンテンツおよび製品のナビゲーション階層
IBE_SESSION_ATTRIBUTES	Web ストアー・セッション属性	Web ストアーにアクセスするユーザーのセッション属性
IBE_SHOPPING_CART	Web ストアー・ショッピング・カート	Web ストアー・ショッピング・カート
IBE_SHOPPING_LIST	Web ストアー・ショッピング・リスト	Web ストアー・ショッピング・リスト
IBE_SITE	Web ストアー・サイト	Web ストアー・サイト
IBE_TEMPLATE	Web ストアー・テンプレート	Web ストアー・ページ・テンプレート
IBE_USER	Web ストアー・ユーザー	ユーザー、担当者、顧客
IBW_PAGE_ACCESS_TRACKING	Web 分析ページ・アクセス追跡	Web 分析レポートに必要な訪問およびページ・アクセスのデータを収集します。
IBY_BANKACCOUNT	External Bank Account	Supplier or Customer Bank Account
IBY_CREDITCARD	クレジット・カード	クレジット・カード支払手段
IBY_EXCEPTION	IBY Exception	エラーが発生したとき IBY コードによって生成される例外です。
IBY_FUNDSCAPTURE_ORDER	資金獲得オーダー	受取要求人により支払システムに送付される単一の資金取得要求です。
IBY_PAYMENT	IBY Payment	仕入先に対する IBY 経由の支払を示します。

ビジネス・エンティティ・コード	表示名	説明
IEO_AGENT	顧客対応センター・エージェント	顧客対応イベントにおいて顧客に対応する個人。
IEX_COLLECTION_CASE	回収事例	回収事例
IEX_COLLECTION_DISPUTE	回収係争	係争では、顧客が認めないすべてまたは一部の請求書の解決のため、売掛/未収金にクレジット・メモ要求が作成されます。
IEX_COLLECTION_PROMISE	回収約束	回収約束
IEX_COLLECTION_SCORE	回収スコア	回収スコア
IEX_COLLECTION_STRATEGY	回収戦略	回収戦略
IEX_PROMISES	回収支払約束	支払約束は、特定の日付に支払を行うという、顧客との非拘束の契約です。
IEX_STRATEGY	売掛/未収金回収戦略	戦略は、未回収売掛金の収集プロセスを自動化するとともに、複雑な回収管理活動をサポートする、順番が事前構成された作業項目です。
IGC_CONTRACT_COMMITMENT	契約約定	契約約定
IGC_ENCUMBRANCE_JOURNAL	引当仕訳	引当仕訳
IGF_AWARD	学資援助学生奨学金	学資援助学生奨学金

ビジネス・エンティティ・コード	表示名	説明
IGF_BASE_RECORD	学資援助学生ベース・レコード	学資援助学生ベース・レコード
IGF_COA	学生出席コスト	学生出席コスト
IGF_DL	学資援助ダイレクト・ローン	学資援助ダイレクト・ローン
IGF_FFELP	学資援助 FFELP ローン	学資援助 FFELP ローン
IGF_FWS	学資援助作業研究	学資援助作業研究
IGF_ISIR	機関学生情報レコード	機関学生情報レコード
IGF_PELL	学資援助ペル奨学金	学資援助ペル奨学金
IGF_PROFILE	学生プロフィール・アプリケーション	学生プロフィール・アプリケーション
IGF_TODO	Financial Aid Student Todo Item	学資援助学生の実行項目
IGF_VERFN	学資援助確認項目	学資援助確認項目
IGS_ADM_APPLICATION	入学申請	入学申請
IGS_ADM_FEE	入学金	入学申請料
IGS_ADV_STAND	既得単位	既得単位
IGS_DA_REQUEST	学位監査要求	学位監査要求
IGS_INQ_APPLICATION	見込応募者の問合せ	見込応募者の問合せ
IGS_INSTITUTION	機関	機関パーティ
IGS_PARTY_CHARGE	高等教育パーティ賦課	高等教育パーティ・アカウント賦課取引
IGS_PARTY_CREDIT	高等教育パーティ・クレジット	高等教育パーティ・アカウント・クレジット取引

ビジネス・エンティティ・コード	表示名	説明
IGS_PARTY_REFUND	高等教育パーティ払戻	高等教育パーティ・アカウント払戻取引
IGS_PERSON_ALTERNATE_ID	代替個人識別子	個人の代替識別子(社会保障番号、運転免許証など)
IGS_PERSON_CONTACT	個人連絡先情報	個人連絡先情報
IGS_PREV_EDUCATION	教育前歴	教育前歴
IGS_PROGRAM	高等教育プログラム	高等教育プログラム
IGS_SPONSORSHIP	学生とスポンサの関係	学生とスポンサの関係
IGS_STUDENT_CONCENTRATION	学生の集合	学生の集合
IGS_STUDENT_PROGRAM	学生プログラム計画	学生プログラム計画
IGS_STUDENT_UNIT	学生単位計画	学生単位計画
IGS_TODO	学資援助学生の実行項目	学資援助学生の実行項目
IGS_UNIT	高等教育単位	高等教育単位
IGW_PROPOSAL	Grants プロポーザル	Grants プロポーザル
IGW_PROPOSAL_BUDGET	Grants プロポーザル予算	Grants プロポーザル予算
INV_ACCOUNTING_PERIOD	在庫会計期間	在庫会計期間のステータス
INV_ALLOCATION	資材割当	在庫資材割当
INV_CONSIGNED_DIAGNOSTICS	預託在庫診断	仕入先機能からの預託在庫の設定データにおける誤りを識別し伝達するユーティリティのセット。

ビジネス・エンティティ・コード	表示名	説明
INV_COUNT	資材カウント	資材カウント
INV_IC_TRANSACTION_FLOW	在庫会社間請求取引フロー	在庫の会社間請求書を生成する取引の実行です。
INV_IC_TRANSACTION_FLOW_SETUP	会社間在庫取引フロー設定	会社間在庫取引フロー設定
INV_LOT	在庫ロット	在庫ロット
INV_MATERIAL_TRANSACTION	資材取引	在庫資材取引
INV_MOVEMENT_STATISTICS	移動統計	2 国間の国境を越える資材の移動に関連する統計。
INV_MOVE_ORDER	資材移動オーダー	倉庫またはその他の施設内での、ある場所から別の場所への在庫の物理的な移動。組織間の在庫の移送は含みません。
INV_ONHAND	在庫手持残高	在庫手持残高
INV_ORGANIZATION_SETUP	在庫組織設定	在庫組織設定
INV_PICK_RELEASE_PUB	在庫ピック・リリース	ピック・リリースのサポートにおける在庫割当
INV_POSITION	在庫位置	特定の期間バケットにおける受入数、出庫数および期末残高などの在庫組織階層の手持残高を示します。

ビジネス・エンティティ・コード	表示名	説明
INV_REPLENISHMENT	在庫補充	在庫資材補充
INV_RESERVATION	資材予約	在庫資材予約
INV_SALES_ORDERS	在庫受注	Oracle Order Management またはサード・パーティの受注管理システムのオーダーを参照した在庫受注の追跡を示します。
INV_SERIAL_NUMBER	在庫シリアル番号	在庫シリアル番号
INV_SUPPLIER_CONSIGNED_INV	仕入先預託在庫	物理的には在庫組織に存在するが、仕入先が所有している商品。
INV_UNIT_OF_MEASURE	単位	在庫単位
IPM_DOCUMENT	イメージング文書	E-Business Suite における取引の入力および実行を容易にする電子的な文書。
IRC_AGENCY	人材採用エージェン ト	必要要員を採用する権限がある第三者代理店。
IRC_CANDIDATE_NOTIFY_PREFS	候補者採用の通知作 業環境	候補者採用の通知作業環境
IRC_CANDIDATE_SAVED_SEARCH	候補者採用の保存済 検索	候補者採用の保存済検索
IRC_CANDIDATE_WORK_PREFERENCES	候補者採用の勤務の 優先事項	候補者採用の勤務の優先事項
IRC_DEFAULT_JOB_POSTING	デフォルトのジョブ掲 示	デフォルトのジョブ掲示
IRC_JOB_BASKET	ジョブ・バスケット	ジョブ・バスケット

ビジネス・エンティティ・コード	表示名	説明
IRC_JOB_OFFER	内定	採用候補者に提供されるジョブの詳細を含みます。
IRC_JOB_OFFER_LETTER_TEMPLATE	内定通知テンプレート	内定通知のテンプレートです。
IRC_JOB_OFFER_NOTES	内定備考	内定に関する備考
IRC_JOB_POSTING	ジョブ掲示	ジョブ掲示
IRC_JOB_SEARCH_LOCATION	ジョブ検索場所	候補者採用の保存済検索または候補者採用の勤務の優先事項の場所を含みます。
IRC_JOB_SEARCH_PROF_AREA	ジョブ検索専門分野	候補者採用の保存済検索または候補者採用の勤務の優先事項に関する専門分野を含みます。
IRC_NOTIFICATION	iRecruitment Notification	採用担当者、面接官および候補者に送信される通知。
IRC_RECRUITING_3RD_PARTY_SITE	採用協力会社サイト	採用協力会社サイト
IRC_RECRUITING_DOCUMENT	採用文書	採用文書
IRC_RECRUITING_SITE	採用サイト	採用サイト
IRC_RECRUITING_TEAM	採用チーム	採用チーム
IRC_RECRUITMENT_CANDIDATE	採用候補者	採用候補者

ビジネス・エンティティ・コード	表示名	説明
IRC_VACANCY_CONSIDERATION	必要要員の考慮	必要要員の考慮
JE_ES_WHT	スペイン源泉徴収課税取引	スペイン源泉徴収課税取引には、買掛管理および他の外部ソースからの源泉徴収課税取引が保存されます。
JL_BR_AP_BANK_COLLECTION_DOC	ブラジル買掛/未払金銀行回収文書	ブラジル買掛/未払金銀行回収文書
JL_BR_AR_BANK_RETURN_DOC	ブラジル売掛/未収金銀行戻し文書	ブラジル売掛/未収金銀行戻し文書
JTA_BUSINESS_RULE	ビジネス・ルール	エスカレーションまたは自動通知用のビジネス・ルール
JTA_ESCALATION	顧客エスカレーション管理	サービス要求やタスクなどの主なビジネス・エンティティの顧客エスカレーションを管理します。
JTF_RS_DYNAMIC_GROUP	リソース・グループ(動的)	動的リソース・グループ(動的 SQL 文を使用して定義)
JTF_RS_DYNAMIC_GROUP	リソース動的グループ	動的リソース・グループ(動的 SQL 文を使用して定義)
JTF_RS_GROUP	リソース・グループ	個別リソースのグループ化
JTF_RS_GROUP_MEMBER	リソース・グループ・メンバー	グループ内のメンバー
JTF_RS_GROUP_MEMBER_ROLE	リソース・グループ・メンバー役割	グループ内のメンバーに割り当てられた役割
JTF_RS_GROUP_RELATION	リソース・グループ階層	リソース・グループ階層要素

ビジネス・エンティティ・コード	表示名	説明
JTF_RS_GROUP_USAGE	リソース・グループ使用	様々なアプリケーションでのリソース・グループの機能的使用
JTF_RS_RESOURCE	個別リソース	個別リソース
JTF_RS_RESOURCE_AVAILABILITY	リソース可用性	個別リソースが、現在作業割当で使用可能かどうか(Yes/No)。
JTF_RS_RESOURCE_LOV	リソース	人的および非人的リソース
JTF_RS_RESOURCE_SKILL	リソース・スキル	作業割当用のリソース・スキルセット
JTF_RS_RESOURCE_SKILL_LEVEL	リソース・スキル・レベル	様々なスキルの初心者、上級者、中級者を示すスキル・レベル
JTF_RS_ROLE	人的リソース役割	個別のリソースに割り当てられた役割
JTF_RS_ROLE_RELATION	人的リソース役割階層	個別リソース、グループおよびグループ・メンバーに関連付けられた役割の階層。
JTF_RS_SALESREP	営業担当	企業の営業部署を表す個別リソース。
JTF_RS_SALES_GROUP_HIERARCHY	営業グループ階層	営業グループ階層
JTF_RS_SRP_TERRITORY	営業担当テリトリ	営業担当に割り当てられたテリトリ。 注意: テリトリ・マネージャと混同しないでください。

ビジネス・エンティティ・コード	表示名	説明
JTF_RS_TEAM	リソース・チーム	チームは、個人およびグループの集合を表します。
JTF_RS_TEAM_MEMBER	リソース・チーム・メンバー	個人およびグループを含む、チームのメンバー。
JTF_RS_TEAM_USAGE	リソース・チーム使用	様々なアプリケーションでのリソース・チームの機能的使用を表します。
JTF_RS_UPDATABLE_ATTRIBUTE	リソースの更新可能属性	変更が許可された個別リソース情報。
JTF_RS_WF_EVENT	リソース・ビジネス・イベント	ワークフロー・ビジネス・イベントを発生させる、リソース・マネージャでの処理。
JTF_RS_WF_ROLE	リソース・ワークフロー・ロール	個別、グループおよびチームの各リソースを表すワークフロー・ロールです。
JTF_RS_WF_USER_ROLE	リソース・ワークフロー・ユーザー・ロール	リソース・ロール、グループ・メンバーまたはチーム・メンバーを表すワークフロー・ユーザー・ロールです。
JTH_INTERACTION	顧客対応	顧客パーティとの通信または通信の試行です。
JTH_INTERACTION_ACTIVITY	顧客対応活動	顧客パーティの対応中に発生したビジネス・イベント。
JTH_INTERACTION_MEDIA	顧客対応メディア	顧客対応で使用した通信の詳細を含みます。電話、Eメール、Web など。

ビジネス・エンティティ・コード	表示名	説明
JTY_TERRITORY	テリトリ	営業担当、サービス技術者および回収エージェントのテリトリ
LSH_APPLICATIONAREA	アプリケーション・エリア	ビジネス・アプリケーションを定義するオブジェクトのコレクション。ビジネスエリア、データマート、ロードセット、プログラム、レポートセット、表、ワークフローなどです。
LSH_BUSINESSAREA	ビジネス・エリア	ビジネス・エリアは外部ビジュアル化システム(EVS)のインタフェースとして機能します。
LSH_DATAMART	データ・マート	データ・マートは取引システムからエクスポートされたデータを保存し、通常は分析目的で使用されます。
LSH_DOMAIN	生命科学データ・ハブ・ドメイン	アプリケーションエリアを所有する最上位レベルのコンテナでライブラリのオブジェクト定義を格納。定義はビジネスエリア、データマート、ロードセット、プログラムなど。
LSH_EXECUTIONSETUP	生命科学データ・ハブ実行設定情報	各 LSH 実行可能オブジェクトインスタンス(プログラム, RS, ロードセット, ワークフローなど)のコンポーネントである定義済オブジェクト。実行可能オブジェクトの実行を制御。
LSH_EXECUTION_FWK	生命科学データ・ハブ実行ジョブ	環境を提供し、オブジェクトを実行するためのルールを設定するエンティティ。たとえばジョブ・サブミッション API、ジョブ・ログ API などです

ビジネス・エンティティ・コード	表示名	説明
LSH_GENERIC_OBJECT	生命科学データ・ハブ汎用 オブジェクト	生命科学データ・ハブ汎用 オブジェクト
LSH_LOADSET	外部データ・ロード実行	ロード・セットは実行可能オ ブジェクトで、外部システム からのデータのロードに使 用されるプログラムのような 構成の構成と動作を定義す るのに使用します。
LSH_MAPPING	生命科学データ・ハブ列 マッピング	マッピングは、表のようなオ ブジェクトとビューのようなオ ブジェクトの間での列レベル のマッピングを定義します。
LSH_OBJ_CLASSIFICATION	オブジェクト分類	オブジェクトを分類するた めのカテゴリおよびルールを 提供するエンティティです。
LSH_OBJ_SECURITY	生命科学データハブオブ ジェクト・セキュリティポリ シー	オブジェクトにデータ・セキュ リティを定義し実装するた めのルールのセットを提供する エンティティです。
LSH_OUTPUT	生命科学データ・ハブ出力	レポート・セット出力、デー タ・マート出力、プログラム出 力など実行可能オブジェクト の実行で生成される実際の 出力です。
LSH_PARAMETER	API パラメータ	単純なスカラー変数として機 能し、プログラムやレポート・ セットをはじめとする入力/出 力パラメータなどの変数に 基づく定義済オブジェクトで す。
LSH_PARAMETERSET	パラメータ・セット	相互関連のあるパラメータの コレクション

ビジネス・エンティティ・コード	表示名	説明
LSH_PLANNEDOUTPUT	生命科学データ・ハブ計画出力	計画出力は、LSH オブジェクトが実行された場合の予定出力です。実行可能オブジェクトとともに定義されます。
LSH_PROGRAM	生命科学データ・ハブ・プログラム	このプログラムはメタデータ・オブジェクトで、データのセットの処理/レポートに使用されるプログラムのような構成の構成と動作を定義するために使用されます。
LSH_REPORTSET	レポート・セット	レポート・セットはレポートのグループで、データのセットを同時レポートすることを目的とした階層構造の構成と動作を定義するために使用されます。
LSH_SOURCECODE	ソフトウェア・ソース・コード	ソース・コードは、プログラムの実行時に実行される実際のプログラム・コードです。
LSH_TABLE	メタデータ登録データ・オブジェクト	表のようなオブジェクト (Oracle ビューまたは SAS データセットなど) のメタデータ摘要です。
LSH_UTILITY	生命科学データ・ハブ設定ユーティリティ	ツールおよびユーティリティのセットです。
LSH_VALIDATION	生命科学データ・ハブ検証	アプリケーションでオブジェクトを検証するためのルールを提供するエンティティです。
LSH_VARIABLE	生命科学データ・ハブ変数	LSH パラメータおよび表列のソース定義となる SAS 変数または Oracle 表列に相当する LSH 定義済オブジェクトです。

ビジネス・エンティティ・コード	表示名	説明
LSH_WORKAREA	アプリケーション・ワーク・エリア	アプリケーション・エリア内のコンテナで、LSH スキーマへのリリースおよびインストールをするための関連 LSH 定義オブジェクトを準備する場所を定義者に提供します。
MES_COMPLETION_TRANSACTION	Assembly Completion in MES	Business Entity for Assembly Completion in MES
MES_MATERIAL_TRANSACTION	MES Material Transaction	Business Entity for Material Transaction in MES
MES_MOVE_TRANSACTION	MES Move Transaction	Business Entity for Move Transaction in MES
MES_TIME_ENTRY	MES での時間入力	ショップ型製造実行システムに時間入力レコードをインポートします
MSC_ATP_ENQUIRY	ATP 問合せ	このインタフェースは品目の可用性を確認し、その可用性の状況を戻します。
MSC_FORECAST	サプライ・チェーン予測	このインタフェースはサプライ・チェーン計画の予測を作成します。
MSC_NOTIFY_PLAN_OUTPUT	サプライ・チェーン計画オーダー	サプライ・チェーン計画オーダー
MSC_ON_HAND	サプライ・チェーン計画手持在庫	このインタフェースはサプライ・チェーン計画の手持供給レコードを作成します。
MSC_PLANNING_SUPPLY_DEMAND	コラボレイティブ計画の供給 / 需要	このインタフェースはコラボレイティブ計画における任意の供給/需要レコードの作成に使用されます。

ビジネス・エンティティ・コード	表示名	説明
MSC_PURCHASE_ORDER	サプライ・チェーン計画 発注	このインタフェースはサ プライ・チェーン計画の 発注供給を作成します。
MSC_REQUISITION	サプライ・チェーン計画 購買依頼	サプライ・チェーン計画 の購買依頼
MSC_SALES_ORDER	サプライ・チェーン計画 受注	このインタフェースはサ プライ・チェーン計画の 受注需要を作成します。
MSC_SHIPMENT_NOTICE	サプライ・チェーン計画 事前出荷通知	このインタフェースはサ プライ・チェーン計画の インバウンド移動供給を 作成します。
MSC_WORK_ORDER	サプライ・チェーン計画 作業指示	このインタフェースはサ プライ・チェーン計画の 作業指示供給を作成し ます。
NETTING_BATCH	Netting Batch	ネットティング・バッチは、 買掛/未払金および売掛 /未収金取引のセットで す。
OCM_GET_DATA_POINTS	クレジット・レビュー・ データ・ポイント	指定されたクレジット分 類、レビュー・タイプ、 データ・ポイント・カテゴ リ、サブカテゴリのデー タ・ポイントのリスト(組織 の信用状態のレビュー 対象となる基準項目)で す。
OCM_GET_EXTRL_DECSN_PUB	インポート済クレジット・ スコアおよび勧告	外部ソースからのイン ポート・スコアおよび勧 告
OCM_GUARANTOR_CREDIT_REQUEST	保証人与信要求	保証人与信要求の作成 をユーザーに許可しま す。

ビジネス・エンティティ・コード	表示名	説明
OCM_RECOMMENDATIONS	与信勧告	与信要求のレビュー担当者による勧告/決定です。たとえば、標準勧告は「承認/否認」などです。
OCM_WITHDRAW_CREDIT_REQUEST	与信要求取下げ	与信要求の取下げをユーザーに許可します。
OIE_CREDIT_CARD_TRXN	クレジット・カード取引	クレジット・カード取引
OIE_PCARD_TRXN	調達カード取引	調達カード取引
OIR_REGISTRATION	Self Registration of user	Self Registration of external user of the application
OKC_DELIVERABLE	契約提出物	契約提出物
OKC_LIBRARY_ARTICLE	契約ライブラリ条項	契約ライブラリ条項
OKC_LIBRARY_CLAUSE	契約ライブラリ条項(1)	契約ライブラリ条項(1)
OKC_REPOSITORY_CONTRACT	Repository Contract	機密保持契約や提携契約など、通常の購買または販売のフロー以外を処理する契約。これらの契約は契約リポジトリに格納されます。
OKC_REP_CONTRACT	リポジトリ契約	機密保持契約や提携契約など、通常の購買または販売のフロー以外で処理される契約。これらの契約は契約リポジトリに格納されます。
OKE_CONTRACT	プロジェクト契約	プロジェクト契約

ビジネス・エンティティ・コード	表示名	説明
OKL_ACCOUNT_DISTRIBUTION	リース勘定科目配分	リース勘定科目配分
OKL_ACCOUNT_ID	リース勘定科目	リース勘定科目
OKL_AGREEMENT	リース基本契約	リース基本契約
OKL_ASSET_MANAGEMENT	Asset Management	Manage portfolios and asset returns
OKL_COLLECTION	Collection	Bill, collect cash and manage collections from customers
OKL_COLLECTION_CASE	リース回収事例	リース回収事例
OKL_CONTRACT	リース契約	リース契約
OKL_CONTRACT_LIFECYCLE	Contract Management Lifecycle	契約の改訂、終了および更改を管理します。
OKL_CONTRACT_PARTY	リース契約パーティ	リース契約パーティ
OKL_CONTRACT_PAYMENT	リース契約支払	リース契約支払
OKL_CONTRACT_TERM	リース契約条件	リース契約条件
OKL_DISBURSEMENT	Disbursement	Process manually initiated or automated disbursements
OKL_EXECUTE_FORMULA	リース算式	リース算式
OKL_FINANCIAL_PRODUCT	リース契約金融商品	リース契約で指定された金融商品
OKL_INSURANCE	リース保険	リース保険
OKL_INTEREST	リース利息	リース利息

ビジネス・エンティティ・コード	表示名	説明
OKL_INVESTMENT_PROGRAM	Manage Investment Program	投資者アカウントおよび投資基本契約を管理します。
OKL_LATE_POLICY	リース遅延支払ポリシー	リース遅延支払ポリシー
OKL_LEASE_RATE	リース・レート・セット	リース・レート・セット
OKL_MARKETING_PROGRAM	Marketing Program	内部およびパートナー価格設定プログラムを管理します。
OKL_ORIGINATION	Origination	マスター基本契約を管理し、リースおよびローン契約を作成します。
OKL_REMARKETING	Remarketing	ベンダーおよび第三者への資産販売を管理します。
OKL_RESIDUAL_VALUE	リース残余価値	リース残余価値
OKL_RISK_MANAGEMENT	Risk Management	クレジット、価格設定、承認および保険証書を管理します。
OKL_SALES	Sales	Qualify, quote and manage deal opportunities
OKL_STREAM	リース・ストリーム	リース・ストリーム
OKL_TERMINATION_QUOTE	リース終了見積	リース終了見積
OKL_THIRD_PARTY_BILLING	リース第三者請求	リース第三者請求
OKL_UNDERWRITING	Manage Underwriting	クレジット・アプリケーションおよび明細を管理します。
OKL_VENDOR_RELATIONSHIP	Manage Vendor Relationship	ベンダー・アカウントおよび基本契約を管理します。

ビジネス・エンティティ・コード	表示名	説明
OXS_AVAILABLE_SERVICE	サービス可用性	顧客サービス情報を取得する API。特にサービスの継続期間、顧客に対するサービスの利用可能性、および顧客について受注可能なサービスのリストなど。
OXS_CONTRACT	サービス契約	サービス契約
OXS_COVERAGE	サービス契約カバレッジ	サービス契約カバレッジ・サービス条件
OXS_ENTITLEMENT	サービス契約資格/権利	サービス契約顧客資格サービス
OXS_FULFILLMENTS	定期購読/購買履行スケジュール	受注履行スケジュールは、サブスクリプション契約、つまり、受注の作成とその履行のスケジュールに対して定義されます。
OXS_IMPORT	サービス契約インポート	サービス契約インポートは、取得時契約または継続中契約のデータを外部システムまたはレガシー・システムから Oracle Service Contracts 表にインポートするプロセスです。
OXS_SALES_CREDITS	サービス契約の営業実績	割当/未割当営業実績の割合は、サービス、サブスクリプション、保証および延長保証契約を担当している営業担当に対して定義されます。
OXS_TERMINATE	サービス契約終了	終了事由と現在または先日付の終了日を含む、顧客に対する下位明細、明細または契約全体の終了。

ビジネス・エンティティ・コード	表示名	説明
ONT_SALES_AGREEMENT	受注基本契約	これはある契約期間で指定された金額または数量を受注および配分するために、顧客と仕入先の間でコミットする契約概略のビジネス文書です。
ONT_SALES_ORDER	受注	「受注」は顧客受注情報が含まれるビジネス文書です。このエンティティは様々な Oracle E- Business Suite アプリケーションで使用されます。
OTA_CATALOG_CATEGORY	学習カタログ・カテゴリ	学習カタログ・カテゴリ
OTA_CERTIFICATION	学習認定	一時的および更改可能な認定に加入し、完了する機会を学習者に提供するカタログ・オブジェクト。
OTA_CHAT	学習チャット	学習者およびインストラクタがオンラインでメッセージを交換できる、予定されたライブ・ディスカッション。
OTA_CONFERENCE_SERVER	会議サーバー	会議サーバーは、Oracle Web Conferencing (OWC) に OLM を統合し、オンライン同期クラスを配信します。
OTA_COURSE_PREREQUISITE	コース前提条件	学習者が指定クラスへの登録前に完了する必要があるコースまたはコンピテンス。

ビジネス・エンティティ・コード	表示名	説明
OTA_ENROLLMENT_JUSTIFICATION	学習登録理由	各登録理由およびその関連する優先度レベルにより、登録者がクラスに自動登録される順序を決定できます。
OTA_ENROLLMENT_STATUS_TYPE	学習登録ステータス・タイプ	事前定義済の登録ステータス(受講希望、登録済、出席、登録希望、取消)を示します。
OTA_FINANCE_HEADER	学習財務ヘッダー	クラス、学習者登録またはリソース登録に対する金額のレコードです。
OTA_FINANCE_LINE	学習財務明細	財務ヘッダー内の個別財務取引です。
OTA_FORUM	学習フォーラム	学習者およびインストラクタがディスカッションに関する一般的な学習トピックの投稿に使用する掲示板を表します。
OTA_LEARNER_ENROLLMENT	学習者登録	学習者登録
OTA_LEARNING_ANNOUNCEMENT	学習のお知らせ	学習のお知らせ
OTA_LEARNING_CATALOG_CAT_USE	学習カタログ・カテゴリ使用	学習カタログ・カテゴリ使用
OTA_LEARNING_CLASS	学習クラス	学習クラス
OTA_LEARNING_COURSE	学習コース	学習コース
OTA_LEARNING_CROSS_CHARGE	学習相互賦課の設定	学習相互賦課の設定

ビジネス・エンティティ・コード	表示名	説明
OTA_LEARNING_EXTERNAL	外部学習レコード	内部学習カタログに予定されていない、個人が参加したクラスまたはコース。
OTA_LEARNING_OFFERING	学習講義	学習講義
OTA_LEARNING_OFFER_RES_CHKLIST	学習講義リソース・チェックリスト	学習講義リソース・チェックリスト
OTA_LEARNING_PATH	学習パス	学習パス
OTA_LEARNING_PATH_CATEGORY	学習パス・カテゴリ	学習パス・カテゴリ
OTA_LEARNING_PATH_COMPONENT	学習パス・コンポーネント	学習パス・コンポーネント
OTA_LP_SUBSCRIPTION	学習パス・サブスクリプション	すべての学習パスとそのコンポーネントの加入を含みます。たとえば、評価や適合性照合などからマネージャが作成したカタログ学習パスと学習パスへの加入などです。
OTA_RESOURCE	学習リソース	指定インストラクタまたは特定の教室など、クラスの提供に必要な個人またはオブジェクトです。
OTA_RESOURCE_BOOKING	学習リソース登録	学習リソース登録
OTA_TRAINING_PLAN	研修プラン	研修プラン
OZF_ACCOUNT_PLAN	取引勘定計画	取引計画および販促活動に対する勘定計画

ビジネス・エンティティ・コード	表示名	説明
OZF_BUDGET	営業およびマーケティング予算	販促オファー、マーケティング・キャンペーン、イベントおよびその他のマーケティング活動の予算です。
OZF_CLAIM	取引要求	販促要求、破損、輸送エラーなどの、顧客が資金を求めることができる要求。
OZF_EXTRACT	会計抽出	第三者の一般会計の会計抽出。見越計上、調整および販促要求精算の勘定科目、金額、顧客、製品についての詳細を含みます。
OZF_INDIRECT_SALES	間接売上	店頭販売データやチャージバックなど。
OZF_OFFERS	販促オファー	仕入先の営業またはマーケティング組織から顧客に提供される販促オファーまたは値引。
OZF_QUOTA	取引計画クォータ	取引計画および販促活動のクォータおよびターゲット
OZF_SOFT_FUND	パートナ資金	パートナ資金要求
OZF_SPECIAL_PRICING	特別価格設定	特別価格設定要求
OZF_SSD_BATCH	Supplier Ship and Debit Batch	仕入先出荷およびデビット・バッチとは販売業者が仕入先に承認と支払を求めるために発行する要求のことです。バッチには、仕入先が支払うことが期待される見越計上額が含まれます。

ビジネス・エンティティ・コード	表示名	説明
OZF_SSD_REQUEST	Supplier Ship and Debit Request	販売業者が仕入先に特別価格を要求することを認める取り決めです。出荷およびデビット要求は、値引き、製品および期間に関連した数量制限に関する指定を許可します。
PAY_BALANCE	給与バランス	給与バランス
PAY_BALANCE_ADJUSTMENT	給与バランス調整	給与バランス調整
PAY_BATCH_ELEMENT_ENTRY	HRMS バッチ・エレメント・エントリ	HRMS バッチ・エレメント・エントリ
PAY_CONTRIBUTION_USAGE	給与拠出金使用	給与拠出金使用
PAY_COST_ALLOCATION	給与原価割当	給与原価割当
PAY_DEFINED_BALANCE	給与定義済バランス	給与定義済バランス
PAY_ELEMENT	HRMS エレメント	HRMS エレメント
PAY_ELEMENT_CLASSIFICATION	HRMS 分類	支給項目、控除項目および情報など、HRMS エレメントのカテゴリを記述します。これらは後続の処理に影響します。
PAY_ELEMENT_ENTRY	HRMS エレメント・エントリ	HRMS エレメント・エントリ
PAY_ELEMENT_LINK	HRMS エレメント適格基準	HRMS エレメント適格基準
PAY_EMP_TAX_INFO	従業員の税情報	従業員の税情報

ビジネス・エンティティ・コード	表示名	説明
PAY_FORMULA_RESULT	給与処理結果ルール	給与処理における Formula の処理方法および結果の使用方法を示します。
PAY_ITERATIVE_RULE	給与反復ルール	給与反復ルール
PAY_LEAVE_LIABILITY	休暇債務	休暇債務処理で利用される休暇タイプおよび関連定義を記述します。
PAY_ORG_PAYMENT_METHOD	組織支払方法	組織が従業員報酬用使用する給与支払方法です。
PAY_PAYMENT_ARCHIVE	給与支払アーカイブ	給与支払アーカイブ
PAY_PAYROLL_DEFINITION	給与定義	給与定義
PAY_PAYROLL_EVENT_GROUP	給与イベント解釈グループ	給与イベント解釈グループ
PAY_PAYROLL_TABLE_REC_EVENT	給与表記録可能イベント	給与表記録可能イベント
PAY_PERSONAL_PAY_METHOD	個人支払方法	個人支払方法
PAY_PROVINCIAL_MEDICAL	州医療口座	州医療口座
PAY_RUN_TYPE	給与計算タイプ	給与計算タイプ
PAY_TIME_DEFINITION	給与時間	期間およびその使用方法に関する情報を保持します。
PAY_USER_DEFINED_TABLE	HRMS ユーザー定義表	HRMS ユーザー定義表

ビジネス・エンティティ・コード	表示名	説明
PAY_WORKERS_COMPENSATION	労働者補償	労働者補償
PA_AGREEMENT	プロジェクト顧客契約	プロジェクト顧客契約
PA_BILLING_EVENT	プロジェクト請求イベント	プロジェクト請求イベント
PA_BUDGET	プロジェクト予算	プロジェクト予算
PA_BURDEN_COST	プロジェクト総原価	プロジェクト総原価
PA_CAPITAL_ASSET	プロジェクト資産	プロジェクト資産
PA_CUSTOMER_INVOICE	プロジェクト顧客請求書	プロジェクト顧客請求書
PA_EXPENDITURE	プロジェクト支出	プロジェクト支出
PA_EXPENSE_RPT_COST	プロジェクト経費精算書原価	プロジェクト経費精算書原価
PA_FINANCIAL_TASK	プロジェクト財務タスク	プロジェクト財務タスク
PA_FORECAST	プロジェクト予測	プロジェクト予測
PA_IC_TRANSACTION	プロジェクト相互賦課	プロジェクト相互賦課
PA_INTERCOMPANY_INVOICE	プロジェクト会社間請求	プロジェクト会社間請求
PA_INTERPROJECT_INVOICE	プロジェクト・プロジェクト間請求	プロジェクト・プロジェクト間請求
PA_INVENTORY_COST	プロジェクト在庫原価	プロジェクト在庫原価
PA_INVOICE	プロジェクト請求書	プロジェクト請求書
PA_LABOR_COST	プロジェクト労務費	プロジェクト労務費
PA_MISCELLANEOUS_COST	プロジェクトその他原価	プロジェクトその他原価

ビジネス・エンティティ・コード	表示名	説明
PA_PAYABLE_INV_COST	プロジェクト仕入先原価	プロジェクト仕入先原価
PA_PERF_REPORTING	プロジェクト報告	プロジェクト報告
PA_PROJECT	プロジェクト	プロジェクト
PA_PROJ_COST	プロジェクト原価	プロジェクト原価
PA_PROJ_DELIVERABLE	プロジェクト成果物	プロジェクト成果物
PA_PROJ_FUNDING	プロジェクト資金	プロジェクト資金
PA_PROJ_PLANNING_RESOURCE	プロジェクト計画資源	プロジェクト計画資源
PA_PROJ_RESOURCE	プロジェクト資源	プロジェクト資源
PA_RES_BRK_DWN_STRUCTURE	プロジェクト資源分解構造	プロジェクト資源分解構造
PA_REVENUE	プロジェクト収益	プロジェクト収益
PA_TASK	プロジェクト・タスク	プロジェクト・タスク
PA_TASK_RESOURCE	プロジェクト・タスク資源	プロジェクト・タスク資源
PA_TOT_BURDENED_COST	プロジェクト合計総原価	プロジェクト合計総原価
PA_USAGE_COST	プロジェクト資産使用原価	プロジェクト資産使用原価
PA_WIP_COST	プロジェクト仕掛原価	プロジェクト仕掛原価
PA_WORKPLAN_TASK	プロジェクト作業計画タスク	プロジェクト作業計画タスク
PER_APPLICANT	応募者	応募者
PER_APPLICANT_ASG	応募者アサイメント	応募者アサイメント
PER_APPRAISAL	就業者評価	就業者評価

ビジネス・エンティティ・コード	表示名	説明
PER_APPRAISAL_PERIOD	評価期間	業績考課プランで使用される評価期間情報を定義します。
PER_ASSESSMENT	就業者査定	就業者査定
PER_BF_BALANCE	Oracle 以外の給与システムのバランス	Oracle 以外の給与システムのバランス
PER_BF_PAYROLL_RESULTS	Oracle 以外の給与システム結果	Oracle 以外の給与システム結果
PER_CHECKLIST	個人タスク・チェックリスト	新規採用チェックリストなど、コピーが可能なタスクを含み、そのコピーしたタスクが従業員、派遣就業者または応募者に割当て可能なチェックリストです。
PER_COLLECTIVE_AGREEMENT	労働協約	労働協約
PER_COLLECTIVE_AGREEMENT_ITEM	労働協約項目	労働協約項目
PER_COMPETENCE	コンピテンス	コンピテンス
PER_COMPETENCE_ELEMENT	コンピテンス・エレメント	コンピテンス・エレメント
PER_COMPETENCE_RATING_SCALE	コンピテンス・レート・スケール	コンピテンス・レート・スケール
PER_CONFIG_WORKBENCH	HCM 構成ワークベンチ	事業所、ビジネス・グループ、役職および職階などのエンティティを設定する、企業体系構成ワークベンチ・ウィザードを管理します。
PER_CONTACT_RELATIONSHIP	続柄	続柄

ビジネス・エンティティ・コード	表示名	説明
PER_CWK	派遣就業者	派遣就業者
PER_CWK_ASG	派遣就業者アサイメント	派遣就業者アサイメント
PER_CWK_RATE	派遣就業者アサイメント・レート	派遣就業者アサイメント・レート
PER_DISABILITY	障害	障害
PER_DOCUMENTS_OF_RECORD	Documents of Record	従業員、派遣就業者、応募者または担当者のレコードの文書
PER_EMPLOYEE	従業員	従業員
PER_EMPLOYEE_ABSENCE	従業員不就業	従業員不就業
PER_EMPLOYEE_ASG	従業員アサイメント	従業員アサイメント
PER_EMPLOYMENT_CONTRACT	雇用契約	雇用契約
PER_ESTAB_ATTENDANCES	就学した学校および大学	就学した学校および大学
PER_EX-EMPLOYEE	退職者	退職者
PER_GENERIC_HIERARCHY	一般階層	一般階層
PER_GRADE	従業員等級	従業員等級
PER_JOB	役職	役職
PER_JOB_GROUP	役職グループ	役職グループ
PER_MEDICAL_ASSESSMENT	医療認定	医療認定

ビジネス・エンティティ・コード	表示名	説明
PER_OBJECTIVE_LIBRARY	目標ライブラリ	個々に作成されるか、または外部ソースからインポートされる、再使用可能な目標のリポジトリ。
PER_ORGANIZATION_HIERARCHY	組織階層	組織階層
PER_PERFORMANCE_REVIEW	従業員業績レビュー	従業員業績レビュー
PER_PERF_MGMT_PLAN	業績考課管理プラン	実行の期間や作成および評価の各期間など、業績考課管理プロセスのパラメータを示します。
PER_PERSON	HR Person	HR 個人
PER_PERSONAL_CONTACT	個人連絡先	個人連絡先
PER_PERSONAL_SCORECARD	個人スコアカード	業績考課管理プラン用の各就業者の目標で、目標設定、業績レビューおよびスコアリング基準を提供します。
PER_PERSON_ADDRESS	個人住所	個人住所
PER_PHONE	電話	電話
PER_POSITION	職階	職階
PER_POSITION_HIERARCHY	職階階層	職階階層
PER_PREVIOUS_EMPLOYMENT	前職	前職
PER_QUALIFICATION	個人資格	個人資格

ビジネス・エンティティ・コード	表示名	説明
PER_RECRUITMENT_ACTIVITY	採用活動	採用活動
PER_SALARY_PROPOSAL	給与条件提示	給与条件提示
PER_SALARY_SURVEY	給与サーベイ	給与サーベイ
PER_SCORECARD_SHARING	スコアカード・アクセス	スコアカード所有者にアクセス権が付与されたスコアカードに対する、個人およびアクセス権限のリストを保持します。
PER_SECURITY_PROFILE	セキュリティ・プロファイル	セキュリティ・プロファイル
PER_SUPPLEMENTARY_ROLE	HR 補足ロール	HR 補足ロール
PER_VACANCY	必要要員	必要要員
PER_VACANCY_REQUISITION	必要要員要件	必要要員要件
PER_WORK_COUNCIL_ELECTION	労使協議会選任	労使協議会選任
PER_WORK_INCIDENT	勤務上の事故	勤務上の事故
PN_CUSTOMER_SPACE_ASSIGNMENT	顧客空間割当	顧客空間割当
PN_EMPLOYEE_SPACE_ASSGNMENT	従業員空間割当	従業員空間割当
PN_INDEX_LEASES	指数賃料	指数賃料基本契約: Oracle Projects の指数賃料の作成および更新を管理します。

ビジネス・エンティティ・コード	表示名	説明
PN_LEASE	リース	リース基本契約: Oracle Projects のリースおよび支払条件の作成と更新について記述しています。
PN_PROPERTY	空間	建物、土地区画、フロアまたは事務所など、不動産または不動産の構成要素。
PN_RECOVERABLE_EXPENSE	プロパティ回収可能経費	プロパティ回収可能経費
PN_VARIABLE_RENTS	変動賃料	変動賃料基本契約: 賃料の作成および更新を定義し、変動賃料計算を上書きし、分岐点、制約、引当金、減額を作成する他に、変動賃料の期間を生成します。
PN_VOLUME_HISTORY	変動賃料出来高履歴	変動賃料出来高履歴
PJM_INVENTORY	Project Manufacturing 在庫	あるプロジェクトから別のプロジェクトへ、または共通在庫からプロジェクト在庫への永続的および一時的な転送を処理する場合に、プロジェクト別に追跡される在庫です。
PO_ACKNOWLEDGEMENT	発注確認	発注確認
PO_ADVANCED_SHIP_NOTFN	事前出荷通知	事前出荷通知
PO_APPROVAL	発注承認	発注承認
PO_APPROVAL_HIERARCHY	発注承認階層	発注承認階層
PO_APPROVED_SUPPLIER_LIST	承認済仕入先リスト	承認済仕入先リストは、品目または商品、仕入先、販売業者または製造業者、納入先組織、承認/認証ステータスの間の関係を保持します。

ビジネス・エンティティ・コード	表示名	説明
PO_ATTACHMENTS	調達添付	調達添付
PO_AUCTION	オークション	オークション
PO_AWARD	落札	落札
PO_BIDDING_ATTRIBUTES	入札属性	入札属性
PO_BLANKET_PURCHASE_AG REEMENT	包括購買契約	包括購買契約
PO_BLANKET_RELEASE	購買包括リリース	包括リリースは、実際の発注の ために包括購買契約に対して 発行されます。
PO_CLM_AWARD	CLM 取得	契約ライフサイクル管理の取得 文書は、商品およびサービスを 提供するための、政府と仕入先 の間での契約の概略を示しま す。
PO_CLM_IDV	CLM 不確定納期手段	契約ライフサイクル管理の不確 定納期手段は、特定の期間で の商品およびサービスの未定 義の戦略的要件を識別しま す。
PO_CATALOG	購買カタログ	購買カタログ
PO_CATALOG_CATEGORY	購買カタログ・カテゴリ	購買カタログ・カテゴリ
PO_CHANGE	発注変更	発注変更
PO_CONSUMPTION_ADVICE	預託在庫消費通知	預託消費に対するリリースまた は標準発注
PO_CONTRACT	購買契約	購買契約

ビジネス・エンティティ・コード	表示名	説明
PO_CONTRACT_PURCHASE_AGREEMENT	購買契約(1)	購買契約(1)
PO_CONTRACT_TEMPLATE	購買契約テンプレート	購買契約テンプレート
PO_CONTRACT_TERM	購買契約条件	購買契約条件(条項、提出物および契約文書)
PO_DOCUMENT_APPROVER	購買文書承認者	購買文書承認者
PO_EXPENSE_RECEIPT	経費領収書	経費領収書
PO_GLOBAL_BLANKET_AGREEMENT	グローバル包括購買契約	グローバル包括購買契約
PO_GLOBAL_CONTRACT_AGREEMENT	グローバル購買契約	グローバル購買契約
PO_GOODS_RECEIPT	商品受入	商品受入
PO_GOODS_RETURN	商品返品	商品返品
PO_INTERNAL_REQUISITION	社内購買依頼	社内購買依頼
PO_NEGOTIATION	ソーシング・ネゴシエーション	ソーシング・ネゴシエーション
PO_PLANNED_PURCHASE_ORDER	計画発注	計画発注
PO_PLANNED_RELEASE	計画 PO リリース	計画 PO リリース
PO_PRICE_BREAKS	ソーシング価格分岐	ソーシング価格分岐

ビジネス・エンティティ・コード	表示名	説明
PO_PRICE_DIFFERENTIAL	購買価格差異	購買価格差異には、購買依頼明細用のレートに基づく詳細、購買オーダー明細、またはエンティティ・タイプに基づく包括価格値引の価格差異が保持されます。
PO_PRICE_ELEMENTS	ソーシング価格要素	ソーシング価格要素
PO_PURCHASE_REQUISITION	購買依頼	購買依頼
PO_QUOTE	ソーシング見積	ソーシング見積
PO_RECEIPT_CORRECTION	受入訂正	受入訂正
PO_RECEIPT_TRAVELER	受入伝票	受入伝票
PO_REQUISITION_APPROVAL	購買依頼承認	購買依頼承認
PO_REQ_APPROVAL_HIERARCHY	購買依頼承認階層	購買依頼承認階層
PO_RFI	情報の要求	情報の要求
PO_RFQ	見積依頼	見積依頼
PO_RFQ_RESPONSE	見積依頼の応答	見積依頼の応答
PO_SERVICES_RECEIPTS	サービス受入	サービス受入
PO_SHIPMENT_AND_BILLING_NOTICE	出荷/請求通知	出荷/請求通知
PO_SOURCING_BID	ソーシング入札	ソーシング入札
PO_SOURCING_RULES	ソーシング・ルール	ソーシング・ルール
PO_SOURCING_RULE_ASSIGNMENTS	ソーシング・ルール割当	ソーシング・ルール割当

ビジネス・エンティティ・コード	表示名	説明
PO_STANDARD_PURCHASE_ORDER	標準発注	標準発注
PO_SUPPLIER_BANK_ACCOUNT	仕入先銀行口座	仕入先銀行口座
PQH_ADDITIONAL_SECOND_PENSION	追加第二年金	追加第二年金
PQH_DEFAULT_HR_BUDGET_SET	デフォルト HR 予算計画 セット	デフォルト HR 予算計画 セット
PQH_EMEA_SENIORITY_SITUATION	ヨーロッパの年功状況	ヨーロッパの年功状況
PQH_EMPLOYEE_ACCOMMODATION	従業員宿泊施設	従業員宿泊施設
PQH_EMPLOYER_ACCOMMODATION	事業主提供宿泊施設	事業主提供宿泊施設
PQH_FR_CORPS	フランスの部隊	フランスの部隊
PQH_FR_SERVICES_VALIDATION	フランスの勤続検証	フランスの勤続検証
PQH_FR_STATUTORY_SITUATION	フランスの法定状況	フランスの法定状況
PQH_GLOBAL_PAY_SCALE	グローバル支給スケール	グローバル支給スケール
PQH_POS_CTRL_BUSINESS_RULE	職階管理ビジネス・ルール	職階管理ビジネス・ルール
PQH_POS_CTRL_ROUTING	職階管理ルーティング	職階管理ルーティング
PQH_POS_CTRL_TRANS_TEMPLATE	職階管理トランザクション・テンプレート	職階管理トランザクション・テンプレート

ビジネス・エンティティ・コード	表示名	説明
PQH_RBC_RATE_MATRIX	個人適格基準レート・マトリックス	レート・マトリックスには、様々な基準値の組合せ、および個人の値が基準値と一致した場合にその個人が適格であるレートが収納されています。
PQH_REMUNERATION_REGULATION	報酬規約	報酬規約
PQH_WORKPLACE_VALIDATION	職場検証プロセス	職場検証
PQP_PENSION_AND_SAVING_TYPE	年金および貯蓄タイプ	年金および貯蓄タイプ
PQP_VEHICLE_ALLOCATION	車両割当	車両割当
PQP_VEHICLE_REPOSITORY	車両リポジトリ	車両リポジトリ
PRP_PROPOSAL	営業提案	営業提案
PSP_EFF_REPORT_DETAILS	従業員作業レポート	ある期間での従業員の労力の配分を要約します。管理と予算ガイドラインの業務に即した労務費用の分配が正確であることの確認に使用されます。
PV_OPPORTUNITY	パートナー商談割当	パートナーへの間接的な商談の割当をサポートします。
PV_PARTNER_PROFILE	パートナー・プロファイリング	パートナーおよびその担当者についての追加情報を収集する拡張可能な属性モデルです。

ビジネス・エンティティ・コード	表示名	説明
PV_PROGRAM	パートナ・プログラム管理	パートナ・プログラムの作成/保守およびそのプログラムへの関連パートナの登録/メンバーシップなど、パートナ・プログラム管理のフレームワークを表します。
PV_REFERRAL	パートナ・ビジネス参照	パートナはベンダーとのビジネス参照のために参照を作成します。参照が販売となった場合パートナは報酬を受けます。パートナは非競争の目的でベンダーとの取引を登録します。
QA_PLAN	品質収集計画	品質収集計画
QA_RESULT	品質結果	直接に取引または収集インポートから収集された収集計画の結果データを示します。
QA_SPEC	品質仕様	顧客または仕入先に固有の品目または品目カテゴリの特性の要件。
QOT_QUOTE	Sales Quote	営業見積
QP_PRICE_FORMULA	価格算式	価格算式
QP_PRICE_LIST	価格表	価格表
QP_PRICE_MODIFIER	価格モディファイア	価格モディファイア
QP_PRICE_QUALIFIER	価格クオリファイア	価格クオリファイア
REPAIR_ORDER	修理オーダー	修理オーダー

ビジネス・エンティティ・コード	表示名	説明
RLM_CUM	仕入先出荷の累積	CUM 管理設定に基づき、特定の顧客品目に対する仕入先で実行された出荷合計の追跡に使用されます。
RLM_SCHEDULE	顧客需要計画	顧客の生産資材リリースを意味します。
RRS_SITE	サイト	実際の、または計画中の構造あるいは構造のセット(建物、ビジネス・パーク、通信塔、高速道路またはモニュメントなど)の空間的な場所を示します。
SOA_DIAGNOSTICS	統合 SOA ゲートウェイの診断	Oracle E-Business Suite Integrated SOA Gateway の診断
UMX_ACCT_REG_REQUESTS	ユーザー・アカウント要求	ユーザー・アカウントに対して実行される要求を表し、システムへのアクセスに必要です。
UMX_ROLE	セキュリティ役割	セキュリティの一連の権限を表します。ロールはユーザーに割り当てられ、ロール継承階層で定義されます。職責は特別なタイプのロールです。
UMX_ROLE_REG_REQUESTS	セキュリティ役割要求	システムの保護された部分へアクセスをするための、(セキュリティ・システムで定義された)役割に対する要求を表します。

ビジネス・エンティティ・コード	表示名	説明
WF_APPROVALS_SERVICES	承認データ・サービス	クライアント・アプリケーションによって起動され、 Oracle Workflow 承認要約と詳細を取得し、承認処理を実行するサービスを提供します。
WF_ENGINE	ワークフロー項目	プロセス、機能、通知およびイベント活動を含むワークフロー項目を示します。
WF_EVENT	ビジネス・イベント	ビジネス・イベント
WF_NOTIFICATION	ワークフロー通知	ワークフロー通知
WF_USER	ワークフロー・ディレクトリ・ユーザー	ワークフロー・ディレクトリ・ユーザー
WF_WORKLIST	ワークフロー・ワークリスト内容	ワークフロー・エンティティ (経費精算書、PO 要求、HR オファー、HR 必要要員)の承認
WIP_ACCOUNTING_CLASS	WIP 会計区分	WIP 会計区分
WIP_COMPLETION_TRANSACTION	WIP Assembly Completion	WIP での組立完了用ビジネス・エンティティ
WIP_EMPLOYEE_LABOR_RATE	WIP 従業員作業レート	WIP 従業員作業レート
WIP_MATERIAL_TRANSACTION	WIP Material Transaction	WIP での資材取引用ビジネス・エンティティ
WIP_MOVE_TRANSACTION	WIP 製造現場移動	WIP 製造現場移動
WIP_PARAMETER	仕掛の設定	仕掛の設定
WIP_PRODUCTION_LINE	生産ライン	生産ライン

ビジネス・エンティティ・コード	表示名	説明
WIP_REPETITIVE_SCHEDULE	ライン型製造オーダー	ライン型製造オーダー
WIP_RESOURCE_TRANSACTION	WIP 生産資源処理フロー	WIP 生産資源取引
WIP_SCHEDULE_GROUP	WIP 計画グループ	WIP 計画グループ
WIP_SHOPFLOOR_STATUS	製造現場ステータス	製造現場ステータス
WIP_WORK_ORDER	作業指示	製造オーダー/作業指示
WMS_BULK_PICK	倉庫バルク・ピック・カスタム API	このオブジェクトは、次のことを決定するカスタム・メソッドを指定するために使用できます。 - 複数のバルク・タスクを同じ LPN にロードできるかどうか - PJM 使用可能組織にプロジェクトとタスクを横断する LPN を連結できるかどうか
WMS_CONTAINER	倉庫管理ライセンス・プレート	倉庫コンテナおよびライセンス・プレート管理
WMS_DEPLOY	Warehouse Management System 配置チェック	WMS 配置モードおよび関連スタンドアロンおよび LPN インストール・ユーティリティを取得するオブジェクト。
WMS_DEVICE_CONFIRMATION_PUB	作業手配タスク	作業手配タスクを更新するステータスを含みます。ASRS タスク、カルーセル・タスク、電光表示によるピッキング・システム・タスクなどです。
WMS_DEVICE_INTEGRATION	倉庫デバイス	倉庫デバイス

ビジネス・エンティティ・コード	表示名	説明
WMS_DOCK_APPOINTMENTS	ドック・アポイントメント	新規ドック・アポイントメントを作成、または既存のドックを変更および削除するためのWMSドック・アポイントメント・オブジェクト。
WMS_EPC_PUB	電気製品コード	GTIN、GID、SSCCなどの電気製品コードを格納します。
WMS_INSTALL	Warehouse Management System インストール・チェック	この API には次の 2 つの目的があります。 - この API は、一部のフラグがフォームで隠されることなく WMS 製品がシステムにインストールされていることを確認します。 - また、この API は、組織で WMS が使用可能であるかを戻します。
WMS_LABEL	ラベル印刷	出荷、パッケージ、コンテナ、品目およびシリアル・ラベルの印刷をサポートする情報が収納されます。
WMS_LICENSE_PLATE	ライセンス・プレート	出荷、倉庫管理および製造現場管理で使用するコンテナ・インスタンスの識別子です。

ビジネス・エンティティ・コード	表示名	説明
WMS_REPLENISHMENT	倉庫補充	このオブジェクトにより、選択した明細の後処理を容易にするために補充コードによって処理される需要明細にアクセスできます。たとえば、一部割当済補充移動オーダーをバックオーダーする場合などです。また、このオブジェクトにより、補充する需要明細のリストを選択し、選択されない需要明細をバックオーダーするカスタム・ロジックも提供できます。
WMS_RFID_DEVICE	倉庫管理 RFID	倉庫管理 RFID 統合
WMS_RULES	倉庫ルール・エンジン	WMS ルール・エンジン実行時に適用されるカスタム・ロジックをサポートするルール・エンジン・オブジェクト。 カスタム戦略検索や、任意の場所で使用可能な生産能力を計算するカスタム・メソッドなどは、カスタム・コードによって処理できます。
WMS_SHIPPING_TRANSACTION	倉庫管理出荷取引	倉庫管理の積荷および出荷
WMS_TASKS	倉庫タスク管理	タスクの問合せ、変更、更新、分割、削除または取消などの倉庫タスク管理のサポート。

ビジネス・エンティティ・コード	表示名	説明
WMS_WAVE_PLANNING	倉庫ウェーブ計画	次のことを目的としてカスタマイズするためのウェーブ計画のサポート。 - 顧客の要件に基づき、作成しているウェーブに対する明細の追加または削除 - カスタム・ロジックに基づくウェーブ例外の発生 - カスタム・ロジックに基づく「未リリース」ステータスのタスクのリリース
WMS_XDOCK	倉庫クロスドック	倉庫クロスドック統合を使用して、クロスドックの方法をカスタマイズできます。 次の目的でカスタム・メソッドを使用します。 - 使用されるクロスドック基準を決定します。 - 予定時間を計算します。 - 予定搬送時間を計算します。 - 供給明細をソートします。 - 需要明細をソートします。
WSH_CONTAINER_PUB	コンテナ	出荷用に商品および資材が積載される船舶。
WSH_DELIVERY	搬送	出荷明細のグループ
WSH_DELIVERY_LINE	搬送明細	出荷明細

ビジネス・エンティティ・コード	表示名	説明
WSH_EXCEPTIONS_PUB	出荷例外	例外は変更数量、OM での出荷取消などで、出荷エンティティに自動的に記録されます。例外の動作は「エラー」、「警告」または「情報のみ」です。
WSH_FREIGHT_COSTS_PUB	運送費	出荷担当者に対する輸送サービスのコスト。たとえば、出荷担当者が輸送サービスに対して運送業者に支払う金額など。
WSH_PICKING_BATCHES_PUB	ピック・リリース	割当およびピッキングのため、倉庫に搬送明細をリリースするプロセス。
WSH_TRIP	トリップ	事業所からの出荷の、計画された出発または取得時出発を記述します。
WSH_TRIP_STOPS_PUB	トリップ/ストップ	商品が荷降または荷積される、トリップの通過地となる物理的な場所。
WSM_INV_LOT_TXN	在庫ロット取引	ロット基準在庫取引
WSM_LOT_BASED_JOB	ロット基準製造オーダー	ロット基準製造オーダー/WIP ロット
WSM_LOT_MOVE_TXN	ロット移動取引	ロット基準製造オーダーの製造現場移動取引
WSM_WIP_LOT_TXN	WIP ロット取引	ロット基準 WIP 取引

ビジネス・エンティティ・コード	表示名	説明
XDP_SERVICE_ORDER	サービス履行オーダー	サービス履行マネージャにより供給される必要がある、1 つ以上のサービスのオーダー。このサービスの供給では多くの場合、Oracle E-Business Suite 以外のシステムが関与します。
XLA_JOURNAL_ENTRY	Subledger Accounting Journal Entry	ヘッダー、明細および配分で構成される補助元帳会計仕訳
XNB_ADD_BILLSUMMARY	請求要約処理	外部請求システムから Oracle E-Business Suite への新規請求要約レコードの挿入、作成または移入に使用されます。
XNB_ADD_GROUPSALORDER	請求システム受注明細グループ	すべての受注明細情報は 1 つの XML メッセージとして生成され、サード・パーティの請求アプリケーションに公開されます。
XNB_ADD_SALORDER	請求システム受注追加	受注情報は XML メッセージとして生成され、サード・パーティの請求アプリケーションに公開されます。
XNB_SYNC_ACCOUNT	請求システム顧客アカウント同期化	アカウント情報は XML メッセージとして生成され、サード・パーティの請求アプリケーションに公開されます。
XNB_SYNC_ITEM	請求システム在庫品目同期化	カタログ情報は XML メッセージとして生成され、サード・パーティの請求アプリケーションに公開されます。
XTR_BANK_BALANCE	銀行口座残高	銀行口座残高
XTR_DEAL_DATA	財務取引	財務取引

ビジネス・エンティティ・コード	表示名	説明
XTR_MARKET_DATA	市場レート	金融市場レート・データ
XTR_PAYMENT	XTR Payment	財務支払は、実行されている支払を表します。
YMS_CUSTOM_LOGIC	ヤード・カスタム・ロジック	カスタム・ロジックは、次のことを行うのに役立ちます。 - ヤード・チェックイン中のチェックイン場所の決定 - 指定された品目の品目単位原価、組織および文書の指定
YMS_DOCK_APPOINTMENTS	ヤード・ドック・アポイントメント	ドック・アポイントメントを作成、変更、問合せまたは削除するためのヤード・ドック・アポイントメント・オブジェクト。
YMS_EQUIPMENT_ITEM	ヤード機器タイプ	ヤード組織で取引できるヤード機器タイプを示します。
YMS_INQUIRY	ヤード照会	ヤード統計、アクセス可能組織、機器の状態、機器積載ステータスなどについて照会します。
YMS_TRANSACTIONS	ヤード取引	ヤード取引は、ヤード組織でのチェックイン、チェックアウト、封印、開封などになります。
ZX_DATA_UPLOAD	インポート済税金コンテンツ	このエンティティ・コードは、Oracle E- Business Suite Tax コンテンツ・アップロード要求セットの全プログラムで使用されます。

例: Create Customer

```

/*#
 *This interface creates a customer.It calls the
 *customer hub API that creates a 'party' to create a
 *party of type 'customer'.
 *@rep:scope public
 *@rep:product OM
 *@rep:displayname Create Customer
 *@rep:category BUSINESS_ENTITY OM_CUSTOMER
 *@rep:lifecycle active
 *@rep:compatibility S
 */

```

コンポジット・サービス - BPEL アノテーション・ガイドライン

この項では、コンポジット・サービス - BPEL の統合リポジトリ・アノテーションについて把握する必要がある内容について説明します。

コンポジット・サービス - BPEL の注釈付け

- BPEL プロジェクトは、*.bpel ファイルで注釈付けする必要があります。
- 注釈付けを行う前に、/*#で始まるコメントが存在しないことを確認してください。スラッシュ、アスタリスク、シャープの文字はリポジトリ・アノテーションの開始を示すために使用するものとなり、通常のコメントで使用すると、エラーまたは望ましくない動作が発生します。
- 注釈付けするには、テキスト・エディタで.bpel ファイルを開いて、ファイルを編集します。
- .bpel ファイルで、ファイルの先頭のコメント・セクション内にアノテーションを配置します。
- ビジネス・イベントが呼び出される条件と、ビジネス・イベントを起動する UI 処理を含む、わかりやすい説明を入力します。
- FND_APPLICATION で製品コードを定義します。
- コンポジット・サービス - BPEL プロセスに対して、既存のビジネス・エンティティを使用します。既存のビジネス・エンティティのリストは、「ビジネス・エンティティ・アノテーション・ガイドライン」(ページ A- 41)を参照してください。
- <BPEL_PROCESS_NAME>.bpel のインタフェース名は、'oracle. apps' + product_code + '.<BPEL_PROCESS_NAME>'として定義する必要があります。
- BPEL プロセス名が BPEL_PROCESS_NAME の場合は、次のようになります
 - BPEL プロセスの Jar ファイルは、<prod>_bpel_<BPEL_PROCESS_NAME>.jar として作成する必要があります。
 - <prod>_bpel_<BPEL_PROCESS_NAME>.jar ファイルの配置先は、\$product_top/patch/115/jar/bpel にする必要があります。

- <prod>_bpel_<BPEL_PROCESS_NAME>.jar ファイルの解凍先は、\$product_top/patch/115/jar/bpel にする必要があります。
- <BPEL_PROCESS_NAME>.jar に対する BPEL ファイルは、\$product_top/patch/115/jar/bpel/<prod>_bpel_<BPEL_PROCESS_NAME>.bpel に配置される必要があります。
- BPEL ファイル名は、<BPEL_PROCESS_NAME>.bpel から変更しないでください。
- <BPEL_PROCESS_NAME>に対する WSDL ファイルは、\$product_top/patch/115/jar/bpel/<prod>_bpel_<BPEL_PROCESS_NAME>.bpel に配置される必要があります。
- WSDL ファイル名は、<BPEL_PROCESS_NAME>wsdl から変更しないでください。
- スタンドアロン・パーサーは、注釈付けされた\$product_top/patch/115/jar/bpel/<prod>_bpel_<BPEL_PROCESS_NAME>.bpel/bpel<BPEL_PROCESS_NAME>.bpel に対して実行する必要があります。
- \$product_top/patch/115/jar/bpel/<prod>_bpel_<BPEL_PROCESS_NAME>.bpel/bpel<BPEL_PROCESS_NAME>_bpel.ildt が統合リポジトリにロードされる必要があります。
- スタンドアロン・パーサーの実行中、*.bpel ファイルのアーカイブ・ファイルの場所は、patch/115/jar/bpel にする必要があります。

コンポジット・サービス - BPEL のアノテーション - 構文

コンポジット・サービス - BPEL のアノテーションは、次のとおりです。

```
/*#
 * This is a bpel file for creating invoice.
 * @rep:scope public
 * @rep:displayname Create Invoice
 * @rep:lifecycle active
 * @rep:product inv
 * @rep:compatibility S
 * @rep:interface oracle.apps.inv.CreateInvoice
 * @rep:category BUSINESS_ENTITY INVOICE_CREATION
 */
```

要素の定義の詳細は、統合リポジトリのアノテーションの一般的なガイドライン(ページ A-1)を参照してください。

必須のアノテーション

次のリンクをたどって、各アノテーションの構文と用途を確認してください。

- 説明文で始める必要あり
- rep:displayname、ページ A-125
- rep:scope、ページ A-123

- rep:product、ページ A-124
- rep:category BUSINESS_ENTITY、A-132 ページ

オプションのアノテーション

- link、ページ A-129
- see、ページ A-130
- rep:lifecycle、ページ A-127
- rep:compatibility、ページ A-128
- rep:ihelp、ページ A-131
- rep:metalink、ページ A-132

テンプレート

コンポジット - BPEL ファイルを注釈付けするときは、次のテンプレートを使用できます。

```
.
.
.
/*#
 * <Put your long bpel process description here
 * it can span multiple lines>
 * @rep:scope <scope>
 * @rep:displayname <display name>
 * @rep:lifecycle <lifecycle>
 * @rep:product <product or pseudoproduct short code>
 * @rep:compatibility <compatibility code>
 * @rep:interface <oracle.apps.[product_code].[bpel_process_name]>
 * @rep:category BUSINESS_ENTITY <entity name>
 */
.
.
.
```

例

次に、注釈付けされたコンポジット - BPEL ファイルの例を示します。

```

////////////////////////////////////
Oracle JDeveloper BPEL Designer

Created: Tue Oct 30 17:10:13 IST 2007
Author:  jdole
Purpose: Synchronous BPEL Process
/*#
 * This is a bpel file for creating invoice.
 * @rep:scope public
 * @rep:displayname Create Invoice
 * @rep:lifecycle active
 * @rep:product PO
 * @rep:compatibility S
 * @rep:interface oracle.apps.po.CreateInvoice
 * @rep:category BUSINESS_ENTITY INVOICE
 */

////////////////////////////////////

-->
<process name="CreateInvoice">
  targetNamespace="http://xmlns.oracle.com/CreateInvoice"
  xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-
process/"
  xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.
tip.pc.services.functions.Xpath20"
  xmlns:ns4="http://xmlns.oracle.
com/pcbpel/adaptor/file/ReadPayload/"
  xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:ns5="http://xmlns.oracle.com/bpel/workflow/xpath"
  xmlns:client="http://xmlns.oracle.com/CreateInvoice"
  xmlns:ns6="http://xmlns.oracle.
com/bpel/services/IdentityService/xpath"
  xmlns:ora="http://schemas.oracle.com/xpath/extension"
  xmlns:ns1="http://xmlns.oracle.
com/soapprovider/plsql/AR_INVOICE_API_PUB_2108/CREATE_SINGLE_INVOICE_10
37 895/"
  xmlns:ns3="http://xmlns.oracle.
com/soapprovider/plsql/AR_INVOICE_API_PUB_2108/APPS/BPEL_CREATE_SINGLE
IN VOICE_1037895/AR_INVOICE_API_PUB-24CREATE_INV/"
  xmlns:ns2="http://xmlns.oracle.com/pcbpel/adaptor/appscontext
/
  xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
  xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.
services.functions.ExtFunc">

<!--
////////////////////////////////////
PARTNERLINKS
      List of services participating in this BPEL process
////////////////////////////////////
-->
<partnerLinks>
  <!--
    The 'client' role represents the requester of this service.It is
    used for callback.The location and correlation information
    associated
    with the client role are automatically set using WS-Addressing.
  -->
  <partnerLink name="client" partnerLinkType="client:CreateInvoice"
    myRole="CreateInvoiceProvider"/>
  <partnerLink name="CREATE_SINGLE_INVOICE_1037895"
    partnerRole="CREATE_SINGLE_INVOICE_1037895_ptt_Role"
    partnerLinkType="ns1:
CREATE_SINGLE_INVOICE_1037895_ptt_PL"/>

```

```

<partnerLink name="ReadPayload" partnerRole="SynchRead_role"
    partnerLinkType="ns4:SynchRead_plt"/>
</partnerLinks>
<!--
////////////////////////////////////
VARIABLES
    List of messages and XML documents used within this BPEL process
////////////////////////////////////
-->
<variables>
<!--Reference to the message passed as input during initiation-->
    <variable name="inputVariable"
        messageType="client:CreateInvoiceRequestMessage"/>
<!--Reference to the message that will be returned to the requester-->
    <variable name="outputVariable"
        messageType="client:CreateInvoiceResponseMessage"/>
    <variable name="
Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
        messageType="ns1:Request"/>
    <variable name="
Invoke_1_CREATE_SINGLE_INVOICE_1037895_OutputVariable"
        messageType="ns1:Response"/>
    <variable name="Invoke_2_SynchRead_InputVariable"
        messageType="ns4:Empty_msg"/>
    <variable name="Invoke_2_SynchRead_OutputVariable"
        messageType="ns4:InputParameters_msg"/>
</variables>
<!--
////////////////////////////////////
ORCHESTRATION LOGIC
    Set of activities coordinating the flow of messages across the
    services integrated within this business process
////////////////////////////////////
-->
<sequence name="main">
    <!--Receive input from requestor.(Note: This maps to operation
defined in CreateInvoice.wsdl)-->
    <receive name="receiveInput" partnerLink="client"
        portType="client:CreateInvoice" operation="process"
        variable="inputVariable" createInstance="yes"/>
    <!--Generate reply to synchronous request-->
    <assign name="SetHeader">
        <copy>
            <from expression="'operations'">
            <to variable="
Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
                part="header"
                query="/ns1:SOAHeader/ns2:ProcedureHeaderType/ns2:Username"
/>
            </copy>
            <copy>
                <from expression="'Receivables, Vision Operations (USA)'">
                <to variable="
Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
                    part="header"
                    query="/ns1:SOAHeader/ns2:ProcedureHeaderType/ns2:
Responsibility"/>
                </copy>
                <copy>
                    <from expression="'204'">
                    <to variable="
Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
                        part="header"
                        query="/ns1:SOAHeader/ns2:ProcedureHeaderType/ns2:ORG_ID"/>
                    </copy>
                <copy>

```

```

<from expression="'Receivables, Vision Operations (USA)'">
  <to variable="
Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
  part="header"
  query="/ns1:SOAHeader/ns1:SecurityHeader/ns1:
ResponsibilityName"/>
  </copy>
</assign>
<invoke name="InvokeReadPayload" partnerLink="ReadPayload"
  portType="ns4:SynchRead_ptt" operation="SynchRead"
  inputVariable="Invoke_2_SynchRead_InputVariable"
  outputVariable="Invoke_2_SynchRead_OutputVariable"/>
<assign name="SetPayload">
  <copy>
    <from variable="Invoke_2_SynchRead_OutputVariable"
      part="InputParameters" query="/ns3:InputParameters"/>
    Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
    part="body" query="/ns1:SOARequest/ns3:InputParameters"/>
  </copy>
</assign>
<assign name="SetDate">
  <copy>
    <from expression="xp20:current-date()">
    <to to variable="
Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
    part="body"
    query="/ns1:SOARequest/ns3:InputParameters/ns3:
P_TRX_HEADER_TBL/ns3:P_TRX_HEADER_TBL_ITEM/ns3:TRX_DATE"/>
  </copy>
</assign>
<invoke name="Invoke_1" partnerLink="CREATE_SINGLE_INVOICE_1037895"
  portType="ns1:CREATE_SINGLE_INVOICE_1037895_ptt"
  operation="CREATE_SINGLE_INVOICE_1037895"
  inputVariable="
Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
  outputVariable="
Invoke_1_CREATE_SINGLE_INVOICE_1037895_OutputVariable"/>
  <assign name="AssignResult">
    <copy>
      <from variable="
Invoke_1_CREATE_SINGLE_INVOICE_1037895_OutputVariable"
      part="body"
      query="/ns1:SOAResponse/ns3:OutputParameters/ns3:
X_MSG_DATA"/>
      <to variable="outputVariable" part="payload"
        query="/client:CreateInvoiceProcessResponse/client:result"/>
    </copy>
  </assign>
  <reply name="replyOutput" partnerLink="client"
    portType="client:CreateInvoice" operation="process"
    variable="outputVariable"/>
</sequence>
</process>

```

アノテーション用語集

この項では、現在サポートされているアノテーション・タイプのリストと、その推奨される使用に関する詳細を取り上げます。

次の表では、<description sentence(s)>のアノテーション情報について説明します。

<description sentence(s)>

アノテーション・タイプ

<description sentence(s)>

構文

タグは必要ありません。

アノテーション・タイプ	<description sentence(s)>
用途	インタフェースまたはメソッドの実行内容のユーザーフレンドリな説明を定義します。 説明は、大文字で始まりピリオドで終わる要約文から開始します。すべてを大文字にしたり、固有名詞でない単語の先頭文字を大文字にしないでください。 適切な開始文の例は次のようになります。 The Purchase Order Data Object holds the purchase order data including nested data objects such as lines and shipments. 一般的に、適切な説明は、複数の文を含み、潜在的顧客が理解しやすい内容です。パッケージ・レベルの説明で詳細なコンテキスト情報が示されるため、(繰返しを避けるために)関連するメソッド・レベルの説明をより簡潔にできる場合は、この複数の文のルールの特例となります。 不適切な例: Create an order. この説明は、ほとんど役立ちません。より適切な例: Use this package to create a customer order, specifying header and line information. 説明内で改行を強制する
タグを使用できます。次に、説明に強制改行を挿入する方法の例を示します。 次に、説明に強制改行を挿入する方法の例を示します。 FEM_BUDGETS_ATTR_T is an interface table for loading and updating Budget attribute assignments using the Dimension Member Loader.
These attribute assignments are properties that further describe each Budget.
When loading Budgets using the Dimension Member Loader, identify each new member in the FEM_BUDGETS_B_T table while providing an assignment row for each required attribute in the FEM_BUDGETS_ATTR_T table.

アノテーション・タイプ	<description sentence(s)>
例	<pre>/*# * This is a sample description. Use * standard English capitalization and * punctuation. Write descriptions * carefully.</pre>
必須	@rep:scope public を扱うすべてのインタフェースに必須です。
デフォルト	設定されていない場合、値は、インタフェースまたはメソッドの Javadoc または PL/SQL 文書からデフォルト設定されます。
レベル	インタフェース(クラス)と API (メソッド)。
複数の許可	いいえ。プログラム要素(クラスまたはメソッド)ごとに 1 つのみ使用します。
コメント	オプションで、説明内に次の HTML タグを使用できます。 <pre><body> <p> <h1> <h2> <h3> <pre> (複数のコード・サンプルの場合) (<code> タグで囲む必要があります)</pre>

次の表では、@rep:scope のアノテーション情報について説明します。

@rep:scope

アノテーション・タイプ	@rep:scope
構文	@rep:scope public private internal

アノテーション・タイプ	@rep:scope
用途	インタフェースを公開する場所を示します(該当する場合)。
例	@rep:scope public は、すべての場所に公開することを意味します。 注意: 公開インタフェースは、顧客が使用する UI に表示されます。 @rep:scope private は、インタフェースは統合リポジトリに公開されるが、所有するチームに使用が制限されることを意味します。 @rep:scope internal は、社内で公開されることを意味します。
必須	すべてのインタフェースに必須です。
デフォルト	なし。
レベル	インタフェース(クラス)と API (メソッド)。
複数の許可	いいえ。プログラム要素(クラスまたはメソッド)ごとに 1 つのみ使用します。

次の表では、@rep:product のアノテーション情報について説明します。

@rep:product

アノテーション・タイプ	@rep:product
構文	@rep:product <i>StringShortCode</i>
用途	インタフェースの製品の短縮名を指定します。
例	@rep:product PO
必須	すべてのインタフェースに必須です。

アノテーション・タイプ	@rep:product
デフォルト	なし。
レベル	インタフェース(クラス)のみ。
複数の許可	いいえ。インタフェースごとに1つのみ使用します。

次の表では、@rep:implementation のアノテーション情報について説明します。

@rep:implementation

アノテーション・タイプ	@rep:implementation
構文	@rep:implementation <i>StringClassName</i>
用途	インタフェースの実装クラス名を指定します。
例	@rep:implementation oracle.apps. po.server.PurchaseOrdersAmImpl
必須	Java でのみ必須です。
デフォルト	なし。
レベル	インタフェース(クラス)。
複数の許可	いいえ。インタフェースごとに1つのみ使用します。

次の表では、@rep:displayname のアノテーション情報について説明します。

@rep:displayname

アノテーション・タイプ	@rep:displayname
構文	@rep:displayname <i>StringName</i>
用途	インタフェースのユーザーフレンドリな名前を定義します。

アノテーション・タイプ	@rep:displayname
例	@rep:displayname Purchase Order Summary
必須	@rep: scope public を扱うすべてのインタ フェースに必須です。
デフォルト	なし。
レベル	インタフェース(クラス)と API (メソッド)。
複数の許可	いいえ。プログラム要素(クラスまたはメソッド)ごと に 1 つのみ使用します。

アノテーション・タイプ	@rep:displayname
コメント	表示名のガイドライン これらのガイドラインは、すべてのテクノロジー(インタフェース、クラス、メソッド、パラメータ、XMLG マップなど)の表示名に適用されます。 表示名は、次の基準を満たす必要があります。 - 大文字と小文字の両方を使用します。すべてを大文字にしたり、すべてを小文字にしないでください。 - 複数形ではなく単数形にします。たとえば、Customers ではなく、Customer を使用します。 - 完全に修飾された、ビジネス・エリアを表現するものにします。 - アンダースコア(_)は使用しません。 - ピリオド(.)で終了しません。 - 内部名と同じにしません。 - 製品コードまたは製品名で開始しません。 Package、API、APIs など、明らかに不適切なものは含みません。表示名を記述するときは、表示名が表示される UI を考慮してください。 たとえば、IEX_PROMISES_PUB ではなく Promise Activity を表示名として使用します。これは、IEX_PROMISES_PUB はアンダースコアを含み、内部名と同じであるためです。 Workflow Process Activity APIs ではなく Process Activity を表示名として使用します。これは、製品名で開始し、APIs で終了するためです。

次の表では、@rep:lifecycle のアノテーション情報について説明します。

@rep:lifecycle

アノテーション・タイプ	@rep:lifecycle
構文	@rep:lifecycle active deprecated obsolete planned
用途	インタフェースのライフサイクル・フェーズを指定します。
例	@rep:lifecycle active は、インタフェースが有効であることを意味します。 @rep:lifecycle deprecated は、インタフェースが非推奨になったことを意味します。 @rep:lifecycle obsolete は、インタフェースが廃止されて使用できなくなったことを意味します。 @rep:lifecycle planned は、インタフェースが将来のリリースで予定されていることを意味します。これは、プロトタイプおよびモックアップに使用されます。
必須	オプション。
デフォルト	デフォルト値は active です。
レベル	インタフェース(クラス)と API (メソッド)。
複数の許可	いいえ。プログラム要素(クラスまたはメソッド)ごとに 1 つのみ使用します。
コメント	パーサーで、このアノテーションが@deprecated の Javadoc アノテーションと同期しているかが検証されます。

次の表では、@rep:compatibility のアノテーション情報について説明します。

@rep:compatibility

アノテーション・タイプ	@rep:compatibility
構文	@rep:compatibility S N

アノテーション・タイプ	@rep:compatibility
用途	S は、インタフェースのライフサイクル・フェーズを示します。 N は、下位互換性が保証されていないことを示します。
例	@rep:compatibility S
必須	オプション。
デフォルト	条件付き。@rep:scope public では、この値は S にデフォルト設定されます。それ以外では、この値は N にデフォルト設定されます。
レベル	インタフェース(クラス)と API (メソッド)。
複数の許可	いいえ。プログラム要素(クラスまたはメソッド)ごとに 1 つのみ使用します。

次の表では、@link のアノテーション情報について説明します。

@link

アノテーション・タイプ	@link
	注意: Java の宛先に対してのみサポートされています。
構文	{@link package.class#member label}
用途	別のインタフェースまたはメソッドへのリンクを提供します。
例	{@link #setAmounts(int,int,int,int) Set Amounts}
必須	オプション。
デフォルト	なし。
レベル	インタフェース(クラス)と API (メソッド)。

アノテーション・タイプ	@link
	注意: Java の宛先に対してのみサポートされています。
複数の許可	はい。
コメント	これは、UI に表示される説明でリンク項目がハイパーリンクとして埋め込まれる、標準の Javadoc の @link アノテーションです。 次のルールに留意してください。公開 API を、非公開または内部の API にリンクしないでください。@link アノテーションは、統合リポジトリのビューワがアクセスできないドキュメントには、リンクしないでください。

次の表では、@see のアノテーション情報について説明します。

@see

アノテーション・タイプ	@see
構文	@see StringLocator
用途	別のインタフェースまたはメソッドへのリンクを提供します。
例	@see #setAmounts(int,int,int,int)
必須	オプション。
デフォルト	なし。
レベル	インタフェース(クラス)と API (メソッド)。
複数の許可	はい。

アノテーション・タイプ	@see
コメント	これは、標準の Javadoc の@see アノテーションです。 リンク項目が、「参照」ヘッダーの下で UI に表示されます。 PL/SQL コードでの使用方法: @see package#procedure

次の表では、@rep:ihelp のアノテーション情報について説明します。

@rep:ihelp

アノテーション・タイプ	@rep:ihelp
構文	1 行で個別の子アノテーションとして使用する場合、次のようにします。 <pre>@rep:ihelp <product_shortname>/@<help_target> #<help_target> <link_text></pre> インライン・アノテーションとして使用する場合、中カッコを追加します。 <pre>{@rep:ihelp <product_shortname>/@<help_target> #<help_target> <link_text>}</pre>
用途	既存の HTML オンライン・ヘルプ・ページへのリンクを提供します。 product_shortname は製品の短縮名です。 help_target は、テクニカル・ライターがファイルに手動で埋め込んだ、jtfacccsum.jsp、aolpo、overview、ast_aboutcollateral などのヘルプ・ターゲットです。 Oracle E-Business Suite ヘルプのカスタマイズ方法の詳細は、<i>Oracle E-Business Suite セットアップ・ガイド</i>の「Oracle E-Business Suite ヘルプの設定」を参照してください。
例	<pre>@rep:ihelp #setAmounts(int,int, int,int)</pre>

アノテーション・タイプ	@rep:ihelp
必須	オプション。
デフォルト	なし。
レベル	インタフェース(クラス)と API (メソッド)。
複数の許可	はい。

次の表では、@rep:metalink のアノテーション情報について説明します。

@rep:metalink

アノテーション・タイプ	@rep:metalink
構文	1 行で個別の子アノテーションとして使用する場 合、次のようにします。 <pre>@rep:metalink <bulletin_number> <link_text></pre> インライン・アノテーションとして使用する場 合、中カッコを追加します。 <pre>{@rep:metalink <bulletin_number> <link_text>}</pre>
用途	既存の My Oracle Support (旧 Oracle <i>MetaLink</i>) のナレッジ・ドキュメントへのリンクを提供します。
例	<pre>@rep:metalink 123456.1 See My Oracle Support Knowledge Document 123456.1</pre>
必須	オプション。
デフォルト	なし。
レベル	インタフェース(クラス)と API (メソッド)。
複数の許可	はい。

次の表では、@rep:category のアノテーション情報について説明します。

@rep:category

アノテーション・タイプ	@rep:category
構文	• @rep:category BUSINESS_ENTITY BUSINESS_ENTITY_CODE • @rep:category IREP_CLASS_SUBTYPE JAVA_BEAN_SERVICES • @rep:category IREP_CLASS_SUBTYPE AM_SERVICES
用途	• インタフェースのビジネス・カテゴリを指定します。 • Java API をサービス可能インタフェースとして指定します。 • アプリケーション・モジュール・クラスをサービス可能インタフェースとして指定します。
例	• @rep:category BUSINESS_ENTITY PO_PLANNED_PURCHASE_ORDER PO_PLANNED_PURCHASE_ORDER はビジネス・エンティティ・コードであり、たとえば、表示名は「計画発注」になります。 • @rep:category IREP_CLASS_SUBTYPE JAVA_BEAN_SERVICES Java API のサブタイプとして「Java Bean サービス」を表示できます。 • @rep:category IREP_CLASS_SUBTYPE AM_SERVICES Java API のサブタイプとして「アプリケーション・モジュール・サービス」を表示できます。 詳細は、「ビジネス・エンティティ・アノテーション・ガイドライン」(ページ A-41)を参照してください。

アノテーション・タイプ	@rep:category
必須	- BUSINESS_ENTITY は、すべてのインタフェースに必須です。クラス ALL に属するメソッドが同じビジネス・エンティティを持つ場合は、クラスを注釈付けすることのみが必要となります。ただし、クラスに属するメソッドが異種ビジネス・エンティティを持つ場合は、それぞれのメソッドを適切に注釈付けする必要があります。 - IREP_CLASS_SUBTYPE JAVA_BEAN_SERVICES のアノテーションは、サービス可能 Java API として識別する必要があるすべての Java インタフェースに必須です。 - IREP_CLASS_SUBTYPE AM_SERVICES のアノテーションは、サービス可能 API として識別する必要があるアプリケーション・モジュールに必須です。
デフォルト	なし
レベル	BUSINESS_ENTITY は、クラス・レベルとメソッド・レベルの両方に適用可能です。ただし、JAVA_BEAN_SERVICES と AM_SERVICES は、クラス・レベルにのみ適用可能です。
複数の許可	はい。
コメント	コードでは rep:category アノテーションを頻繁に使用することをお勧めします。

次の表では、@rep:usestable のアノテーション情報について説明します。

@rep:usestable

アノテーション・タイプ	@rep:usestable
構文	@rep:usestable <table or view name> <sequence> <direction flag>

アノテーション・タイプ	@rep:usestable
用途	関連付けられたオープン・インタフェース表またはオープン・インタフェース・ビューを識別するために、コンカレント・プログラムを注釈付けする場合に使用されます。 <table or view name>は、表またはビューの名前です。 <sequence>は、様々なピースの表示順序を UI に通知するために使用される整数です。慣例的に、rep: category OPEN_INTERFACE (ページ A-132)アノテーションで、コンカレント・プログラムに 1 を使用します。この rep:usestable アノテーションでは、入力表を順序付けします(詳細(明細)表の前にマスター(ヘッダー)表をリストします)。最後に、順序の終わりに出力ビューまたは表を配置します。 <direction flag>はオプションで、IN (デフォルト)、OUT または BOTH のいずれかを指定します。
例	@rep:usestable SampleTable 3 IN
必須	コンカレント・プログラムがオープン・インタフェースの一部である場合のみ。
デフォルト	なし。
レベル	インタフェース。
複数の許可	はい。

次の表では、@rep:standard のアノテーション情報について説明します。

@rep:standard

アノテーション・タイプ	@rep:standard
構文	@rep:standard StringType StringVersionNumber StringSpecName たとえば、@rep:standard OAG 7.2 Process_PO_001 の場合、StringType は OAG、StringVersionNumber は 7.2、StringSpecName は Process_PO_001 です このアノテーションの構文の詳細は、「アノテーションの構文」(ページ A-1)を参照してください。

アノテーション・タイプ	@rep:standard
用途	ビジネスの標準名を指定します。このアノテーションは、Oracle が業界標準に準拠する箇所のために予約されています。
例	たとえば、@rep:standard RosettaNet 02.02.00 'Pip3B12-Shipping Order Confirmation' の場合、仕様名に空スペースが含まれるため、StringSpecName は単一引用符で囲まれています。@rep:standard RosettaNet 02.02.00 Pip3B12- PurchaseOrderConfirmation の例のように、StringSpecName に空スペースが含まれない場合、これらの引用符は必要ありません。
必須	オプション。
デフォルト	メソッドは、クラスで設定された値にデフォルト設定されます。
レベル	ドキュメントおよびデータ行。
複数の許可	いいえ。プログラム要素(クラスまたはメソッド)ごとに 1 つのみ使用します。

次の表では、@rep:httpverb のアノテーション情報について説明します。

@rep:httpverb

アノテーション・タイプ	@rep:httpverb
構文	@rep:httpverb <HTTP_Method_Types> メソッド・レベルで、HTTP 動詞(GET および POST)のカンマ区切りリストを使用します。 注意: このリリースでは、GET および POST は、Java Bean サービスとアプリケーション・モジュール・サービスのサポートされる HTTP メソッドです。

アノテーション・タイプ	@rep:httpverb
用途	このアノテーションを使用して、現在のメソッドまたは操作に適した HTTP 動詞を指定します。 メソッドが POST httpverb で注釈付けされない場合、引き続き、選択したインタフェース詳細ページで「POST」チェック・ボックスがデフォルトで有効になります。これにより、統合管理者は、サービスの配置前に必要な場合に、そのメソッドに対して POST 動詞を選択できます。ただし、POST HTTP 動詞とは異なり、メソッドが GET httpverb で注釈付けされない場合は、そのメソッドの「GET」チェック・ボックスが無効になります。これは、メソッドが GET 操作を備えた REST サービスとして公開されないことを意味します。 このアノテーションは、Java ファイルにのみ使用可能です。
例	カンマ区切りリストは、次の方法で使用できます。 <pre>@rep:httpverb get @rep:httpverb post @rep:httpverb get, post</pre>
必須	オプション。
デフォルト	なし。
レベル	メソッド・レベル
複数の許可	いいえ。メソッドごとに 1 つのみ使用します。
コメント	このアノテーションを使用して、GET や POST などの HTTP メソッドを最適化します。

次の表では、@rep:interface のアノテーション情報について説明します。

@rep:interface

アノテーション・タイプ	@rep:interface
構文	<pre>@rep:interface StringClassName (StringClassName の構文は transactiontype: subtype)。次の例を参照してください。</pre>

アノテーション・タイプ	@rep:interface
用途	解析ツールでインタフェース名を容易にイントロスペクトできないテクノロジーのために、インタフェース名を指定します。
例	StringClassName は常に transactiontype: subtype です @rep:interface PO:POC
必須	オプション。
デフォルト	なし。
レベル	インタフェースのみ。
複数の許可	いいえ。インタフェースごとに 1 つのみ使用します。
コメント	XML ゲートウェイや E-Commerce ゲートウェイなどの、インタフェースの強力なネイティブ定義がないテクノロジーで使用されます

次の表では、@param のアノテーション情報について説明します。

@param

アノテーション・タイプ	@param
構文	@param paramName paramDescription
用途	メソッド、プロシージャまたは関数パラメータの名前と説明を指定します(IN、OUT または両方)。
例	@param PONumber The purchase order number.
必須	オプション。
デフォルト	なし。
レベル	メソッド、プロシージャおよび関数。
複数の許可	はい。

アノテーション・タイプ	@param
コメント	便宜上、Java アノテーションもサポートされています。

次の表では、@return のアノテーション情報について説明します。

@return

アノテーション・タイプ	@return
構文	@return <i>StringDescription</i>
用途	メソッドまたは関数の戻りパラメータの説明を指定します。
例	@return The purchase order status.
必須	オプション。
デフォルト	なし。
レベル	メソッド、プロシージャおよび関数。
複数の許可	はい。

次の表では、@rep:paraminfo のアノテーション情報について説明します。

@rep:paraminfo

アノテーション・タイプ	@rep:paraminfo
構文	@rep:paraminfo {@rep:innertype typeName} {@rep:precision value} {@rep:required} {@rep:key_param}

アノテーション・タイプ	@rep:paraminfo
用途	rep:paraminfo rep:paraminfo アノテーションは、説明しているパラメータの @param または @return のアノテーションの直後の行に配置する必要があります。 rep:innertype 回収などの汎用オブジェクトの内部タイプを説明するオプションのインライン・アノテーション。 rep:precision パラメータの精度を指定するオプションのインライン・アノテーション。文字列と数値に使用されます。 rep:required null 以外を指定する必要があることを示すオプションのインライン・アノテーション。これは、PL/SQL 以外のテクノロジーにのみ必要です。 rep:key_param パラメータを REST サービスのキー・パラメータまたはパス変数として定義するオプションのインライン・アノテーション。これは、Java またはアプリケーション・モジュールのメソッドに適用可能です。このアノテーションが使用される場合は、rep:required が存在する必要があります。また、このアノテーションは、rep:paraminfo (@return アノテーションの後)には適用できません。
例	<pre>/** * Gets the price for a purchase order line * item. * * @param poNumber purchase order unique * identifier * @paraminfo {@rep:precision 10} {@rep: * required} {@rep:key_param} * @param lineNumber purchase order line * unique identifier * @paraminfo {@rep:precision 10} {@rep: * required} * @return the item price for the given * purchase order line * @paraminfo {@rep:precision 10} * * @rep:scope public * @rep:displayname Get Purchase Order Line * Item Price */ public Number getItemPrice(Number poNumber, Number lineNumber);</pre>
必須	オプション。

アノテーション・タイプ	@rep:paraminfo
デフォルト	なし。
レベル	メソッドのみ。
複数の許可	はい。各種パラメータで、複数の値を割り当てることができます。

次の表では、@rep:businesssevent のアノテーション情報について説明します。

@rep:businesssevent

アノテーション・タイプ	@rep:businesssevent
構文	@rep:businesssevent BusinessEvent
用途	このメソッドによって呼び出されるビジネス・イベントの名前を指定します。
例	@rep:businesssevent oracle.apps.wf.notification.send
必須	オプション。
デフォルト	ビジネス・イベントを導出できるファイル・タイプでデフォルト設定されます。
レベル	メソッドのみ。
複数の許可	はい。
コメント	このアノテーションは、ビジネス・イベントを呼び出すすべてのインスタンスで使用するようにしてください。ビジネス・イベント自体にはアノテーションは必要ありません。

次の表では、@rep:direction のアノテーション情報について説明します。

@rep:direction

アノテーション・タイプ	@rep:direction
構文	@rep:direction <OUT IN>
用途	インタフェースがアウトバウンドかインバウンドかを指定します。
例	@rep:direction OUT
必須	EDI および XML ゲートウェイ・アノテーションでのみ必須です。
デフォルト	なし。
レベル	インタフェース。
複数の許可	いいえ。

次の表では、@rep:service のアノテーション情報について説明します。

@rep:service

アノテーション・タイプ	@rep:service
構文	@rep:service
用途	Java ファイルがビジネス・サービス・オブジェクト(BSO)であることを指定します。 このタグは、Java ファイル内と同様に使用します。次の例の項を参照してください。パラメータはありません。

アノテーション・タイプ	@rep:service
例	<pre> /** * The Purchase Order service lets you to * view, update, acknowledge and * approve purchase orders.It also lets you * receive items, and obtain * pricing by line item. * * @see oracle.apps.fnd.framework.toolbox. * tutorial.PurchaseOrderSDO * @see oracle.apps.fnd.framework.toolbox. * tutorial.PurchaseOrderAcknowledgementsSDO * @see oracle.apps.fnd.framework.toolbox. * tutorial.PurchaseOrderReceiptsSDO * * @rep:scope public * @rep:displayname Purchase Order Service * @rep:implementation oracle.apps.fnd. * framework.toolbox.tutorial.server. * PurchaseOrderSAMImpl * @rep:product PO * @rep:category BUSINESS_ENTITY * PO_PURCHASE_ORDER * @rep:service */ </pre>
必須	ビジネス・サービス・オブジェクトに必須です。
デフォルト	なし。
レベル	クラス。
複数の許可	いいえ。

次の表では、@rep:servicedoc のアノテーション情報について説明します。

@rep:servicedoc

アノテーション・タイプ	@rep:servicedoc
構文	@rep:servicedoc
用途	Java ファイルが(通常の Java API とは対照的に) SDO であることを指定します。このタグは、Java ファイル内と同様に使用します。次の例の項を参照してください。パラメータはありません。

アノテーション・タイプ	@rep:servicedoc
例	<pre> /** * The Purchase Order Data Object holds the * purchase order data including * nested data objects such as lines and * shipments. * * @see oracle.apps.fnd.framework.toolbox. * tutorial.PurchaseOrderLineSDO * * @rep:scope public * @rep:displayname Purchase Order Data * Object * @rep:product PO * @rep:category BUSINESS_ENTITY * PO_PURCHASE_ORDER * @rep:servicedoc */ </pre>
必須	サービス・データ・オブジェクトに必須です。
デフォルト	なし。
レベル	クラス。
複数の許可	いいえ。
コメント	このアノテーションは自動的に生成されるため、開発者はこのアノテーションを入力する必要はありません。

次の表では、@rep:synchronicity のアノテーション情報について説明します。

@rep:synchronicity

アノテーション・タイプ	@rep:synchronicity
構文	@rep:synchronicity <SYNCH or ASYNCH>
用途	同期または非同期の動作を指定します。
例	@rep:synchronicity SYNCH
必須	オプション。

アノテーション・タイプ	@rep:synchronicity
デフォルト	モジュール・タイプに基づいてデフォルト設定されます。たとえば、XML ゲートウェイの場合は ASYNCH で、ビジネス・サービス・オブジェクトの場合は SYNCH です。
レベル	クラスまたはメソッド。
複数の許可	いいえ。

次の表では、@rep:appscontext のアノテーション情報について説明します。

@rep:appscontext

アノテーション・タイプ	@rep:appscontext
構文	@rep:appscontext <NONE, APPL, RESP, USER, NLS, or ORG>
用途	メソッドの実行に必要なコンテキストを指定します。
例	@rep:appscontext USER
必須	オプション。
デフォルト	NONE
レベル	メソッド。
複数の許可	いいえ、メソッドごとに 1 つのみ許可されます。

次の表では、@rep:comment のアノテーション情報について説明します。

@rep:comment

アノテーション・タイプ	@rep:comment
構文	@rep:comment <comment>
用途	このアノテーションは、パーサーではスキップされます。これは、非公開のコメントが必要になった場合に、製品チームが使用します。

アノテーション・タイプ	@rep:comment
例	@rep:comment This is a sample comment.
必須	オプション。
デフォルト	なし。
レベル	すべて。

次の表では、@rep:primaryinstance のアノテーション情報について説明します。

@rep:primaryinstance

アノテーション・タイプ	@rep:primaryinstance
構文	@rep:primaryinstance
用途	過負荷のメソッドまたプロシージャのプライマリ・インスタンスを指定します。
必須	すべての過負荷のメソッドおよびプロシージャに必須です。
デフォルト	なし。
レベル	メソッドまたはプロシージャ。
複数の許可	いいえ。
コメント	プライマリ・インスタンスの表示名と説明は、メソッドのリストが表示されるときに、ブラウザ UI で使用されます。非プライマリ・インスタンス(過負荷など)には、プライマリとはどのように異なるかを強調する説明(This variant allows specification of the org_id. など)が必要になります。非プライマリの表示名と説明は、過負荷のインタフェースの詳細を表示するときのみ表示されます。

次の表では、@rep:usesmap のアノテーション情報について説明します。

@rep:usesmap

アノテーション・タイプ	@rep:usesmap
構文	@rep:usesmap <map_name> <sequence_number>
用途	コンカレント・プログラムに関連付けられる E-Commerce ゲートウェイ・マップを指定します。 <map_name>の map_name はデフォルトのマップ名です。 <sequence_number>は、様々なピースの表示順序を UI に通知するために使用される整数です。
例	@rep:usesemap SampleMap 2
必須	オプション。
デフォルト	なし。
レベル	すべて。
複数の許可	はい。
コメント	デフォルトのマップ名には EC_XXXX_FF の命名規則があり、ここで XXXX は、取引の 4 文字の頭文字です。

サーバー接続の構成

アプリケーションサーバー接続

セキュリティはサービス・コンテンツを不正なアクセスから保護するための最も重要な機能です。SOA コンポジット・アプリケーションおよび Oracle SOA Suite WebLogic Server ドメインに配置されたその他のリソースを保護し、サービス起動が正常に実行されるようにするには、必要なサーバー接続が実行される必要があります。

この項では、次のサーバー接続を取り上げます。

- アプリケーション・サーバー接続の作成、ページ B-1

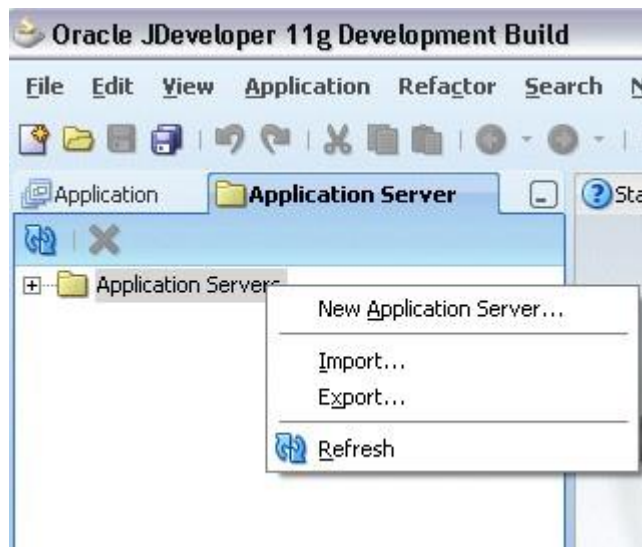
アプリケーション・サーバー接続の作成

設計時環境と、その配置先となるサーバー間の接続を確立する必要があります。このような接続を確立するには、アプリケーション・サーバー(Oracle WebLogic Server)接続を作成する必要があります。

次のステップを使用して、Oracle WebLogic Server への接続を作成します。

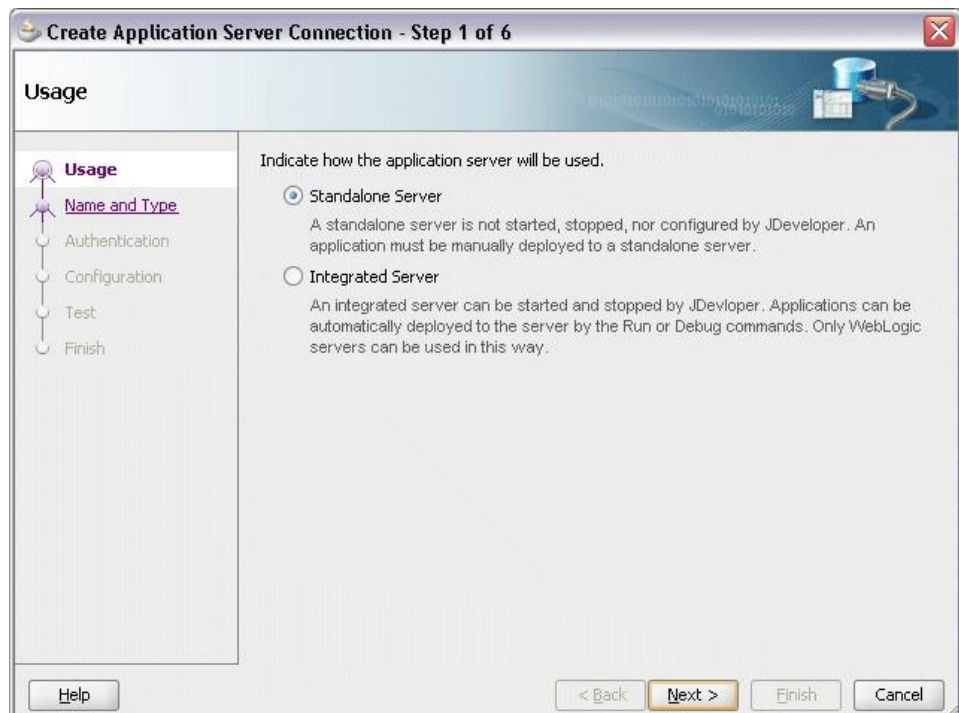
1. Oracle JDeveloper で、「表示」→「アプリケーション・サーバー・ナビゲータ」を選択して「アプリケーション・サーバー」タブを開きます。

新規アプリケーション・サーバー接続の作成



2. 「アプリケーション・サーバー」を右クリックして「新規アプリケーション・サーバー」を選択し、「アプリケーション・サーバー接続の作成」ウィザードを開きます。
3. 「使用方法」で「スタンドアロン・サーバー」を選択し、「次へ」をクリックします。

アプリケーション・サーバー接続の作成:「使用方法」ページ



4. 接続名(soa-server1 など)を入力し、接続タイプとして「**WebLogic 10.3**」を選択します。

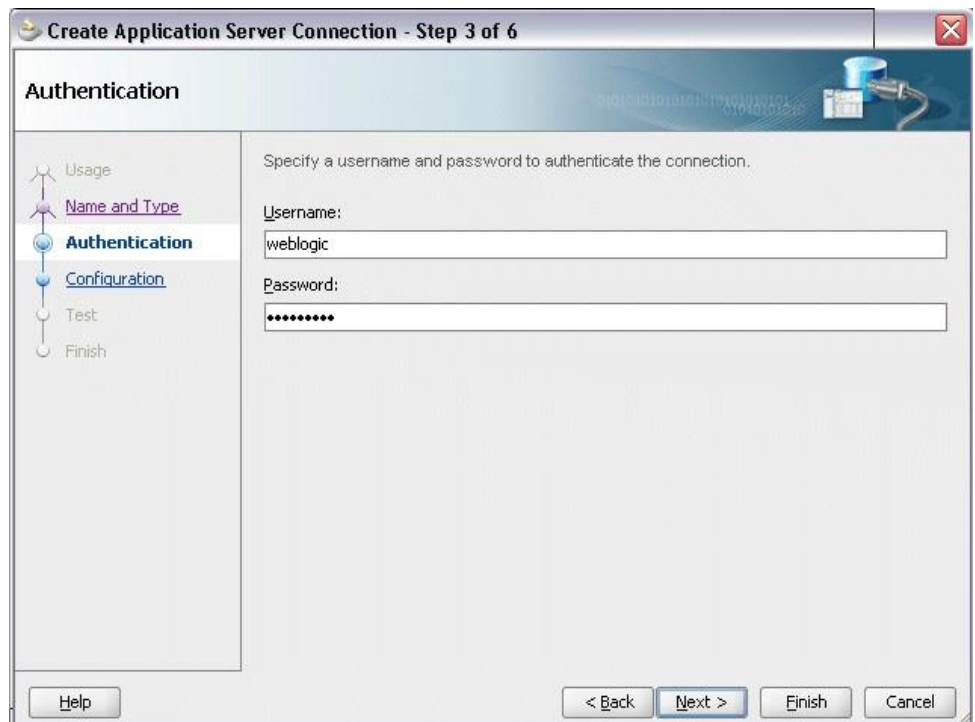
アプリケーション・サーバー接続の作成: 「名前とタイプ」ページ

The screenshot shows a Windows-style dialog box titled "Create Application Server Connection - Step 2 of 6". The main title bar also includes the Japanese text "アプリケーション・サーバー接続の作成: 「名前とタイプ」ページ". The dialog is divided into two main sections. On the left is a vertical navigation pane with five items: "Usage" (with a purple icon), "Name and Type" (with a purple icon and highlighted in blue), "Authentication" (with a blue icon), "Configuration" (with a grey icon), "Test" (with a grey icon), and "Finish" (with a grey icon). The right section has a header "Name and Type" and a sub-header "Specify a unique name and type for the connection. The name must be a valid Java identifier." Below this, there are two labels: "Create connection in:" with the value "Resource Palette", and "Connection Name:" with a text input field containing "soa-server1". Below the text field is a label "Connection Type:" with a dropdown menu showing "WebLogic 10.3". At the bottom of the dialog are four buttons: "Help", "< Back", "Next >", "Finish", and "Cancel". The "Next >" button is highlighted with a blue border.

「次へ」をクリックします。

5. Oracle SOA Suite のインストール時に指定した有効なユーザー名(weblogic など)とパスワードを入力します。「次へ」をクリックします。

アプリケーション・サーバー接続の作成:「認証」ページ



Create Application Server Connection - Step 3 of 6

Authentication

Usage
Name and Type
Authentication
Configuration
Test
Finish

Specify a username and password to authenticate the connection.

Username:
weblogic

Password:
.....

Help < Back Next > Finish Cancel

6. WebLogic Server 接続のホスト名およびポート情報を入力します。「WebLogic ドメイン」フィールドに soainfra と入力します。

アプリケーション・サーバー接続の作成: 「構成」ページ

The screenshot shows a Windows-style dialog box titled "Create Application Server Connection - Step 4 of 6". The "Configuration" page is active, indicated by a blue highlight in the left-hand navigation pane. The navigation pane includes links for Usage, Name and Type, Authentication, Configuration, Test, and Finish. The main area contains the following text and fields:

WebLogic Server connections use a host name and port to establish a connection. The Domain of the target will be verified

Weblogic Hostname (Administration Server):
localhost

Port: 7001 SSL Port: 7002

☐ Always use SSL

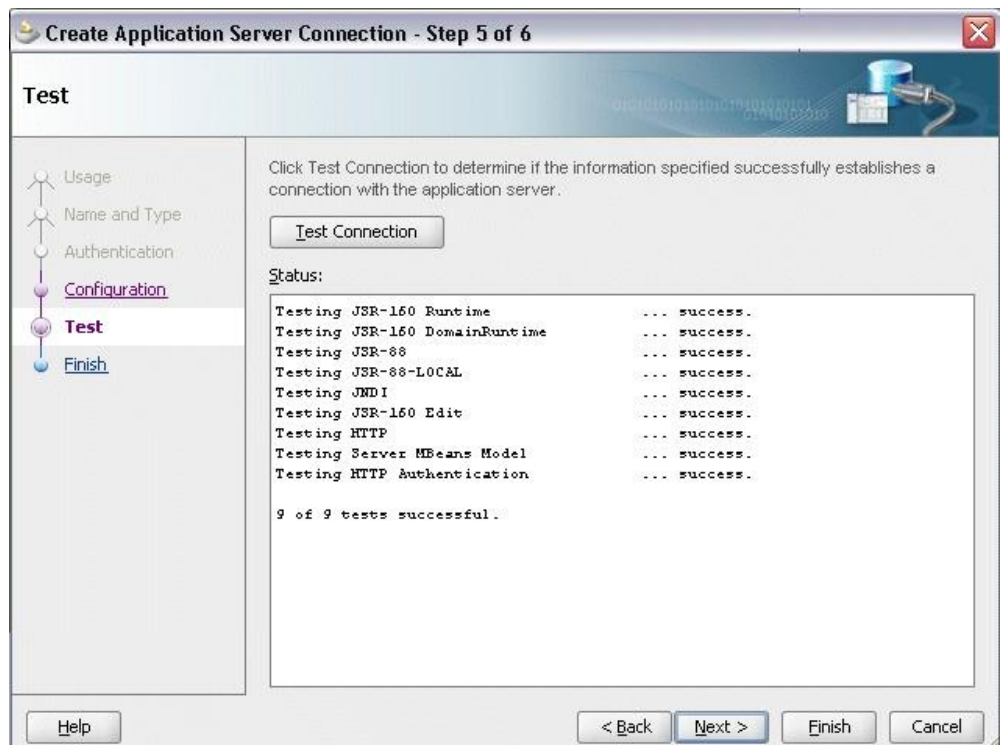
Weblogic Domain:
soainfra

At the bottom, there are buttons for Help, < Back, Next >, Finish, and Cancel.

「次へ」をクリックします。「テスト」ページが表示されます。

7. 「接続のテスト」をクリックし、サーバー構成を検証します。「ステータス」ウィンドウに成功メッセージが移入されます。

サーバー構成の検証



「終了」をクリックします。

サンプル・ペイロード

仕入先出荷およびデビット要求の作成のためのサンプル・ペイロード

次の情報は、InputCreateSDRequest.xml ファイルのサンプル・ペイロードを示しています。

```

<?xml version="1.0" encoding="UTF-8"?>
<cre:InputParameters xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:cre="http://xmlns.oracle.
com/apps/ozf/soapprovider/plsql/ozf_sd_request_pub/create_sd_request/">
  <cre:P_API_VERSION_NUMBER>1.0</cre:P_API_VERSION_NUMBER>
<cre:P_INIT_MSG_LIST>T</cre:P_INIT_MSG_LIST>
  <cre:P_COMMIT>F</cre:P_COMMIT>
<cre:P_VALIDATION_LEVEL>100</cre:P_VALIDATION_LEVEL>
  <cre:P_SDR_HDR_REC>
    <cre:REQUEST_NUMBER>SDR-CREATE-BPEL1</cre:REQUEST_NUMBER>
    <cre:REQUEST_START_DATE>2008-08-18T12:00:00</cre:
      REQUEST_START_DATE>
    <cre:REQUEST_END_DATE>2008-10-
      18T12:00:00</cre:REQUEST_END_DATE>>
    <cre:USER_STATUS_ID>1701</cre:USER_STATUS_ID>
    <cre:REQUEST_OUTCOME>IN_PROGRESS</cre:REQUEST_OUTCOME>
    <cre:REQUEST_CURRENCY_CODE>USD</cre:REQUEST_CURRENCY_CODE>
    <cre:SUPPLIER_ID>601</cre:SUPPLIER_ID>
    <cre:SUPPLIER_SITE_ID>1415</cre:SUPPLIER_SITE_ID>
    <cre:REQUESTOR_ID>100001499</cre:REQUESTOR_ID>
    <cre:ASSIGNEE_RESOURCE_ID>100001499</cre:ASSIGNEE_RESOURCE_ID>
    <cre:ORG_ID>204</cre:ORG_ID>
    <cre:ACCRUAL_TYPE>SUPPLIER</cre:ACCRUAL_TYPE>
    <cre:REQUEST_DESCRIPTION>Create</cre:REQUEST_DESCRIPTION>
    <cre:SUPPLIER_CONTACT_EMAIL_ADDRESS>sdr.supplier@example.
      com</cre:SUPPLIER_CONTACT_EMAIL_ADDRESS>
    <cre:SUPPLIER_CONTACT_PHONE_NUMBER>2255</cre:
      SUPPLIER_CONTACT_PHONE_NUMBER>
    <cre:REQUEST_TYPE_SETUP_ID>400</cre:REQUEST_TYPE_SETUP_ID>
    <cre:REQUEST_BASIS>Y</cre:REQUEST_BASIS>
    <cre:USER_ID>1002795</cre:USER_ID>
  </cre:P_SDR_HDR_REC>
  <cre:P_SDR_LINES_TBL>
    <cre:P_SDR_LINES_TBL_ITEM>
      <cre:PRODUCT_CONTEXT>PRODUCT</cre:PRODUCT_CONTEXT>
      <cre:INVENTORY_ITEM_ID>2155</cre:INVENTORY_ITEM_ID>
      <cre:ITEM_UOM>Ea</cre:ITEM_UOM>
      <cre:REQUESTED_DISCOUNT_TYPE>%</cre:REQUESTED_DISCOUNT_TYPE>
      <cre:REQUESTED_DISCOUNT_VALUE>20</cre:REQUESTED_DISCOUNT_VALUE>
      <cre:COST_BASIS>200</cre:COST_BASIS>
      <cre:MAX_QTY>200</cre:MAX_QTY>
      <cre:APPROVED_DISCOUNT_TYPE>%</cre:APPROVED_DISCOUNT_TYPE>
      <cre:APPROVED_DISCOUNT_VALUE>20</cre:APPROVED_DISCOUNT_VALUE>
      <cre:APPROVED_MAX_QTY>200</cre:APPROVED_MAX_QTY>
      <cre:VENDOR_APPROVED_FLAG>Y</cre:VENDOR_APPROVED_FLAG>
      <cre:PRODUCT_COST_CURRENCY>USD</cre:PRODUCT_COST_CURRENCY>
      <cre:END_CUSTOMER_CURRENCY>USD</cre:END_CUSTOMER_CURRENCY>
    </cre:P_SDR_LINES_TBL_ITEM>
  </cre:P_SDR_LINES_TBL>
  <cre:P_SDR_CUST_TBL>
    <cre:P_SDR_CUST_TBL_ITEM>
      <cre:CUST_ACCOUNT_ID>1290</cre:CUST_ACCOUNT_ID>
      <cre:PARTY_ID>1290</cre:PARTY_ID>
      <cre:SITE_USE_ID>10479</cre:SITE_USE_ID>
      <cre:CUST_USAGE_CODE>BILL_TO</cre:CUST_USAGE_CODE>
      <cre:END_CUSTOMER_FLAG>N</cre:END_CUSTOMER_FLAG>
    </cre:P_SDR_CUST_TBL_ITEM>
    <cre:P_SDR_CUST_TBL_ITEM>
      <cre:CUST_ACCOUNT_ID>1287</cre:CUST_ACCOUNT_ID>
      <cre:PARTY_ID>1287</cre:PARTY_ID>
      <cre:SITE_USE_ID>1418</cre:SITE_USE_ID>
      <cre:CUST_USAGE_CODE>CUSTOMER</cre:CUST_USAGE_CODE>
      <cre:END_CUSTOMER_FLAG>Y</cre:END_CUSTOMER_FLAG>
    </cre:P_SDR_CUST_TBL_ITEM>
  </cre:P_SDR_CUST_TBL>
</cre:InputParameters>

```

インバウンド・プロセス発注 XML 取引のためのサンプル・ペイロード

次の情報は、order_data_xm1g.xml ファイルのサンプル・ペイロードを示しています。

```

<?xml version = '1.0' encoding = 'UTF-8' standalone = 'no'?>
<!-- Oracle eXtensible Markup Language Gateway Server -->
<!DOCTYPE PROCESS_PO_007 SYSTEM "003_process_po_007.dtd">
  <PROCESS_PO_007>
    <CNTROLAREA>
      <BSR>
        <VERB>PROCESS</VERB>
        <NOUN>PO</NOUN>
        <REVISION>007</REVISION>
      </BSR>
    <SENDER>
      <LOGICALID>ORACLE</LOGICALID>
      <COMPONENT>PURCHASING</COMPONENT>
      <TASK>POISSUE</TASK>
      <REFERENCEID>MO4YD220.US.EXAMPLE.COM:oracle.apps.po.event.xmlpo:
32636-148970</REFERENCEID>
      <CONFIRMATION>0</CONFIRMATION>
      <LANGUAGE>us</LANGUAGE>
      <CODEPAGE>UTF8</CODEPAGE>
      <AUTHID>APPS</AUTHID>
    </SENDER>
    <DATETIME qualifier="CREATION" type="T" index="1">
      <YEAR>2011</YEAR>
      <MONTH>04</MONTH>
      <DAY>05</DAY>
      <HOUR>23</HOUR>
      <MINUTE>44</MINUTE>
      <SECOND>09</SECOND>
      <SUBSECOND>0000</SUBSECOND>
      <TIMEZONE>+0000</TIMEZONE>
    </DATETIME>
    <OPERAMT qualifier="EXTENDED" type="T">
      <VALUE>1329432</VALUE>
      <NUMOFDEC>2</NUMOFDEC>
      <SIGN>+</SIGN>
      <CURRENCY>USD</CURRENCY>
      <UOMVALUE>1</UOMVALUE>
      <UOMNUMDEC>0</UOMNUMDEC>
      <UOM/>
    </OPERAMT>
    <POID>PO-4466-5</POID>
    <POTYPE>STANDARD</POTYPE>
    <ACKREQUEST>N</ACKREQUEST>
    <CONTRACTB/>
    <CONTRACTS/>
    <DESCRIPTN/>
    <PORELEASE/>
    <USERAREA><DATETIME qualifier="ACTSTART" type="T" index="1"
><YEAR/><MONTH/><DAY/><HOUR/><MINUTE/><SECOND/><SUBSECOND/><TIMEZONE/>
</DATETIME><DATETIME qualifier="ACTEND" type="T"
index="1"><YEAR/><MONTH/><DAY/><HOUR/><MINUTE/><SECOND/><SUBSECOND/><T
IMEZONE/></DATETIME><DATETIME><FOB><DESCRIPTN>Vendor's responsibility
ceases upon transfer to
carrier</DESCRIPTN><TERMID>Origin</TERMID></FOB><TANDC/><FTTERM><DESCR
IPTN>Buyer pays
freight</DESCRIPTN><TERMID>Due</TERMID></FTTERM><EXCHRATE/><DATETIMEqu
alifier="EXCHRATE"
><YEAR>2011</YEAR><MONTH>04</MONTH><DAY>05</DAY><HOUR>00</HOUR><MINUTE
>0
0</MINUTE><SECOND>00</SECOND><SUBSECOND>0000</SUBSECOND><TIMEZONE>+000
0</TIMEZONE></DATETIME><DATETIME ualifier="APPREQ" type="T" index="1"
><YEAR/><MONTH/><DAY/><HOUR/><MINUTE/><SECOND/><SUBSECOND/><TIMEZONE/>
</
DATETIME><CONFIRM>N</CONFIRM><SHIPPINGCONTROL/><DFFPOHEADER><ATTRIBUTE
1/><ATTRIBUTE2/><ATTRIBUTE3/><ATTRIBUTE4/><ATTRIBUTE5/><ATTRIBUTE6/><A
TTRI
BUTE7/><ATTRIBUTE8/><ATTRIBUTE9/><ATTRIBUTE10/><ATTRIBUTE11/><ATTRIBUT
E1
2/><ATTRIBUTE13/><ATTRIBUTE14/><ATTRIBUTE15/><ATTRIBUTE16/></DFFPOHEAD
ER><PCARDHDR><MEMBERNAME/><PCARDNUM>0</PCARDNUM><DATETIME qualifier="

```

```

EXPIRATION"
><YEAR>2011</YEAR><MONTH>04</MONTH><DAY>05</DAY><HOUR>00</HOUR><MINUTE>0
0</MINUTE><SECOND>00</SECOND><SUBSECOND>0000</SUBSECOND><TIMEZONE>+000
0</TIMEZONE></DATETIME><PCARDBRAND/></PCARDHDR></USERAREA>
  <PARTNER>
    <NAME index="1">Hilman and Associates</NAME>
    <ONETIME>0</ONETIME>
    <PARTNRID><PARTNRID/>
    <PARTNRTYPE>Supplier</PARTNRTYPE>
    <CURRENCY>USD</CURRENCY>
    <DUNSNUMBER/>
    <PARTNRIDX>HIL-Tulsa</PARTNRIDX>
    <TAXID/>
    <TERMID/>
  <USERAREA><DFFVENDOR><ATTRIBUTE1/><ATTRIBUTE2/><ATTRIBUTE3/><ATTRIBUTE
4/><ATTRIBUTE5/><ATTRIBUTE6/><ATTRIBUTE7/><ATTRIBUTE8/><ATTRIBUTE9/><A
TTIBUTE10/><ATTRIBUTE11/><ATTRIBUTE12/><ATTRIBUTE13/><ATTRIBUTE14/><A
TTIBU TE15/><ATTRIBUTE16/></DFFVENDOR><CUSTOMERNUM/></USERAREA>
    <ADDRESS>
      <ADDRLINE index="1"></ADDRLINE>
      <ADDRLINE index="2"/>
      <ADDRLINE index="3"/>
      <ADDRTYPE/>
      <CITY></CITY>
      <COUNTRY></COUNTRY>
      <COUNTY/>
      <DESCRIPTN></DESCRIPTN>
      <FAX index="1"/>
      <POSTALCODE></POSTALCODE>
      <REGION/>
      <STATEPROVN></STATEPROVN>
      <TAXJRSRCTN/>
      <TELEPHONE index="1"></TELEPHONE>
      <TELEPHONE index="2"/>
      <TELEPHONE index="3"/>
      <URL/>
    <USERAREA><DFFVENDORSITE><ATTRIBUTE1/><ATTRIBUTE2/><ATTRIBUTE3/><ATTRI
BUTE4/><ATTRIBUTE5/><ATTRIBUTE6/><ATTRIBUTE7/><ATTRIBUTE8/><ATTRIBUTE9
/><ATTRIBUTE10/><ATTRIBUTE11/><ATTRIBUTE12/><ATTRIBUTE13/><ATTRIBUTE14
/><ATT RIBUTE15/><ATTRIBUTE16/></DFFVENDORSITE></USERAREA>
    </ADDRESS>
    <CONTACT>
      <NAME index="1"/>
      <EMAIL/>
      <FAX index="1"></FAX>
      <TELEPHONE index="1"></TELEPHONE>
    </CONTACT>
  </PARTNER>
  <PARTNER>
    <NAME index="1">Vision Operation</NAME>
    <ONETIME>0</ONETIME>
    <PARTNRID>204<PARTNRID/>
  <PARTNRTYPE>SoldTo</PARTNRTYPE>
    <CURRENCY>USD</CURRENCY>
    <DUNSNUMBER/>
    <PARTNRIDX>HIL-Tulsa</PARTNRIDX>
    <PAYMETHOD/>
    <TAXID/>
    <TERMID/>
    <USERAREA/>
    <ADDRESS>
    </ADDRESS>
    <CONTACT>
      <NAME index="1">Stock, Ms. Pat</NAME>

```

```

<CONTACTTYPE/>
  <DESCRIPTN/>
  <EMAIL>pstock@example.com</EMAIL>
  <FAX index="1"/>
  <TELEPHONE index="1"/>
  <USERAREA/>
</CONTACT>
</PARTNER>
<PARTNER>
  <NAME index="1">Vision Operation</NAME>
  <ONETIME>0</ONETIME>
  <PARTNRID>204</PARTNRID/>
  <PARTNRTYPE>BillTo</PARTNRTYPE>
  <CURRENCY/>
  <DUNSNUMBER/>
  <PARTNRIDX>HIL-Tulsa</PARTNRIDX>
  <PAYMETHOD/>
  <TERMD/>
  <USERAREA/>
  <ADDRESS>
  </ADDRESS>
</PARTNER>
<PARTNER>
  <NAME index="1">UPS</NAME>
  <ONETIME>0</ONETIME>
  <PARTNRID>204</PARTNRID/>
  <PARTNRTYPE>Carrier</PARTNRTYPE>
  <PARTNRIDX>UPS</PARTNRIDX>
</PARTNER>
<POTERM>
  <DESCRIPTN>Scheduled for payment 30 days from the invoice date
(invoice terms date = system date, goods received date, invoice date
or invoice received date). Invoice terms date can default from
supplier header, site, PO, system default, etc.</DESCRIPTN>
  <TERMD>30 Net (terms date + 30)</TERMD>
  <DAYSNUM/>
  <QUANTITY qualifier="PERCENT">
    <VALUE/>
    <NUMOFDEC/>
    <SIGN/>
    <UOM/>
  </QUANTITY>
  <USERAREA/>
</POTERM>
</POORDERHDR>
<POORDERLIN>
  <QUANTITY qualifier="ORDERED">
    <VALUE>12</VALUE>
    <NUMOFDEC/>
    <SIGN>+</SIGN>
    <UOM>Ea</UOM>
  </QUANTITY>
  <OPERAMT qualifier="UNIT" type="T">
    <VALUE>110786</VALUE>
    <NUMOFDEC>2</NUMOFDEC>
    <SIGN>+</SIGN>
    <CURRENCY>USD</CURRENCY>
    <UOMVALUE>1</UOMVALUE>
    <UOMNUMDEC>0</UOMNUMDEC>
    <UOM>Ea</UOM>
  </OPERAMT>
  <POLINENUM>1</POLINENUM>
  <HAZRDMATL/>
  <ITEMRV></ITEMRV>
  <ITEMRVX/>
  <POLNSTATUS/>

```

```

<DESCRIPTN></DESCRIPTN>
  <ITEM>AS54888</ITEM>
  <ITEMX/>
  <USERAREA><REQUESTOR/><CATEGORYID>PRODUCTN.
  FINGOODS</CATEGORYID><CONTRACTNUM/><CONTRACTPONUM/><CONTRACTPOLINENUM/
  >< VENDORQUOTENUM/><CONFIGID/><LISTPRICE>1107.
86
</LISTPRICE><MARKETPRICE>0</MARKETPRICE><PRICENOTTOEXCEED/><NEGPRICE>
N</NEGPRICE><TAXABLE>N</TAXABLE><TXNREASONCODE/><TYPE1099>AVAN/N</TYPE
1099><LINEORDERTYPE>Goods</LINEORDERTYPE><HAZRDUNNUM/><HAZRDUNDESC/><D
FFLINE><ATTRIBUTE1/><ATTRIBUTE2/><ATTRIBUTE3/><ATTRIBUTE4/><ATTRIBUTE5
/><ATTRIBUTE6/><ATTRIBUTE7/><ATTRIBUTE8/><ATTRIBUTE9/><ATTRIBUTE10/><A
TTRIBUTE11/><ATTRIBUTE12/><ATTRIBUTE13/><ATTRIBUTE14/><ATTRIBUTE15/><A
TTRIBUTE16/></DFFLINE><DFFITEM><ATTRIBUTE1/><ATTRIBUTE2/><ATTRIBUTE3/>
<ATTRIBUTE4/><ATTRIBUTE5/><ATTRIBUTE6/><ATTRIBUTE7/><ATTRIBUTE8/><ATTR
IBUTE9/><ATTRIBUTE10/><ATTRIBUTE11/><ATTRIBUTE12/><ATTRIBUTE13/><ATTRI
BUTE14/><ATTRIBUTE15/><ATTRIBUTE16/></DFFITEM><KFFITEM><ATTRIBUTE1>PRO
DUCTN</ATTRIBUTE1><ATTRIBUTE2>FINGOODS</ATTRIBUTE2><ATTRIBUTE3><ATTRI
BUTE4><ATTRIBUTE5><ATTRIBUTE6><ATTRIBUTE7><ATTRIBUTE8><ATTRIBUTE9
/><ATTRIBUTE10><ATTRIBUTE11/><ATTRIBUTE12/><ATTRIBUTE13/><ATTRIBUTE14
/><ATTRIBUTE15/><ATTRIBUTE16/><ATTRIBUTE17/><ATTRIBUTE18/><ATTRIBUTE19
/><ATTRIBUTE20/></KFFITEM><GLOBALCONTRACT/><GLOBALCONTRACTLIN/><JOBTIT
LE/><AMOUNT qualifier="TOTAL" type="T"
><VALUE/><NUMOFDEC/><SIGN/><CURRENCY/><DRCR>C</DRCR></AMOUNT><CONTRACT
OR FIRSTNAME/><CONTRACTORLASTNAME/><DATETIME qualifier="ACTSTART"
type="T" index="1"
><YEAR/><MONTH/><DAY/><HOUR/><MINUTE/><SECOND/><SUBSECOND/><TIMEZONE/>
</DATETIME><DATETIME qualifier="ACTEND" type="T" index="1"
><YEAR/><MONTH/><DAY/><HOUR/><MINUTE/><SECOND/><SUBSECOND/><TIMEZONE/>
</ DATETIME></USERAREA>
  <POLINESCHD>
  <DATETIME qualifier="NEEDELV" type="T" index="1">
    <YEAR>2011</YEAR>
    <MONTH>04</MONTH>
    <DAY>21</DAY>
    <HOUR>00</HOUR>
    <MINUTE>00</MINUTE>
    <SECOND>00</SECOND>
    <SUBSECOND>0000</SUBSECOND>
    <TIMEZONE>+0000</TIMEZONE>
  </DATETIME>
  <QUANTITY qualifier="ORDERED">
    <VALUE>12</VALUE>
    <NUMOFDEC/>
    <SIGN>+</SIGN>
    <UOM>Ea</UOM>
  </QUANTITY>
  <DESCRIPTN/>
  <PSCLINENUM>1</PSCLINENUM>
  <USERAREA><DATETIME qualifier="PROMSHIP"
  ><YEAR/><MONTH/><DAY/><HOUR/><MINUTE/><SECOND/><SUBSECOND/><TIMEZONE/>
  </ DATETIME><DATETIME qualifier="APPROVAL"
  ><YEAR/><MONTH/><DAY/><HOUR/><MINUTE/><SECOND/><SUBSECOND/><TIMEZONE/>
  </DATETIME><OPERAMT qualifier="UNIT" type="T"
  ><VALUE>110786</VALUE><NUMOFDEC>2</NUMOFDEC><SIGN>+</SIGN><CURRENCY>US
  D</CURRENCY><UOMVALUE>1</UOMVALUE><UOMNUMDEC>0</UOMNUMDEC><UOM>Ea</UOM
  ></O PERAMT><PRICEOVRD/><TAXABLE>N</TAXABLE><TAXCODE/><PARTNER><NAME
  index="1">Vision
  Operations</NAME><ONETIME>0</ONETIME><PARTNRID>204</PARTNRID><PARTNRTY
  PE>ShipTo</PARTNRTYPE><CURRENCY>USD</CURRENCY><DUNSNUMBER/><PARTNRIDX>
  HIL- Tulsa</PARTNRIDX><PAYMETHOD/><TERMID/><USERAREA/>
  <ADDRESS></ADDRESS>
  <CONTACT><NAME index="1"/><CONTCTTYPE/><EMAIL/><FAX index="1"
  /><TELEPHONE index="1"/></CONTACT>
  </PARTNER>

```

```

    <PARTNER><NAME index="1"
/><ONETIME/><PARTNRID/><PARTNRTYPE>DeliveryTo</PARTNRTYPE><PARTNRID/>
<USERAREA/><ADDRESS><ADDRLINE index="1"/><ADDRLINE index="2"
/><ADDRTYPE/></ADDRESS><CONTACT><NAME index="1"
/><CONCTTYPE/><EMAIL/><FAX index="1"/><TELEPHONE index="1"/>
</CONTACT></PARTNER><DROPSHIPDETAILS><DROPSHIPMENT/><DROPSHIPCUSTNAME/
><SHIPINSTR/><PACKINSTR/><SHIPMETHOD/><CUSTOMERPONUM/><CUSTOMERLINENUM
/><CUSTOMERSHIPNUM/><CUSTOMERDESC/></DROPSHIPDETAILS><CONSIGNEDINV>N</
CONSIGNEDINV><DISTPROJECT><REQUESTOR/><DISTNUM>1</DISTNUM><PROJECTNUM/
><PROJEC TTYPE/><TASKNUM/><QUANTITY qualifier="ORDERED"
><VALUE>12</VALUE><NUMOFDEC/><SIGN>+</SIGN><UOM>Ea</UOM></QUANTITY><CO
NV RATE/><DATETIME qualifier="EXCHRATEDATE"
><YEAR>2011</YEAR><MONTH>04</MONTH><DAY>05</DAY><HOUR>00</HOUR><MINUTE
>00</MINUTE><SECOND>00</SECOND><SUBSECOND>0000</SUBSECOND><TIMEZONE>+0
000</TIMEZONE></DATETIME><DESTTYPE>INVENTORY</DESTTYPE><DFFDISTRIBUTN>
<ATTRIBUTE1/><ATTRIBUTE2/><ATTRIBUTE3/><ATTRIBUTE4/><ATTRIBUTE5/><ATTR
IBUTE6/><ATTRIBUTE7/><ATTRIBUTE8/><ATTRIBUTE9/><ATTRIBUTE10/><ATTRIBUT
E11/><ATTRIBUTE12/><ATTRIBUTE13/><ATTRIBUTE14/><ATTRIBUTE15/><ATTRIBUT
E16/></DFFDI STRIBUTN></DISTPROJECT><AMOUNT qualifier="TOTAL" type="T"
><VALUE/><NUMOFDEC/><SIGN/><CURRENCY/><DRCR>C</DRCR></AMOUNT></USERAREA>

    </POLINESCHD>
    </POORDERLIN>
    <POORDERLIN>
        <QUANTITY qualifier="ORDERED">
            <VALUE>13</VALUE>
            <NUMOFDEC/>
            <SIGN>+</SIGN>
            <UOM>Ea</UOM>
        </QUANTITY>
        <OPERAMT qualifier="UNIT" type="T">
            <VALUE>110786</VALUE>
            <NUMOFDEC>2</NUMOFDEC>
            <SIGN>+</SIGN>
            <CURRENCY>USD</CURRENCY>
            <UOMVALUE>1</UOMVALUE>
            <UOMNUMDEC>0</UOMNUMDEC>
            <UOM>Ea</UOM>
        </OPERAMT>
        <POLINENUM>2</POLINENUM>
        <HAZRDMATL/>
        <ITEMRV></ITEMRV>
        <ITEMRVX/>
        <POLNSTATUS/>
        <DESCRIPTN></DESCRIPTN>
        <ITEM>ASO0021</ITEM>
        <ITEMX/>
        <USERAREA><REQUESTOR/><CATEGORYID>PRODUCTN.
FINGOODS</CATEGORYID><CONTRACTNUM/><CONTRACTPONUM/><CONTRACTPOLINENUM/
>< VENDORQUOTENUM/><CONFIGID/><LISTPRICE>1107.
86
</LISTPRICE><MARKETPRICE>0</MARKETPRICE><PRICENOTTOEXCEED/><NEGPRICE>
N</NEGPRICE><TAXABLE>N</TAXABLE><TXNREASONCODE/><TYPE1099>AVAN/N</TYPE
1099><LINEORDERTYPE>Goods</LINEORDERTYPE><HAZRDUNNUM/><HAZRDUNDESC/><D
FFLINE><ATTRIBUTE1/><ATTRIBUTE2/><ATTRIBUTE3/><ATTRIBUTE4/><ATTRIBUTE5
/><ATTRIBUTE6/><ATTRIBUTE7/><ATTRIBUTE8/><ATTRIBUTE9/><ATTRIBUTE10/><A
TTRIBUTE11/><ATTRIBUTE12/><ATTRIBUTE13/><ATTRIBUTE14/><ATTRIBUTE15/><A
TTRIBUTE16/></DFFLINE><DFFITEM><ATTRIBUTE1/><ATTRIBUTE2/><ATTRIBUTE3/>
<ATTRIBUTE4/><ATTRIBUTE5/><ATTRIBUTE6/><ATTRIBUTE7/><ATTRIBUTE8/><ATTR
IBUTE9/><ATTRIBUTE10/><ATTRIBUTE11/><ATTRIBUTE12/><ATTRIBUTE13/><ATTRI
BUTE14/><ATTRIBUTE15/><ATTRIBUTE16/></DFFITEM><KFFITEM><ATTRIBUTE1>PRO
DUCTN</ATTRIBUTE1><ATTRIBUTE2>FINGOODS</ATTRIBUTE2><ATTRIBUTE3><ATTRI
BUTE4><ATTRIBUTE5><ATTRIBUTE6><ATTRIBUTE7><ATTRIBUTE8><ATTRIBUTE9
/><ATTRIBUTE10><ATTRIBUTE11><ATTRIBUTE12><ATTRIBUTE13><ATTRIBUTE14
/><ATTRIBUTE15><ATTRIBUTE16><ATTRIBUTE17><ATTRIBUTE18><ATTRIBUTE19
/><ATTRIBUTE20></KFFITEM><GLOBALCONTRACT/><GLOBALCONTRACTLIN/><JOBTIT
LE/><AMOUNT qualifier="TOTAL" type="T"

```



```

><VALUE/><NUMOFDEC/><SIGN/><CURRENCY/><DRCR>C</DRCR></AMOUNT><CONTRACT
OR FIRSTNAME/><CONTRACTORLASTNAME/><DATETIME qualifier="ACTSTART"
type="T" index="1"
><YEAR/><MONTH/><DAY/><HOUR/><MINUTE/><SECOND/><SUBSECOND/><TIMEZONE/>
</DATETIME><DATETIME qualifier="ACTEND" type="T" index="1"
><YEAR/><MONTH/><DAY/><HOUR/><MINUTE/><SECOND/><SUBSECOND/><TIMEZONE/>
</ DATETIME></USERAREA>
  <POLINESCHD>
    <DATETIME qualifier="NEEDELV" type="T" index="1">
      <YEAR>2011</YEAR>
      <MONTH>04</MONTH>
      <DAY>21</DAY>
      <HOUR>00</HOUR>
      <MINUTE>00</MINUTE>
      <SECOND>00</SECOND>
      <SUBSECOND>0000</SUBSECOND>
      <TIMEZONE>+0000</TIMEZONE>
    </DATETIME>
    <QUANTITY qualifier="ORDERED">
      <VALUE>12</VALUE>
      <NUMOFDEC/>
      <SIGN>+</SIGN>
      <UOM>Ea</UOM>
    </QUANTITY>
    <DESCRIPTN/>
    <PSCLINENUM>1</PSCLINENUM>
    <USERAREA><DATETIME qualifier="PROMSHIP"
><YEAR/><MONTH/><DAY/><HOUR/><MINUTE/><SECOND/><SUBSECOND/><TIMEZONE/>
</ DATETIME><DATETIME qualifier="APPROVAL"
><YEAR/><MONTH/><DAY/><HOUR/><MINUTE/><SECOND/><SUBSECOND/><TIMEZONE/>
</DATETIME><OPERAMT qualifier="UNIT" type="T"
><VALUE>110786</VALUE><NUMOFDEC>2</NUMOFDEC><SIGN>+</SIGN><CURRENCY>US
D</CURRENCY><UOMVALUE>1</UOMVALUE><UOMNUMDEC>0</UOMNUMDEC><UOM>Ea</UOM>
></O PERAMT><PRICEOVRD/><TAXABLE>N</TAXABLE><TAXCODE/><PARTNER><NAME
index="1">Vision
Operations</NAME><ONETIME>0</ONETIME><PARTNRID>204</PARTNRID><PARTNRTY
PE>ShipTo</PARTNRTYPE><CURRENCY>USD</CURRENCY><DUNSNUMBER/><PARTNRIDX>
HIL- Tulsa</PARTNRIDX><PAYMETHOD/><TERMID/><USERAREA/>
  <ADDRESS></ADDRESS>
  <CONTACT><NAME index="1"/><CONTCTTYPE/><EMAIL/><FAX index="1"
/><TELEPHONE index="1"/></CONTACT>
  </PARTNER>
  <PARTNER><NAME index="1"
/><ONETIME/><PARTNRID/><PARTNRTYPE>DeliveryTo</PARTNRTYPE><PARTNRIDX/>
<U SERAREA/><ADDRESS><ADDRLINE index="1"/><ADDRLINE index="2"
/><ADDRTYPE/></ADDRESS><CONTACT><NAME index="1"
/><CONTCTTYPE/><EMAIL/><FAX index="1"/><TELEPHONE index="1"
/></CONTACT></PARTNER><DROPSHIPDETAILS><DROPSHIPMENT/><DROPSHIPCUSTNAM
E/><SHIPINSTR/><PACKINSTR/><SHIPMETHOD/><CUSTOMERPONUM/><CUSTOMERLINEN
UM/><CUSTOMERSHIPNUM/><CUSTOMERDESC/></DROPSHIPDETAILS><CONSIGNEDINV>N
</CONSIGNEDINV><DISTPROJECT><REQUESTOR/><DISTNUM>1</DISTNUM><PROJECTNU
M/><PROJ ECTTYPE/><TASKNUM/><QUANTITY qualifier="ORDERED"
><VALUE>12</VALUE><NUMOFDEC/><SIGN>+</SIGN><UOM>Ea</UOM></QUANTITY><CO
NV RATE/><DATETIME qualifier="EXCHRATEDATE"
><YEAR>2011</YEAR><MONTH>04</MONTH><DAY>05</DAY><HOUR>00</HOUR><MINUTE
>00</MINUTE><SECOND>00</SECOND><SUBSECOND>0000</SUBSECOND><TIMEZONE>+0
000</TIMEZONE></DATETIME><DESTTYPE>INVENTORY</DESTTYPE><DFFDISTRIBUTN>
<ATTRIBUTE1/><ATTRIBUTE2/><ATTRIBUTE3/><ATTRIBUTE4/><ATTRIBUTE5/><ATTR
IBUTE6/><ATTRIBUTE7/><ATTRIBUTE8/><ATTRIBUTE9/><ATTRIBUTE10/><ATTRIBUT
E11/><ATTRIBUTE12/><ATTRIBUTE13/><ATTRIBUTE14/><ATTRIBUTE15/><ATTRIBUT
E16/></DFFDI STRIBUTN></DISTPROJECT><AMOUNT qualifier="TOTAL" type="T"
><VALUE/><NUMOFDEC/><SIGN/><CURRENCY/><DRCR>C</DRCR></AMOUNT></USERAREA>

  </POLINESCHD>
  </POORDERLIN>
  </PROCESS_PO>
  </DATAAREA>
</PROCESS_PO_007>

```

用語集

エージェント

システム内通信の指定地点。

エージェント・リスナー

インバウンド・エージェントに対するイベント・メッセージを処理するサービス・コンポーネントのタイプ。

非同期操作

結果を即時に取得する同期サービス実行と異なり、非同期操作では要求の処理に非常に長い時間が必要になる場合があります。

ただし、Oracle E-Business Suite Web サービスを起動したクライアントは、応答を待機するのではなく、その間に他の処理を続けることができます。

BPEL

ビジネス・プロセス実行言語(BPEL)では、実行可能で抽象的なビジネス・プロセスの指定のための言語が提供されます。このことによって、サービス相互作用モデルが拡張され、ビジネス・トランザクションのサポートが可能になります。BPEL では、企業内と企業間の両方の領域で自動化されたプロセス統合の拡張を促進する、相互運用可能な統合モデルが定義されます。

ビジネス・イベント

「イベント」を参照してください。

コールバック・パターン

コールバック・パターンは、非同期サービスにおける重要な通信方法です。非同期コールバックは、サービス・プロバイダに対して要求が行われ、準備が整ったときに応答(コールバック)が要求者に返されることを意味します。このパターンを確認とともに使用して、要求者が送信した要求の受信を認識できます。

コンカレント・マネージャ

要求のキューイングおよびコンカレント・プログラムの操作を管理する Oracle E-Business Suite のコンポーネント。

コンカレント・プログラム

コンカレント・プログラムは、特定のタスク(仕訳の転記、レポートの生成など)を実行する実行ファイルです。

イベント

インターネットまたはイントラネット・アプリケーションやプログラム内における、システム内の他のオブジェクトや外部エージェントに影響を及ぼす可能性がある事象。

イベント・アクティビティ

ワークフロー・プロセスに含めることができるようにアクティビティとしてモデル化されたビジネス・イベント。

イベント・データ

イベントを説明する一連の追加詳細。イベント・データは XML 文書として構造化できます。イベント名、イベント・キーおよびイベント・データの組み合わせにより、イベントで発生したことが詳細に伝達されます。

イベント・キー

イベントのインスタンスを一意に識別する文字列。イベント名、イベント・キーおよびイベント・データの組み合わせにより、イベントで発生したことが詳細に伝達されます。

イベント・メッセージ

ビジネス・イベントを通信するための標準的なワークフロー構造。データ型 WF_EVENT_T によって定義されます。イベント・メッセージには、イベント・データおよび複数のヘッダー・プロパティ(イベント名、イベント・キー、アドレッシング属性、エラー情報など)が含まれています。

イベント・サブスクリプション

特定のイベントがシステムに対して重要であることを示し、トリガー・イベントが発生したときに実行する処理を指定する登録。サブスクリプション処理には、カスタム・コードのコール、ワークフロー・プロセスへのイベント・メッセージの送信またはエージェントへのイベント・メッセージの送信を含めることができます。

ファンクション

ビジネス・ルールの定義、アプリケーション内での自動タスクの実行またはアプリケーション情報の取得が可能な PL/SQL ストアド・プロシージャ。ストアド・プロシージャは、標準的な引数を受け入れて完了結果を返します。

統合リポジトリ

Oracle Integration Repository は、Oracle E- Business Suite Integrated SOA Gateway の重要なコンポーネントまたはユーザー・インタフェースです。この集中管理されたリポジトリには、ネイティブのパッケージ化された統合インタフェース定義およびコンポジット・サービスが格納されます。

統合リポジトリ・パーサー

アノテーションの標準に対して注釈付けされたカスタム・インタフェース定義を検証して統合リポジトリ・ローダー・ファイル(iLDT)を生成するために、統合管理者によって使用されるスタンドアロンの設計時ツールです。この生成された iLDT ファイルは、カスタム・インタフェースをすべてのユーザーに公開できる統合リポジトリにアップロードできます。

インタフェース・タイプ

統合インタフェースは、異なるインタフェース・タイプにグループ化されます。

JSON

JSON (JavaScript Object Notation)は、人間が判読可能なデータ交換用に設計されたテキストベースのオープン標準です。JSON 形式は、多くの場合、REST サービスで、サーバーと Web アプリケーションの間で構造化データを伝送するために使用され、XML の代替方法となるものです。

疎結合

疎結合は、なんらかの交換関係を持つ 2 つ以上のシステムまたは組織間の、レジリエンスのある関係を示します。取引のそれぞれの側は要件を明確にし、相手側に関する想定をほぼなくします。

参照コード

参照タイプで定義された値の内部名。

参照タイプ

事前定義された値リスト。参照タイプのそれぞれの値には、内部名と表示名があります。

メッセージ

通知アクティビティによって送信される情報。メッセージを通知アクティビティに関連付けるには、まず定義する必要があります。メッセージには、件名、優先度、本文、および場合によっては 1 つ以上のメッセージ属性が含まれています。

メッセージ属性

通知内でメッセージを送信するとき、情報を提供するか、応答を要求するプロンプトを表示するために、特定のメッセージに対して定義する変数。事前定義項目タイプ属性をメッセージ属性として使用できます。送信ソースとして定義されると、メッセージが送信されるとき、メッセージ属性はランタイム値に置き換えられます。応答ソースとして定義されると、メッセージが送信されるとき、メッセージ属性によってユーザーに応答を要求するプロンプトが表示されます。

通知

ユーザーに配信されるメッセージのインスタンス。

通知ワークリスト

ワークフロー通知の問合せや応答のためにアクセスできる Web ページ。

操作

サービスによってサポートされる処理の抽象的な説明。

ポート

ポートは、バインディング用の単一のアドレスを指定することで個別のエンドポイントを定義します。

ポート・タイプ

ポート・タイプは、関連する抽象操作および抽象メッセージの名前付きセットです。

プロセス

ビジネス目標を達成するために実行する必要がある一連のアクティビティ。

REST

Representational State Transfer (REST)は、Web サービスをリソースとみなし、URL で一意に識別できるとするアーキテクチャ原則です。REST サービスの重要な特性は、様々な操作の起動を示す HTTP メソッド(GET、POST、PUT および DELETE)の明示的な使用です。

SAML トークン(Sender-Vouches)

このタイプのセキュリティ・モデルでは、Security Assertion Markup Language (SAML)アサーションによってのみユーザー名を送信することで、Web サービスを認証します。

SAML は、セキュリティ・ドメイン間、つまりアイデンティティ・プロバイダとサービス・プロバイダとの間で認証データおよび認可データを交換するための XML ベースの標準です。SAML トークンは、sender-vouches 方式を使用して、SOAP メッセージと SOAP メッセージに追加された SAML アサーションの間で通信を確立します。

「ユーザー名トークン」を参照してください。

Service

サービスは、関連するエンドポイントのコレクションです。

サービス・コンポーネント

一般サービス・コンポーネント・フレームワークで管理できるように、この標準に従って定義された Java プログラムのインスタンス。

サービス・モニター

これは Oracle E-Business Suite のモニタリングと監査のツールとなり、これを使用して、Oracle E-Business Suite Integrated SOA Gateway によって提供される Web サービスのランタイム・メッセージを表示できます。

以前のリリースでは SOA モニターと呼ばれています。

SOA

サービス指向アーキテクチャ(SOA)は、相互作用するソフトウェア・コンポーネント間の疎結合を実現し、異種 IT エコシステムでのシームレスで標準に基づいた統合を可能にするアーキテクチャです。

SOAP

Simple Object Access Protocol (SOAP)は、非集中的な分散環境での構造化された情報の交換を目的とした軽量プロトコルです。基礎となる様々なプロトコルで交換できるメッセージ構造を提供する拡張可能なメッセージング・フレームワークを定義するために、XML テクノロジが使用されます。

サブスクリプション

「イベント・サブスクリプション」を参照してください。

同期操作

同期操作では、問合せに対する即時の応答が提供されます。この状況では、サーバーに要求が送信された時点からクライアント接続が開いたままになります。クライアントは、サーバーが応答メッセージを返すまで待機します。

ユーザー名トークン

ユーザー名とパスワードに基づいて実行時に SOAP 要求を認証するセキュリティ・モデルのタイプ。

「SAML トークン(Sender-Vouches)」を参照してください。

WADL

Web Application Description Language (WADL)は、HTTP ベースの Web アプリケーションを、マシンが処理可能な記述で提供するために設計されています。サービスによって提供されるリソースおよびリソース間の関係がモデル化されます。

Web サービス

Web サービスは、ネットワークでの相互運用可能なマシン間相互作用をサポートするために設計されたソフトウェア・システムです。WSDL で記述されたインタフェースを備えています。他のシステムは、SOAP メッセージを使用した記述で規定された方法で Web サービスとやり取りします。SOAP メッセージは、通常、他の Web 関連の標準と組み合わせて XML シリアルライズで HTTP を使用して伝達されます。

ワークフロー・エンジン

ワークフロー・プロセス定義を実装する Oracle Workflow コンポーネント。ワークフロー・エンジンは、項目のすべてのアクティビティの状態を管理し、機能を自動的に実行して通知を送信し、完了したアクティビティの履歴を保持し、エラー状態を検出してエラー・プロセスを開始

します。ワークフロー・エンジンはサーバーPL/SQL に実装され、エンジン API がコールされたときにアクティブ化されます。

WSDL

Web Services Description Language (WSDL)は、ドキュメント指向またはプロシージャ指向の情報を含むメッセージを処理する一連のエンドポイントとしてネットワーク・サービスを記述するための XML 形式です。操作とメッセージは抽象的に記述され、具体的なネットワーク・プロトコルとメッセージ形式にバインドされてエンドポイントを定義します。

WS-Addressing

WS-Addressing は、SOAP メッセージ自体の内部で、メッセージの受信者(および送信者)のアドレスを記述する方法です。

WS セキュリティ

WS セキュリティは、HTTPS を使用したチャネルの保護の代替または拡張として、SOAP で XML 署名を使用してメッセージ交換を保護する方法を定義します。

XML

XML (Extensible Markup Language)は、人間とマシンの両方が判読可能な形式でドキュメントをエンコードするためのルール・セットを定義するマークアップ言語です。

索引

B

ビジネス・イベント

注釈付け, 11-18

ダウンロード, 11-15

データベースへのファイルのアップロード, 11-21

iLDT ファイルのアップロード, 11-22

検証, 11-20

ビジネス・サービス・オブジェクト設計タスク

Assign アクティビティの追加, 9-18

Invoke アクティビティの追加, 9-14

ファイル・アダプタのパートナ・リンクの追加, 9-9

パートナ・リンクの作成, 9-9

BPEL プロセスによる SOA コンポジット・アプリケーションの作成, 9-6

C

コンポジット・サービス

ダウンロード, 10-2

変更, 10-3

概要, 10-1

表示, 10-2

接続情報

アプリケーション・サーバー接続, B-1

カスタム・インタフェースの作成およびアップロード

ビジネス・イベント, 11-15

コンポジット・サービス検証, 11-13

作成, 11-5, 11-10

コンポジット・サービスの表示および管理, 11- 14

カスタム・インタフェースの作成および使用

作成ステップ, 11-2

概要, 11-1

カスタム・インタフェースの使用, 11-23

カスタム・インタフェースの作成

コンポジット・サービス, 11-9

インタフェース・タイプ, 11-2

ビジネス・イベントを使用した BPEL の作成

割当て, 6-21

新規 BPEL プロジェクトの作成, 6-3

パートナ・リンク AQ アダプタの作成, 6-5

パートナ・リンク・ファイル・アダプタの作成, 6-13

起動, 6-19

受入, 6-11

起動側イベント・サブスクリプションの作成

エラー・サブスクリプションの作成, 12-12

「Web サービスの起動」によるサブスクリプションの作成, 12-7

D

BPEL のデプロイおよびテスト

BPEL のデプロイ, 9-26

BPEL のテスト, 9-29

コンカレント・プログラムのデプロイおよびテスト

BPEL のデプロイ, 7-23, 7-26

カスタム BPEL のデプロイおよびテスト

BPEL のデプロイ, 11-39

BPEL のテスト, 11-42

イベント BPEL のデプロイおよびテスト

BPEL のデプロイ, 6-23

BPEL のテスト, 6-26

PL/SQL BPEL のデプロイおよびテスト

BPEL のデプロイ, 3-38, 3-64

BPEL のテスト, 3-43, 3-69

統合インタフェースの検出および表示

Web サービスの生成, 2-6

概要, 2-1

REST メッセージ, 2-44

詳細のレビュー, 2-5

WADL 詳細のレビュー, 2-18

WSDL 詳細のレビュー, 2-8

インタフェースの検索および表示, 2-1

SOAP メッセージ, 2-25

E

拡張性

addCustomSOAPHeaders, 12-50

addWSSSecurityHeader, 12-47

postInvokeService, 12-47

preInvokeService, 12-46

setInputParts, 12-48

I

統合リポジトリ・アノテーションの標準

アノテーション用語集, A-120

ビジネス・エンティティ, A-41

ビジネス・イベント, A-35

コンポジット・サービス - BPEL, A-115

コンカレント・プログラム, A-23

ガイドライン, A-1

Java, A-4

PL/SQL, A-17

XML ゲートウェイ, A-25

Web サービスの起動

例, 12-32

Web サービスの起動

Web サービス応答での Oracle E-Business Suite
へのコールバック, 12-27

Oracle Workflow による Web サービスの起動

概要, 12-1

SIF を使用した Web サービス起動, 12-2

カスタム Java Bean サービスの起動

カスタム Java API の作成およびコンパイル, 4-
17

セキュリティ付与の作成, 4-37

カスタム Java Bean サービスの配置, 4-36

カスタム Java クラスおよびソース・ファイルの配
置, 4-34

カスタム Java Bean サービスのアップロード, 4-34

Java Bean サービスの起動

WADL の記録, 4-38

トークン・ベース認証および XML ペイロードを使用
したアプリケーション・モジュール・サービスの起
動

Java クラスでのプロジェクトの作成, 4-52

セキュリティ付与の作成, 4-51

REST サービスの配置, 4-49

Java クラスを使用した REST サービスの起動, 4-
60

配置済 WADL URL の記録, 4-50

REST サービスの起動

Java クラスでのプロジェクトの作成, 4-9

REST サービスの配置, 4-2

WADL の記録, 4-4, 4-6

Web サービスの起動ステップ

受入イベントの作成, 12-14

起動および受入イベントの作成, 12-6

起動側ローカルおよびエラー・イベント・サブス
クリプションの作成, 12-7

J

REST ヘッダー付き JSON ペイロード

PL/SQL REST Web サービスの配置, 3-85

Java クライアントを使用した REST サービスの起
動(設計時), 3-88

Java クライアントを使用した REST サービスの起
動(実行時), 3-99

配置済 WADL URL の記録, 3-87

O

Oracle E-Business Suite Integrated SOA Gateway

コンポーネントの機能, 1-2

主な機能, 1-1

概要, 1-1

S

サンプル・ペイロード

インバウンド発注, C-3

仕入先出荷およびデビット要求, C-1

T

サービス起動のテスト

コマンドライン, 12-40

「ビジネス・イベントのテスト」ページ, 12-36

Web サービス起動失敗のトラブルシューティング

コンカレント・マネージャ(CM)層 JVM, 12-44
OACORE WebLogic Server, 12-41
スタンドアロン JVM, 12-45

U

SOAP メッセージについて

XML ゲートウェイ・メッセージの SOA ヘッダー, 2-34
アプリケーション・コンテキストの SOAP ヘッダー, 2-31

SOA プロバイダによる SOAP メッセージ, 2-38

SOAP セキュリティ・ヘッダー, 2-27

カスタム・インタフェース設計タスクの使用, 11-24

概要, 11-23

実行時タスク, 11-39

ビジネス・イベントの使用

BPEL のデプロイおよびテスト, 6-22

概要, 6-1

ビジネス・イベントの使用, 6-2

ビジネス・サービス・オブジェクトの使用

BPEL のデプロイおよびテスト, 9-26

概要, 9-1

ビジネス・サービス・オブジェクト WSDL の使用 (設計時), 9-2

コンカレント・プログラムの使用

設計タスク, 7-2

概要, 7-1

実行時タスク, 7-22

コンカレント・プログラム設計タスクの使用

ファイル・アダプタのパートナー・リンクの追加, 7-9

Assign アクティビティ, 7-16

新規 BPEL プロジェクトの作成, 7-6

パートナー・リンクの作成, 7-8

Invoke アクティビティ, 7-14

カスタム WSDL の使用

Assign アクティビティの追加, 11-34

Invoke アクティビティの追加, 11-33

カスタム WSDL の使用

ファイル・アダプタのパートナー・リンクの追加, 11-28

新規 BPEL プロジェクトの作成, 11-26

パートナー・リンクの作成, 11-28

REST サービスとしての Java API の使用

アプリケーション・モジュール・サービスの起動例, 4-48

Java Bean サービスの使用

カスタム Java Bean サービスの起動, 4-16

JavaScript を使用した HTML からのカスタム Java サービスの起動, 4-41

REST サービスの起動, 4-2

Java クラスを使用した REST サービスの起動, 4-15

概要, 4-1

オープン・インタフェース表の使用

セキュリティ付与の作成, 8-3

REST サービスの配置, 8-2

REST サービスの起動, 8-4

概要, 8-1

WADL の記録, 8-4

オープン・インタフェース表およびビューの使用

概要, 8-1

PL/SQL の使用

BPEL のデプロイおよびテスト, 3-37, 3-63

概要, 3-1

同期 BPEL プロセス, 3-6, 3-46

PL/SQL REST サービスの使用, 3-72

PL/SQL WSDL の使用, 3-2

PL/SQL REST サービスの使用

JSON ペイロード, 3-83

REST ヘッダー付き XML ペイロード, 3-73

REST ヘッダー付き XML ペイロード(実行時), 3-83

PL/SQL WSDL の使用

Assign アクティビティの追加, 3-23

Invoke アクティビティの追加, 3-19, 3-53

受入アクティビティの追加, 3-55

Assign アクティビティの追加, 3-57

ファイル・アダプタのパートナー・リンクの追加, 3-11, 3-52

新しい BPEL プロセスによる SOA コンポジット・アプリケーションの作成, 3-7, 3-49

パートナー・リンクの作成, 3-10

XML ゲートウェイの使用

BPEL のデプロイ, 5-55

概要, 5-1

BPEL のテスト, 5-58

XML ゲートウェイ・インバウンドの使用, 5-2

XML ゲートウェイ・インバウンドの使用

XML ゲートウェイ・インバウンドの使用(設計時), 5-2

XML ゲートウェイ・インバウンドの使用

新規 BPEL プロジェクトの作成, 5-7

SOA プロバイダによる XML ゲートウェイ・インバウンドの使用

割当て, 5-17

パートナー・リンクの作成, 5-11, 5-12

- 起動, 5-16
- XML ゲートウェイ・インバウンド SOA プロバイダ
実行時タスクの使用, 5-22
- XML ゲートウェイ・インバウンド SOA プロバイダ実
行時タスクの使用
 - デプロイ, 5-22, 5-26
- XML ゲートウェイ・アウトバウンドの使用
 - XML ゲートウェイ・アウトバウンドの使用, 5-32
- XML ゲートウェイ・アウトバウンドの使用
 - BPEL のデプロイおよびテスト, 5-54
- XML ゲートウェイ・アウトバウンド設計タスクの使用
 - Assign アクティビティの追加, 5-53
 - Invoke アクティビティの追加, 5-51
 - ファイル・アダプタのパートナー・リンクの追加, 5-48
 - 受入アクティビティの追加, 5-46
 - 新規 BPEL プロジェクトの作成, 5-36
 - AQ アダプタのパートナー・リンクの作成, 5-37
 - 概要, 5-32

W

- Web サービス起動
 - 考慮事項, 12-53
 - シード Java ルール関数の拡張, 12-46
 - Web サービス起動のテスト, 12-36
 - Web サービス起動失敗のトラブルシューティング,
12-40
- SIFを使用した Web サービス起動
 - Web サービスの起動, 12-28
 - メッセージ・パターン, 12-3
 - メタデータ定義, 12-5
 - WS セキュリティのサポート, 12-21
 - Web サービス入力メッセージ・パート, 12-16

X

- REST ヘッダー付き XML ペイロード
 - PL/SQL REST Web サービスの配置, 3-74
 - Java クライアントを使用した REST サービスの起
動, 3- 77
 - 配置済 WADL URL の記録, 3-76