

バージョン 6.0.1



## 製品概要



SD88-6686-02  
(英文原典：SC10-4208-03)

お願い

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

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

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

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

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

(URL は、変更になる場合があります)

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

原 典： SC10-4208-03  
WebSphere Integration Developer  
Technical product overview  
Version 6.0.1

発 行： 日本アイ・ビー・エム株式会社

担 当： ナショナル・ランゲージ・サポート

第1刷 2006.7

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

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

© Copyright International Business Machines Corporation 2006. All rights reserved.

© Copyright IBM Japan 2006

---

# 目次

## 第 1 章 ビジネス・インテグレーション . . . 1

ビジネス単位間の統合 . . . . .	2
企業間の統合 . . . . .	3
WebSphere Integration Developer . . . . .	3
標準 . . . . .	5
統合開発者のロール . . . . .	5

## 第 2 章 Service Component

### Architecture . . . . . 7

サービス・コンポーネント . . . . .	8
サービス・データ・オブジェクト . . . . .	10
サービス修飾子 . . . . .	11
モジュール . . . . .	12
インポートおよびエクスポート . . . . .	14
サービス・インポートおよびエクスポートのバインディング・タイプ . . . . .	14
適切なバインディングの選択 . . . . .	16
サービス実装タイプ . . . . .	17

Java オブジェクト . . . . .	17
BPEL プロセス . . . . .	18
ステート・マシン . . . . .	19
ビジネス・ルール . . . . .	20
セレクター . . . . .	21
ヒューマン・タスク . . . . .	22
インターフェース・マップ . . . . .	23
メディエーション・フロー . . . . .	24
スタンドアロン参照 . . . . .	25
関連情報 . . . . .	25

## 第 3 章 ツールについての学習 . . . . . 27

「ようこそ」ビューの「概要」 . . . . .	27
「ようこそ」ビュー内のチュートリアル . . . . .	28
「ようこそ」ビュー内のサンプル . . . . .	28
インフォメーション・センター . . . . .	29

## 特記事項 . . . . . 31

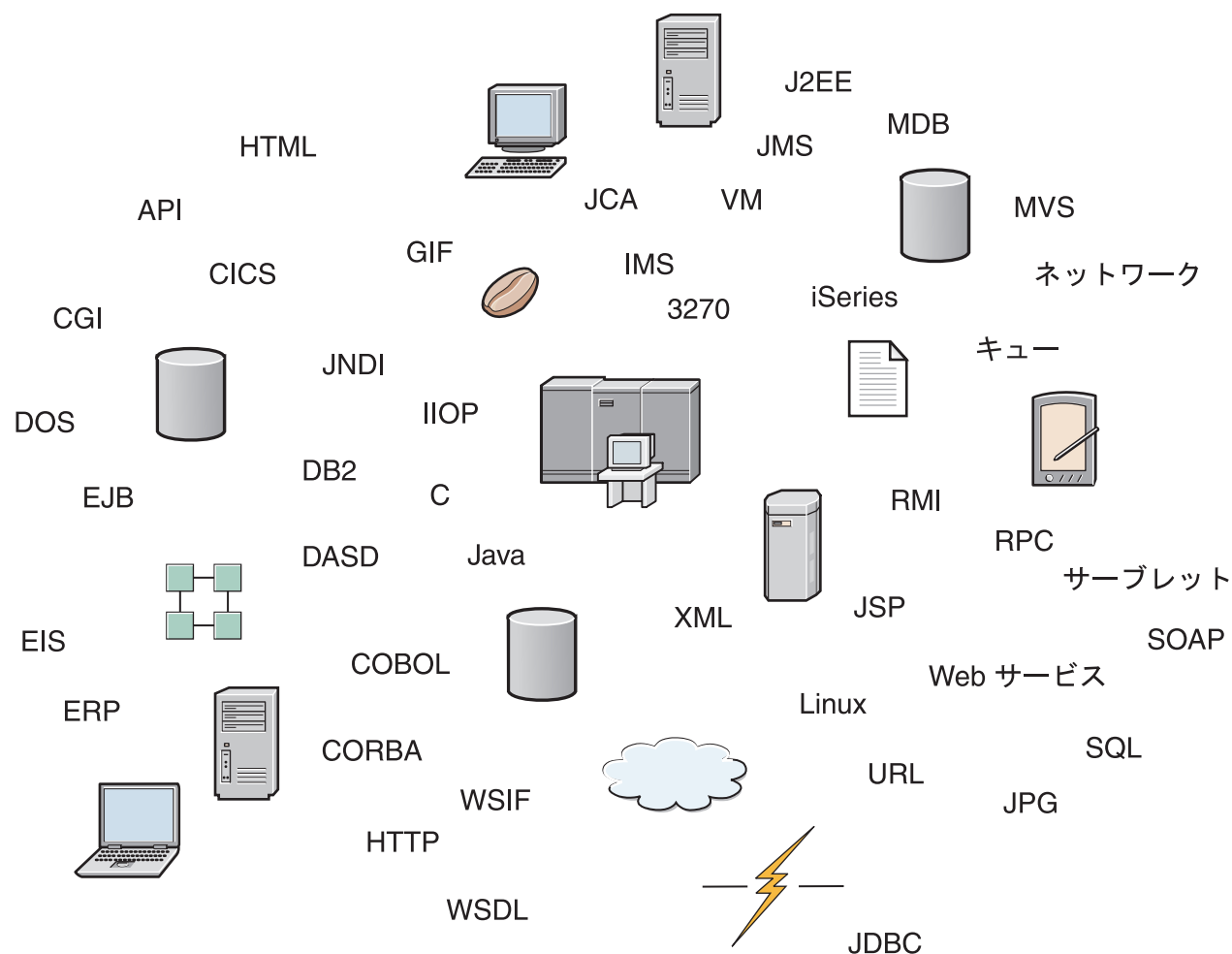


## 第 1 章 ビジネス・インテグレーション

ビジネス・インテグレーションとは、企業内または企業の集合でアプリケーション、データ、およびプロセスを統合することを意味します。このタスクの課題、およびその課題を WebSphere Integration Developer で対処する方法を検討します。

顧客のためにビジネス・アプリケーションのポータルを作成するように求められているとします。ユーザーのビジネス単位全体に広がる数十の主要ビジネス・アプリケーション、および関連データへのアクセスを提供する必要があります。また、ビジネス・パートナーのアプリケーションをポータルに追加することも求められています。統合とは、プロセスの開発も意味します。これは、それらの一連の組み立てられたアプリケーションにとって、何らかのロジックがあるからです。20 の事業単位と 12 のビジネス・パートナーがあるとします。ポータルは、一日 24 時間、Web で利用可能でなければなりません。自分自身を含めて 6 名の開発者がスタッフとして与えられ、4 カ月で立ち上げと稼働を行う必要があります。

その課題を与えられた大部分の人は、過去数十年にわたって蓄積されたテクノロジーを調べてみると、次の図のような、さまざまな断片の寄せ集めであることがわかります。



これは、手に負えないように見える状況かもしれませんが、対処が不可能な状況ではありません。直面する最も困難な問題は、ハードウェアとソフトウェアのバラバラの集合に加えて、自分の時間とリソースの制約です。関係する企業全体に分散したアプリケーションとデータを速やかにまとめるために、強力なツールを見つける必要があります。手作業によるコーディングは選択できません。

この状況に直面しているのは、自分だけでしょうか？ いいえ、そうではありません。これは、広く蔓延している長年の問題なのです。2001年12月のCIO調査によると、アプリケーション統合は、テクノロジー関連の重点事項のトップ3に常に入っています。2001年冬の「*The Business Integrator Journal*」によると、最近の調査で、開発者の3分の2が、Webベースのソリューションを開発するために統合ソフトウェアを使用していることがわかりました。平均して、1人の開発者が3つの異なるシステムを統合していました。

この問題を深刻な状況にしているビジネスの2つの力、すなわち企業内のビジネス単位間の統合と、企業間の統合を調べてみましょう。次に、WebSphere Integration Developerがどのようにこれらの問題に対処するか、そして特に業界標準に対するコミットメントの重要性を見ていきます。最後に、WebSphere Integration Developerのツールを使用して前述の問題を解決する担当者、つまり統合の専門家について検討します。

---

## ビジネス単位間の統合

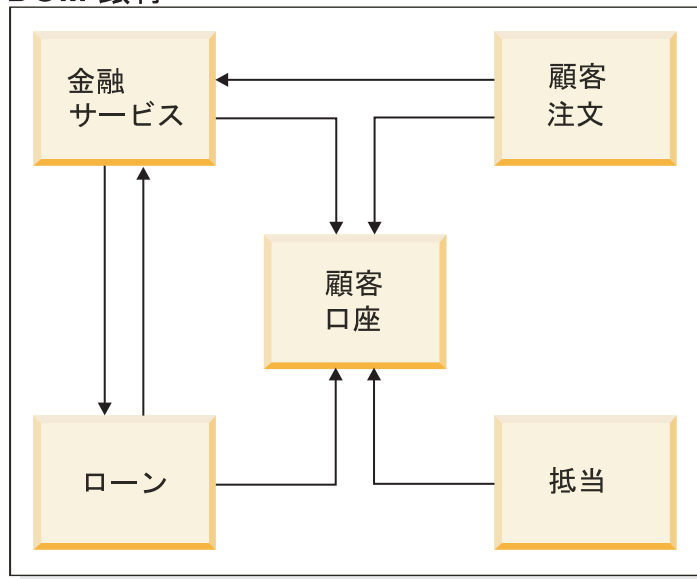
現在、事業単位のコラボレーションは頻繁にみられ、アプリケーションの緊密な統合の必要性が増えています。

以前は自律型であったビジネス単位が統合されつつあります。これは、テクノロジーにより、ビジネス単位が接続できるようになったこと、および効率化のためにオーバーヘッドを最小限に抑え、出力を最大化するために協調して運用する必要があることからです。一般的な企業目標も、事業単位の結合を促しています。マーケティング単位と研究開発単位はどちらも、収益性の高い製品の生産を求めています。マーケットの知識と製品開発の情報を統合することによって、そうした収益性の高い製品を生産する可能性が向上します。また、事業単位間のコラボレーションにより、企業は、さまざまなビジネス・コンテキストで既存のビジネス・アプリケーションを再使用することによって、多くのビジネス・アプリケーションを活用できるようになります。

ビジネス単位間での統合は、企業間での統合より簡単です。セキュリティ上のリスクが少なく、ビジネス単位間の相互作用の管理が企業間ほど困難ではないからです。多くの場合、ビジネス単位は同じプロトコル、オペレーティング・システム、およびコンピューター言語を使用しています。つまり、比較的、環境が均質です。ただし、重要な点は、アプリケーションを速やかに統合するための適切なツールがあるかどうかです。以下の図では、DOM銀行には複数のビジネス単位があり、お互いに情報を共用する必要があります。数年前は、DOM銀行では、あるビジネス単位の情報のコピーを印刷し、別のビジネス単位に歩いて届けに行くことで対処してきました。それぞれのビジネス単位のシステムとアプリケーションは、そのビジネス単位に固有のものでした。今日、DOM銀行が競合銀行に後れを取らないようにするには、ビジネス単位を包含する統合アプリケーションを作成する必要があります。



## DOM 銀行



## 企業間の統合

ビジネス単位間でアプリケーションの統合が促進されている理由は、企業間にも適用されます。企業の提携または引き継ぎにはデータとプロセスの共用が必要だからです。

テクノロジーにより、企業は、相互に有利なエリアでリンクできます。例えば、自動車メーカーは、タイヤの在庫が足りないときに自動的にサプライヤーに通知されるように、タイヤ・サプライヤーとの統合プロセスをセットアップできます。企業間の統合は、経済的な必要性によって行われます。企業間のつながりを密接にすることは、遅延時間が減り、作業を実行するためのオーバーヘッドが少なくなります。これらの自動化プロセスは、人々が企業間でのトランザクションの処理に費やす時間が減り、交通費や出席する会議の時間を大幅に削減できることを意味します。管理コストも同じく削減され、通知、配送、および送り状間の送受反転時間が改善されます。

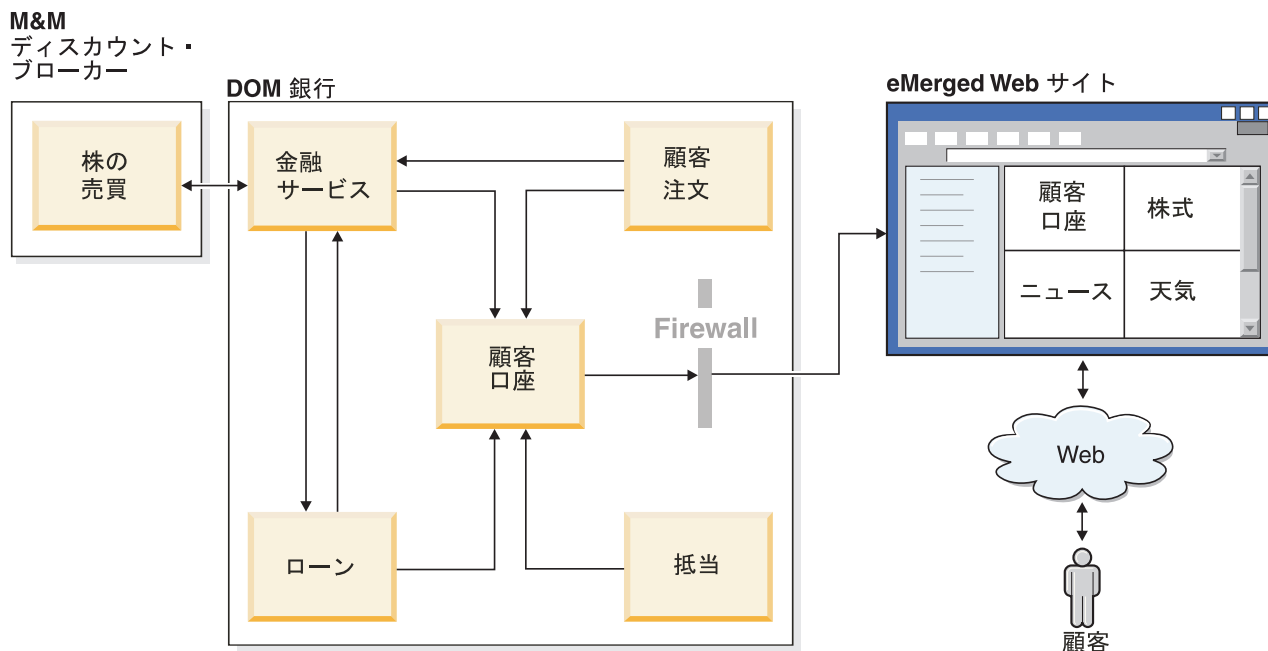
しかし、企業が異なれば、ヒストリーも異なります。アプリケーションは、異なる通信プロトコルを使用する別々のプラットフォームで、さまざまな言語でコード化されます。異なる組織を処理する場合は、セキュリティ上のリスクも大きくなります。企業間統合の利点と必要性がどのようなものであっても、適切なツールがないと、開発時間のコストが大きくなります。

## WebSphere Integration Developer

WebSphere Integration Developer は、組織が日常的に直面する統合の課題する解決策を提供するものです。これは、統合アプリケーションをビルドする人々のための完全な統合開発環境として設計されています。この環境は、統合アプリケーションの開発を単純化して促進するために、基礎的な実装からビジュアル表示された操作対象コンポーネントを分離する抽象化レイヤーを提供します。

統合アプリケーションは単純なものではありません。これらはエンタープライズ情報システム (EIS) 上のアプリケーションを呼び出し、部門または企業全体にわたるビジネス・プロセスに関与させ、さまざまな言語で記述された、各種オペレーティング・システムで実行されるアプリケーションをローカルまたはリモートで起動することができます。例えば、eMerged コーポレーションは DOM 銀行と M&M ディスカウント・ブローカーを合併して設立されました。この合併は、上記のすべてを意味しました。つまり、EIS シス

テム上のアプリケーション、ビジネス・プロセス、およびそれぞれの旧コーポレーション内のアプリケーションをコーポレーション間で共用し、新しい一連の顧客にシームレスに提供する必要がありました。それでも eMerged はこのタスクを達成し、以下の図に示すように、以前は別々であったビジネスの両方の顧客が、オンラインで各自の金融情報にアクセスできるようになっています。



eMerged は自社および顧客のための統合アプリケーションをビルドするために、 WebSphere Integration Developer のツールを使用しました。これらのツールは、EIS システム上にリモートで存在するアプリケーションを含むアプリケーションと、ビジネス・プロセスをコンポーネントとして示します。これらのコンポーネントはビジュアル・エディターによって作成され、その他の統合アプリケーション (すなわち、一連のコンポーネントから作成されたアプリケーション) に組み立てられます。ビジュアル・エディターは、コンポーネントとその実装の間に抽象化レイヤーを表示します。これらのツールを使用する開発者は、各コンポーネントの基礎実装についての詳細な知識を持たずに、統合アプリケーションを作成することができます。

これらのツールでは、トップダウンの設計方法 (1 つ以上のコンポーネントの実装が存在せず、後で追加される) およびボトムアップの設計方法 (コンポーネントが実装済みで、開発者がビジュアル・エディターでコンポーネントをドラッグ・アンド・ドロップすることによって組み立て、コンポーネントを線で結合して論理フローを作成する) の両方で統合アプリケーションをビルドできます。デバッグおよびテスト環境は、アプリケーションを実動サーバーにデプロイする前のフル・テストになります。 モニター・ポイントを設定すると、アプリケーションがどのように使用されているかをリアルタイムで表示し、パフォーマンスを最適化するために微調整することができます。

WebSphere Integration Developer のツールはサービス指向アーキテクチャーに基づいています。コンポーネントはサービスであるため、多数のコンポーネントに関係する統合アプリケーションはサービスとなります。作成されるサービスは、主要な業界標準に準拠します。ビジネス・プロセスもコンポーネントになり、これも同様に、業界標準の Business Process Execution Language (BPEL) に準拠した使いやすいビジュアル・ツールで作成されます。 WebSphere Integration Developer は、Windows と Linux プラットフォームの両方で使用できます。

WebSphere Integration Developer のツールの利点には、以下のようなものがあります。

- 容易に習得できる

- 複雑な統合状態に適用できる
- 業界標準に準拠したアプリケーションを迅速に作成できる

## 標準

WebSphere Integration Developer で作成されるアプリケーションは、サービス指向アーキテクチャーに関連する業界標準に準拠しています。

独自のコードに縛られたアプリケーションを作成しようと思う人はいません。このようなアプリケーションは数年サポートされなかったり、高いライセンス交付費用を伴う可能性があります。このため、標準ベースでの統合が WebSphere Integration Developer の基本的な特徴となっています。コネクティビティーには J2EE コネクター・アーキテクチャー標準が使用されます。データの送達を保証する必要がある大規模なアプリケーションでよく使用される非同期メッセージングには、Java Message Service (JMS) 標準が使用されます。WebSphere Integration Developer では、Simple Object Access Protocol (SOAP) に基づく Web サービスを簡単に統合できます。サービスの記述には、十分に確立された Web サービス記述言語 (WSDL) 標準が使用されます。ビジネス・プロセスの定義には、Business Process Execution Language (BPEL) 標準が採用されています。

これらの標準に基づくインターフェースおよびコンポーネントは、開放型の交換可能なアーキテクチャーを構成します。ただし、専有エレメントは除外されていません。これらのエレメントには、標準化されたインターフェースを使用してアクセスします。つまり、WebSphere Integration Developer で作成されたアプリケーションは、.NET アプリケーションなどと相互作用できます。アーキテクチャー・セクションには、サービス・コンポーネント・アーキテクチャー全般へのリンクが提供されています。これは、多くのサポートされる標準が挙げられた広範囲のリストにリンクされています。

## 統合開発者のロール

統合開発者は、WebSphere Integration Developer の主要なユーザーです。基礎実装についての詳細な知識がなくても、統合開発者はビジュアル・ツールを使用して複雑な統合アプリケーションを作成できます。

WebSphere Integration Developer は、アプリケーションおよびビジネス・プロセスをコンポーネント として表示します。コンポーネントの実装は非表示のままで、インターフェースによってコンポーネントが相互操作されます。このため、統合開発者は、コンポーネントの基礎実装についての詳細な知識を必要とせず、これらのコンポーネントを使用する統合アプリケーションを作成できます。ただし、統合開発者はおそらく統合分野についての幅広い技術知識を持っています。これは、統合開発者は EIS システム、ビジネス・プロセス、および Java またはその他の言語でコード化されたアプリケーションについて理解する必要があります。例えば、設計者はそれぞれのコンポーネントの機能を詳しくは知らずに、システムがどのように機能するかについて幅広く理解しています。設計者と同様に、統合開発者は、組織内でアプリケーション全体を設計する担当者となり、別の担当者に特定のコンポーネントの実装をコード化させることができます。

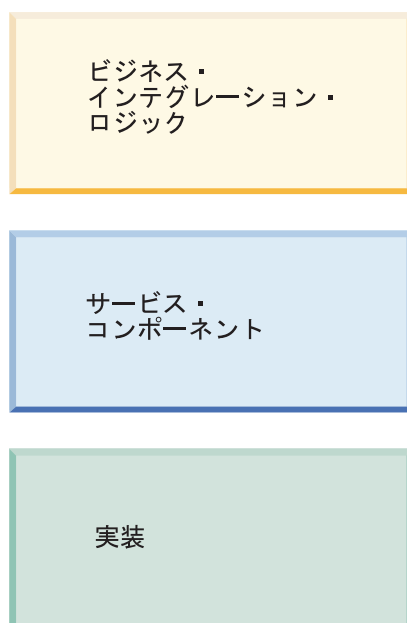


---

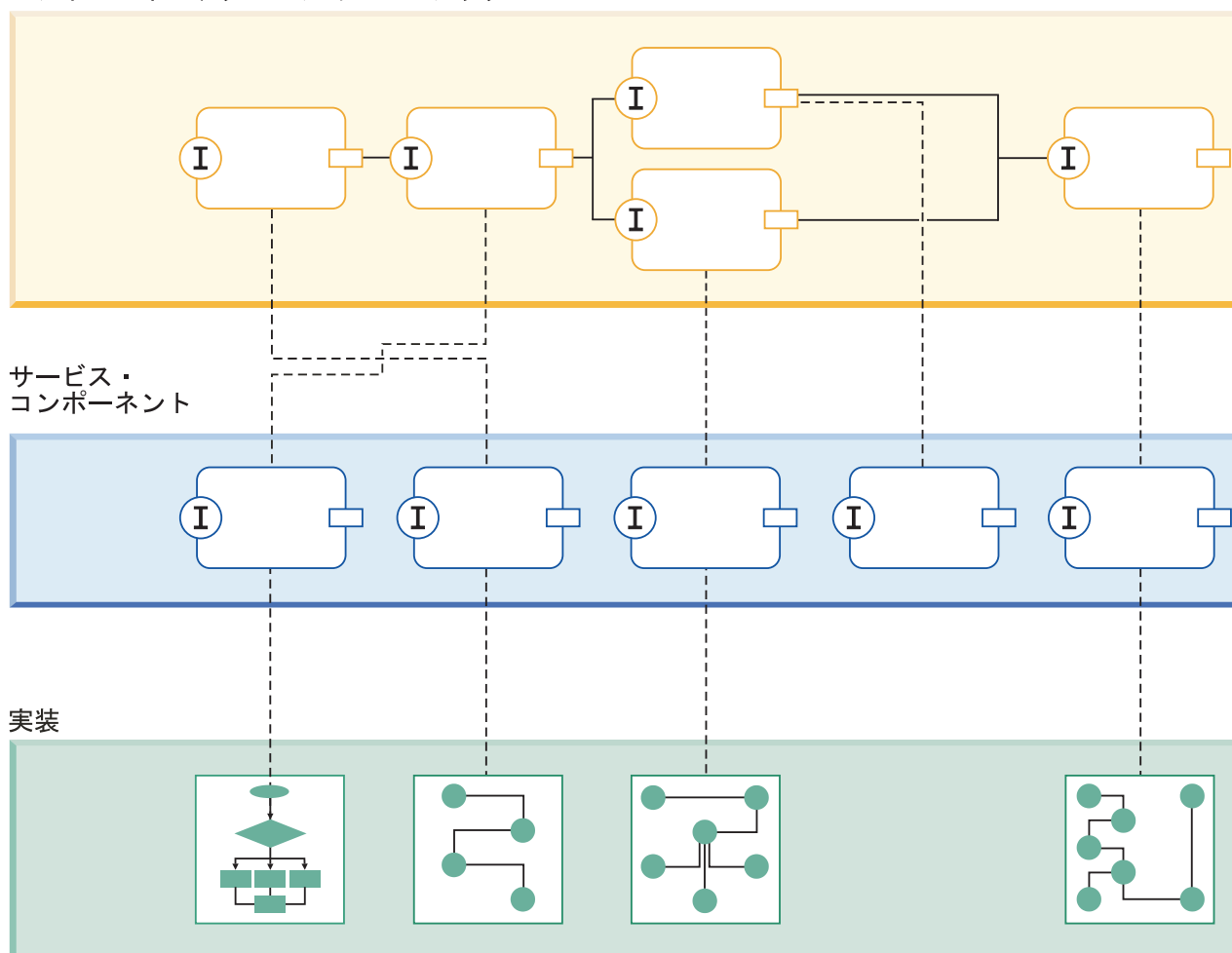
## 第 2 章 Service Component Architecture

業界標準サービス指向アーキテクチャーに基づく Service Component Architecture は、すべてのビジネス・プロセス (Web サービス、エンタープライズ情報システム (EIS) のサービス資産、ワークフロー、データベースなど) をサービス指向方式で表示します。このセクションでは、このアーキテクチャーで作成されるサービスおよびサービス・データ・オブジェクトをハイレベルで検討します。サービスおよびサービス・データ・オブジェクトはともにビジネス・ロジックを表現し、ビジネス・データを参照します。

Service Component Architecture のゴールは、ビジネス・インテグレーション・ロジックを実装から分離し、統合開発者が、実装の詳細ではなく、統合アプリケーションの組み立てに集中できるようにすることです。このゴールに到達するために、ビジネス・プロセスが必要とする個々のサービスの実装を含むサービス・コンポーネントが作成されています。その結果、アーキテクチャーの構成は次の図に示す 3 つのレイヤー (ビジネス・インテグレーション・ロジック、サービス・コンポーネント、および実装) となります。



サービス・コンポーネントには実装が含まれるため、統合開発者は、下位の実装の詳細についての知識がなくても、サービス・コンポーネントを視覚的に組み立てることができます。サービス・コンポーネントは、統合開発者または統合開発者に代わって作業する人が後で実装を追加するためのオプションも提供します。製品を見るとわかるように、視覚的に、複数のコンポーネントと一緒に組み立てられます。つまり、コンポーネント内のコードがユーザーに対して明らかにされることはありません。次の図に示すビジネス・ロジック・レベルでは、コンポーネントはその実装とは独立して組み立てられます。このため、サービス指向アーキテクチャーによって、使用するサービスの実装テクノロジーに注意を向けるのではなく、コンポーネントを使用および再使用することによってビジネス上の問題の解決に集中できます。

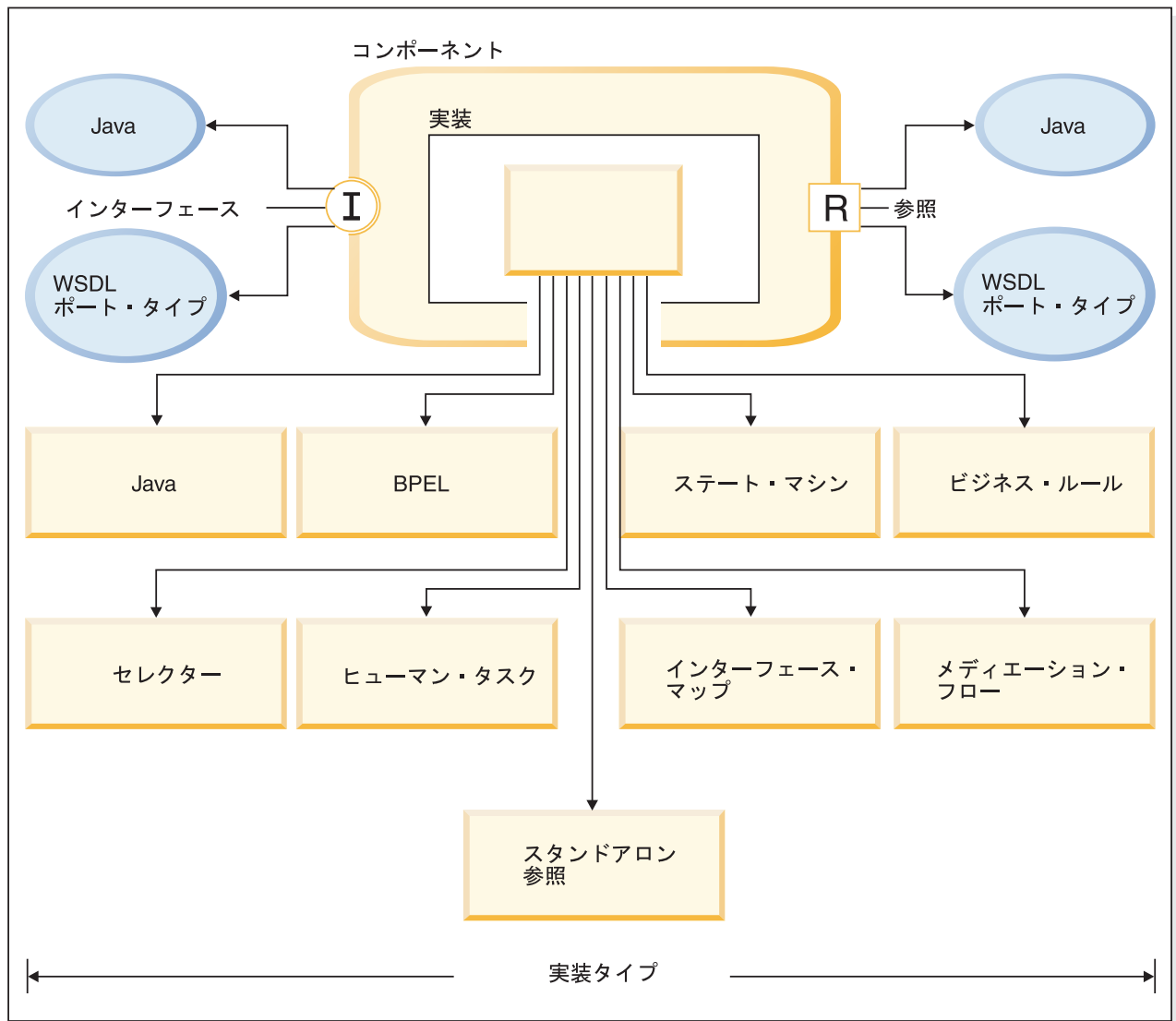


## サービス・コンポーネント

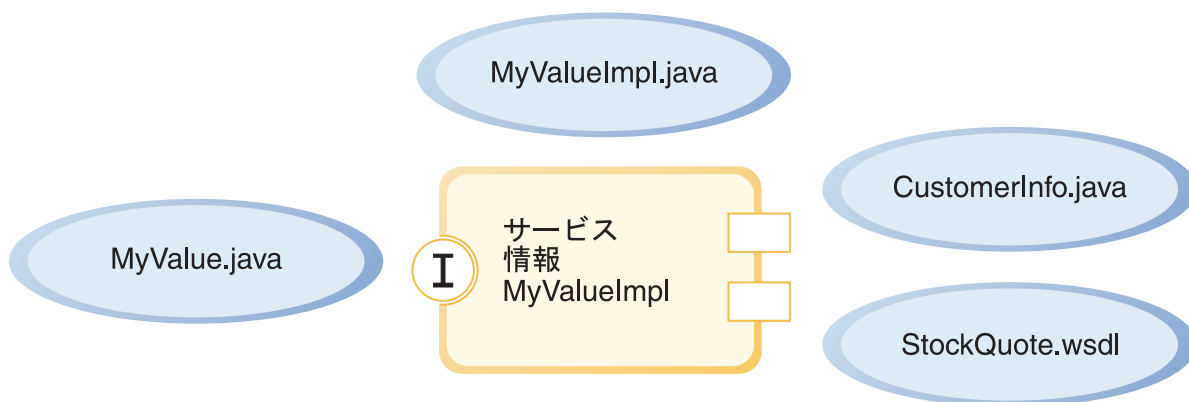
サービス・コンポーネントは、サービス実装を構成します。 サービス・コンポーネントは、標準ブロック図で表示されます。

コンポーネントは、1 個の実装 (WebSphere® Integration Developer のツールの使用中は非表示にされる)、1 個以上のインターフェース (入力、出力、およびフォールトを定義する)、および 0 個上の参照で構成されます。参照は、コンポーネントが必要とするか、利用する別のサービスまたはコンポーネントのインターフェースを識別します。インターフェースは、WSDL ポート・タイプまたは Java™ の 2 つの言語のいずれかで定義できます。インターフェースは同期および非同期対話形式をサポートします。 コンポーネントの実装は、各種の言語で可能です。

推奨インターフェース・タイプは WSDL です。チュートリアルやサンプルでは、一貫して WSDL インターフェース・タイプを使用しています。ただし、ステートレス・セッション EJB がインポートされる場合、多くは Java インターフェースがサポートされ、使用されます (14 ページの『インポートおよびエクスポート』で後述)。トップダウンで Java コンポーネントを開発する場合 (つまり、まずコンポーネントを定義し、後から Java 実装を追加する場合) は、やはり WSDL インターフェースを使用することをお勧めします。 WSDL インターフェース・ベースのコンポーネントを Java インターフェース・ベースのコンポーネントと混用することはできません。



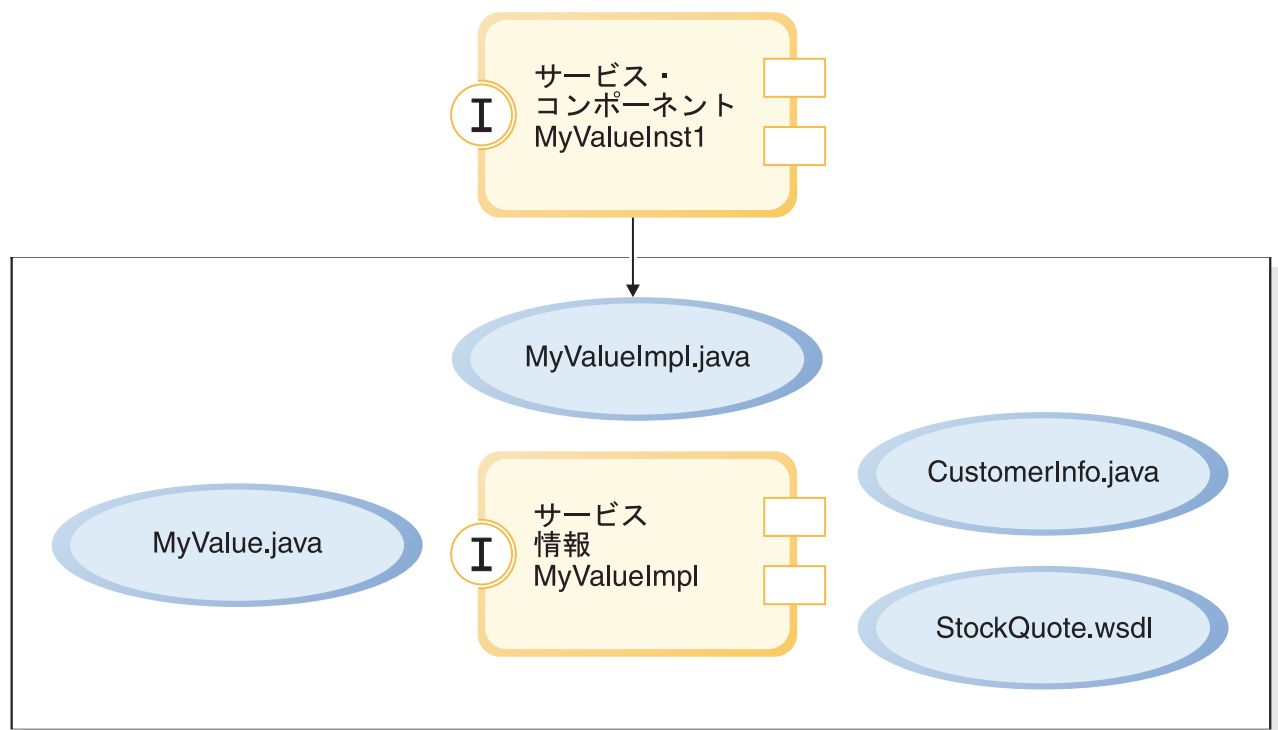
以下の図には、中央にコンポーネントがあります。コンポーネントの実装である MyValueImpl は、そのインターフェースと同様に Java です。これには、別の Java インターフェースと WSDL インターフェースの 2 つの参照があります。





以下に示すように、このコンポーネントの操作中にはコンポーネント自体だけが効果的に表示されます。別のコンポーネントからのこのコンポーネントへの参照は、インターフェースへの線によって視覚的に分かります。このコンポーネントからの参照は、その参照点から別のコンポーネントのインターフェースへの線によって視覚的に分かります。参照は、このコンポーネントが利用するサービスを表します。参照に名前を付け、そのインターフェースのみを指定することによって、コンポーネント実装の作成者は、実際のサービスへの参照のバインディングを後まで延期することができます。後で、統合の専門家が、参照から別のコンポーネントまたはインポートのインターフェースにワイヤリングすることによってバインディングを行います。このような疎結合は、据え置きバインディングおよび実装の再利用を可能にし、WebSphere Integration Developer の Service Component Architecture を使用する主な理由の 1 つになります。

また、コンポーネントにプロパティと修飾子がある場合もあります。修飾子は、ランタイムのインターフェースおよび参照に関するサービスの品質 (QoS) ディレクティブです。



## サービス・データ・オブジェクト

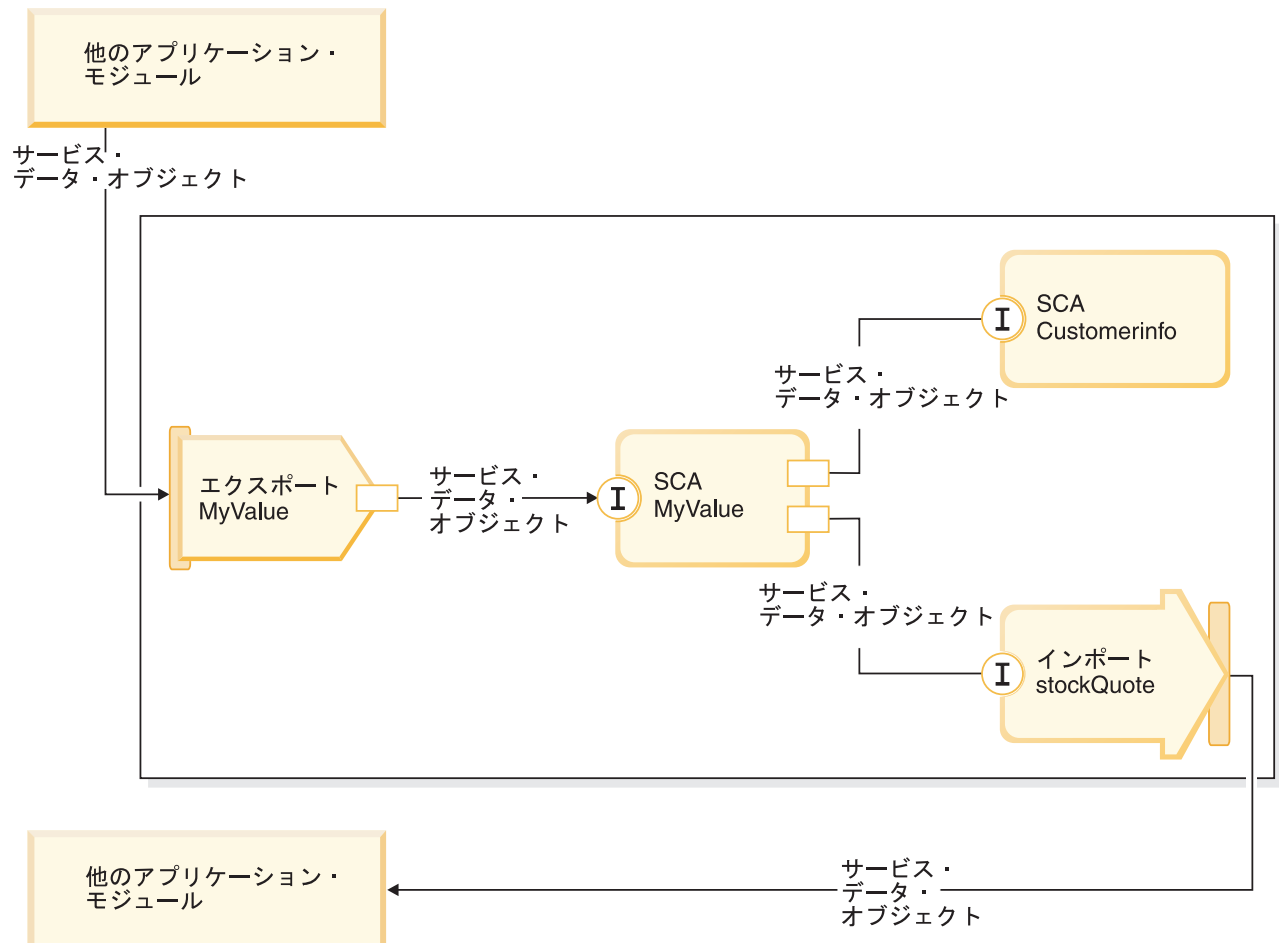
サービス・データ・オブジェクトは、Service Component Architecture を補完します。Service Component Architecture は、コンポーネントとしてのサービス、およびサービス間の接続を定義します。サービス・データ・オブジェクト は、コンポーネント間を流れるデータを定義します。

それぞれのコンポーネントは、情報を入力および出力として受け渡します。サービスが呼び出されると、データ・オブジェクトが、文書リテラル・エンコードを使用した XML 文書として (WSDL ポート・タイプを使用している場合)、または Java オブジェクトとして (Java インターフェースを使用している場合) 渡されます。データ・オブジェクトは、Service Component Architecture のサービスにおけるデータおよびメタデータの優先書式です。コンポーネントと同様に、サービス・データ・オブジェクトはデータ・オブジェクトをその実装から分離します。例えば、コンポーネントは購入注文と相互作用し、購入注文自体は JDBC、EJB などを使用してデータを更新することができます。サービス・データ・オブジェクトにより、統合開発者はビジネス成果物の操作に集中することができます。実際には、統合開発者がサービス・デー



タ・オブジェクトを意識することはありません。サービス・データ・オブジェクトは、service data objects Java Specification Request (JSR) によって定義されます。

次の図では、サービス・データ・オブジェクトが外部サービスからエクスポートに、エクスポートからコンポーネントに、コンポーネントからコンポーネントに、コンポーネントからインポートに、そしてインポートからサービスに渡されます。インポートとエクスポートについては、後続の『インポートおよびエクスポート』セクションで説明します。



## サービス修飾子

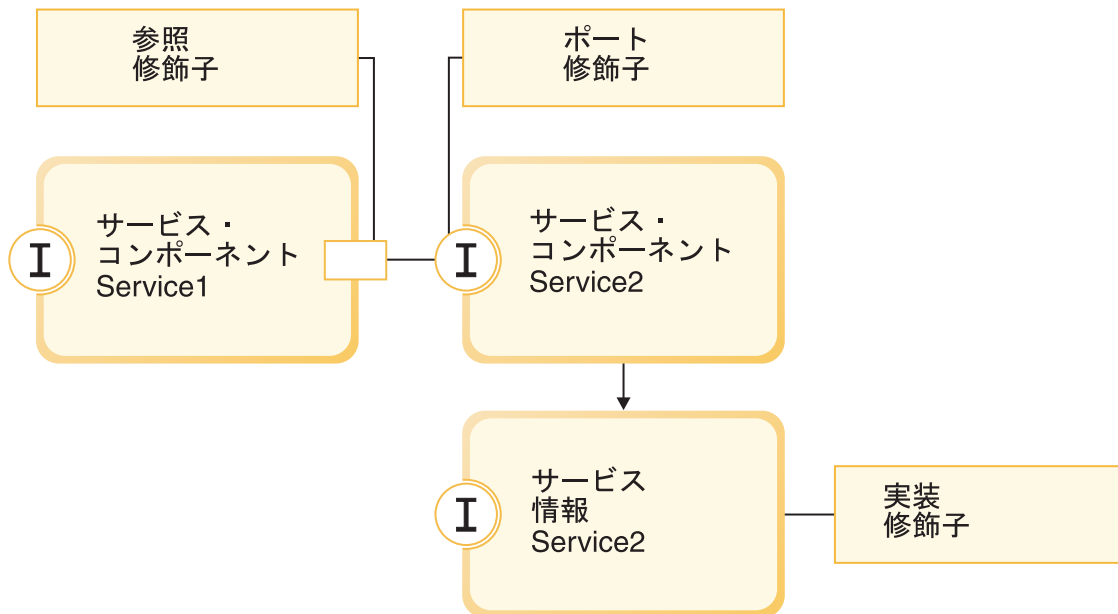
アプリケーションは、サービス修飾子を指定してサービスの品質 (QoS) の必要性をランタイム環境に伝えます。これらは、サービス・クライアントとターゲット・サービス間の相互作用を制御します。

修飾子は、サービス・コンポーネント参照、インターフェース、および実装に指定できます。QoS 値の宣言は実装の外部であるため、同じ実装の複数のインスタンスが異なるコンテキストで使用されている場合、実装を変更せずにこれらの値を変更したり、または値がそれぞれ異なるように設定したりすることができます。

修飾子のカテゴリは以下のとおりです。

- ・ トランザクション - トランザクション・タイプに対するルール
- ・ アクティビティ・セッション - アクティブ・セッションの結合に対するルール

- セキュリティー - アクセス権に対するルール
- 非同期信頼性 - 非同期メッセージ送達に対するルール



## モジュール

モジュールは、エンタープライズ・アーカイブ (EAR) ファイルと一緒にパッケージされる成果物を決定するデプロイメントのユニットです。モジュールを持つコンポーネントがパフォーマンスを考慮して連結されます。このコンポーネントは、参照によってデータを渡すことができます。モジュールはスコープのメカニズムとみなすことができます。つまり、モジュールによって、成果物の組織境界が設定されます。

モジュールは、サービス・コンポーネント、インポート、およびエクスポートの複合体です。サービス・コンポーネント、インポート、およびエクスポートは同じプロジェクトの、同じルート・フォルダー内に存在します。これには、インポートおよびエクスポートに必要なバインディングとコンポーネントをリンクするワイヤリングも含まれます。モジュールには、コンポーネント、インポート、およびエクスポートによって参照される実装とインターフェースも含まれていることがあります。また、こうした実装とインターフェースは、ライブラリー・プロジェクトなど、他のプロジェクトに入っている場合もあります。

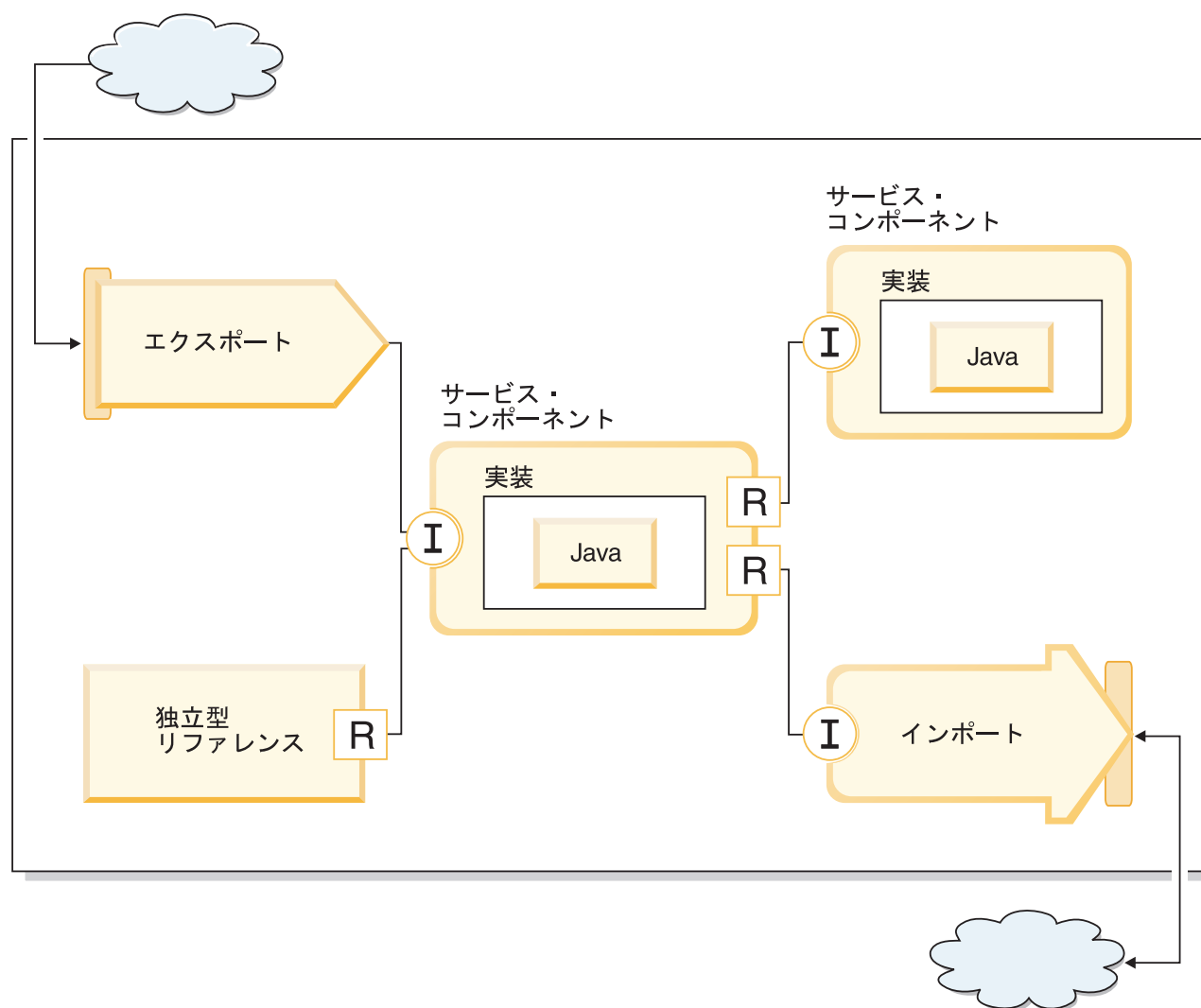
モジュールには 2 つのタイプがあります。1 つはモジュールと呼ばれるモジュールです (ビジネス・インテグレーション・モジュールと呼ばれることもあります)。この中には多くのコンポーネント・タイプから選択したものが含まれており、このモジュールはビジネス・プロセスをサポートするためによく使用されます。もう 1 つはメディエーション・モジュールと呼ばれるモジュールです。この中には、1 個のコンポーネント、1 個のメディエーション・フロー・コンポーネント、およびメディエーション・フロー・コンポーネントを増補する 0 個以上の Java コンポーネントが含まれます。

なぜ、2 つのモジュール・タイプがあるのでしょうか? 最初のモジュール・タイプは、基本的に、ビジネス・プロセス用に設計されています。メディエーション・モジュールは、既存の外部サービスへのゲートウェイのようなものです。これは、エンタープライズ・サービス・バス・アーキテクチャーでよく使用されます。これらの外部サービスまたはエクスポートは、インポートまたはサービス・プロバイダーによってメディエーション・モジュール内でアクセスされます。クライアント・サービス要求元とサービス・プロバイダーの結合をメディエーション・フローによって分離することで、アプリケーションの柔軟性と回復力が増

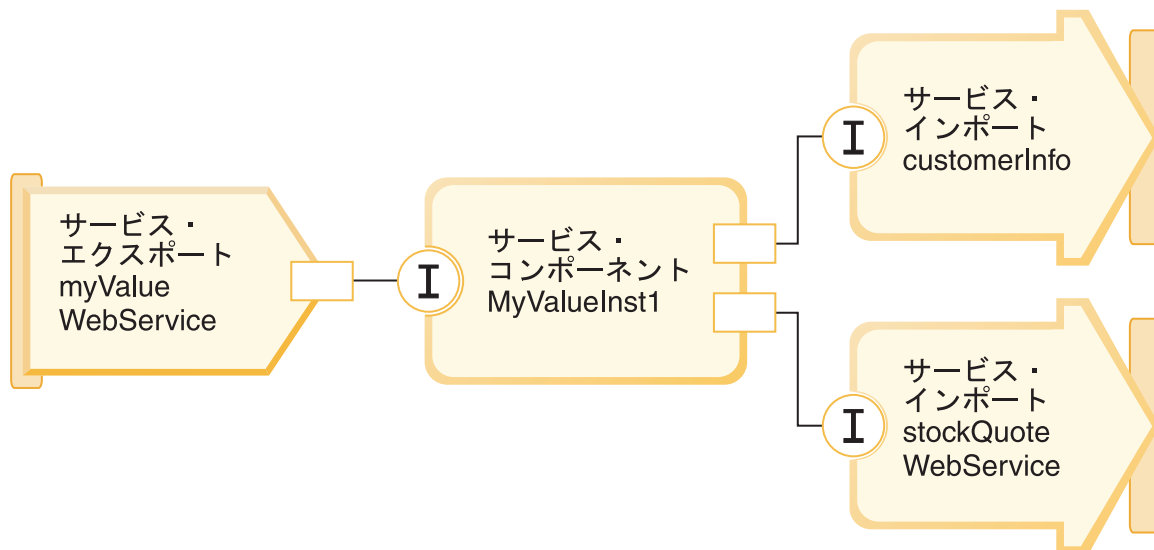
し、サービス指向アーキテクチャーというゴールに到達できます。例えば、メディエーション・フローで着信メッセージを記録したり、実行時に決定された特定のサービスにメッセージを経路指定したり、別のサービスへの受け渡しに適したものになるようにデータを変換したりすることができます。これらの機能は、要求元またはプロバイダー・サービスを変更せずに長期的に追加または変更できます。

モジュールにより、サービス・アプリケーションがテストされて、 WebSphere Process Server にデプロイされます。メディエーション・モジュールにより、サービス・アプリケーションがテストされて、 WebSphere Process Server または WebSphere Enterprise Service Bus サーバーのいずれかにデプロイされます。どちらのタイプのモジュールでも、インポートとエクスポートがサポートされます。

多くの場合、実装、インターフェース、ビジネス・オブジェクト、ビジネス・オブジェクト・マップ、ロール、関係、およびその他の成果物をモジュールで共有する必要があります。ライブラリー は、これらの共有リソースを保管するために使用されるプロジェクトです。



次の図では、モジュールに 1 つのエクスポート、2 つのインポート、およびこれらを使用する 1 つのサービス・コンポーネントが含まれています。インターフェースと参照をリンクするワイヤリングが示されています。



モジュールおよびメディエーション・モジュールの成果物には、以下が含まれます。

- モジュール定義 - モジュールを定義します。
- サービス・コンポーネント - モジュールに含まれるサービスの定義です。モジュール内のサービス・コンポーネント名は固有です。ただし、サービス・コンポーネントには任意の表示名を付けることができます。一般に、表示名はユーザーにとってよりわかりやすい名前にします。
- インポート - インポートの定義。これは、このモジュールにとって外部のサービスの呼び出しです。インポートにはバインディングが含まれます。これに関しては、『インポートおよびエクスポート』セクションで説明します。
- エクスポート - エクスポートの定義。これは、このモジュールにとって外部の呼び出し元にコンポーネントを公開するために使用されます。エクスポートにはバインディングが含まれます。これに関しては、『インポートおよびエクスポート』セクションで説明します。
- 参照 - モジュール内のあるコンポーネントから別のコンポーネントの参照です。
- スタンドアロン参照 - Service Component Architecture コンポーネント (JavaServer Pages など) として定義されていないアプリケーションを参照します。スタンドアロン参照を使用すると、これらのアプリケーションが Service Component Architecture コンポーネントと相互作用できます。モジュールごとに 1 つのみのスタンドアロン参照成果物が存在できます。
- その他の成果物 - これらの成果物には WSDL ファイル、Java クラス、XSD ファイル、BPEL プロセスなどがあります。

---

## インポートおよびエクスポート

インポートおよびエクスポートは、モジュールの外部インターフェースまたはアクセス・ポイントを定義します。インポートはモジュール外部のサービスを識別し、モジュール内からそれらのサービスを呼び出せるようにします。エクスポートでは、コンポーネントがサービスを外部クライアントに提供できるようにします。インポートまたはエクスポートには、バインディング情報が必要です。いくつかのバインディングが使用可能です。ご使用のアプリケーションに適切なタイプはどれなのかについて説明します。

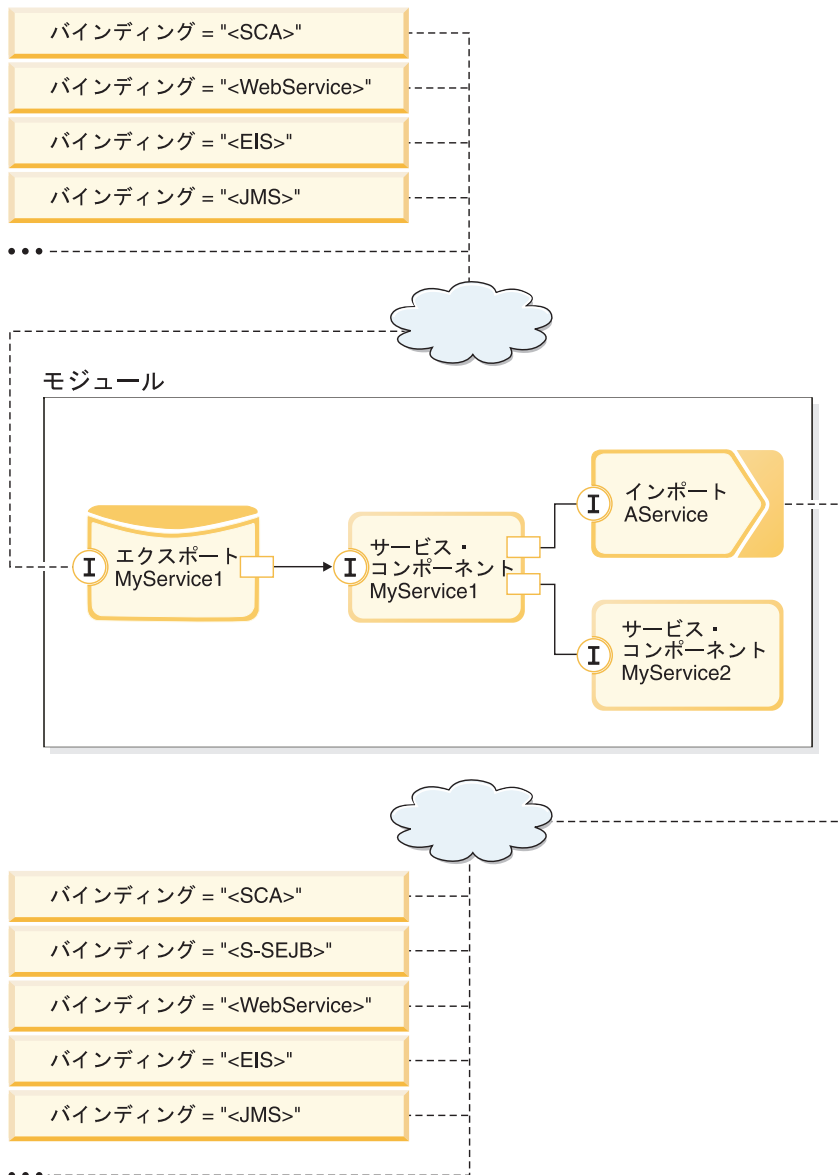
## サービス・インポートおよびエクスポートのバインディング・タイプ

インポートおよびエクスポートには、バインディング情報が必要です。この情報は、モジュールからデータを移送する方法を指定します。インポート・バインディングは、外部サービスをインポート・コンポーネ

ントにバインドする特定の方法を記述します。 エクスポート・バインディング は、モジュールのサービスをクライアントが使用できるようにする特定の方法を記述します。

SCA またはデフォルト・バインディングでは、他のモジュール内のその他のサービスと通信できます。 SCA バインディングによるインポートでは、別のモジュール内のサービスにアクセスできます。 SCA バインディングによるエクスポートでは、サービスを他のモジュールに提供できます。 Web サービス・インポート・バインディングでは、外部 Web サービスをインポートにバインドできます。 Web サービス・エクスポート・バインディングでは、サービスを Web サービスとして外部クライアントに提供できます。 エンタープライズ・サービス・ディスカバリー・ウィザードは、EIS システムのサービスを表すインポートおよびエクスポートを作成します。作成されるバインディングは EIS タイプまたは Java Message Service (JMS) タイプです。 EIS バインディングは、EIS システムのサービスとの同期通信を提供します。 JMS バインディングは通常、大規模 EIS システム (メッセージ・キューを介した非同期通信が、信頼性のための重要なポイントになります) との相互作用で使用されます。インポートも (エクスポートではありませんが)、ステートレス・セッション EJB バインディングを持つことがあります。

エクスポートにアクセスするインポートを作成する場合、アセンブリー・エディターがサポートされるバインディングをリストし、バインディングの作成を単純化します。アセンブリー・エディターの「プロパティ」ビューには、インポートまたはエクスポートのバインディング情報が表示されます。



## 適切なバインディングの選択

このセクションでは、アプリケーションのニーズに応じて、どのような場合に特定のバインディングがより適切になるかについて説明します。

WebSphere Integration Developer で使用可能なバインディングには、選択の幅があります。以下に挙げる一覧は、アプリケーションのニーズに応じて、どのような場合に、あるバインディングが別のタイプのバインディングよりも適切となり得るかを識別するのに役立ちます。

以下の要因が当てはまる場合は、SCA バインディングを検討してください。

- すべてのサービスが WebSphere Integration Developer モジュールの中に含まれている場合、つまり、外部モジュールがない場合
- パフォーマンスを重視している場合
- モジュールが密結合されている場合

以下の要因が当てはまる場合は、*Web* サービス・バインディングを検討してください。

- インターネットを介して外部のサービスにアクセスしたり、インターネットによりサービスを提供する必要がある場合
- サービスが疎結合されている場合
- アクセスする外部のサービスまたは提供するサービスのプロトコルが SOAP/HTTP または JMS/HTTP である場合

以下の要因が当てはまる場合は、*EIS* バインディングを検討してください。

- リソース・アダプターを使用して *EIS* システム上のサービスにアクセスする必要がある場合
- 信頼性よりもパフォーマンスを重視している場合、つまり、非同期データ伝送よりも同期データ伝送が好まれる場合

以下の要因が当てはまる場合は、*JMS* バインディングを検討してください。

- メッセージング・システムにアクセスする必要がある場合
- サービスが疎結合されている場合
- パフォーマンスよりも信頼性を重視している場合、つまり、同期データ伝送よりも非同期データ伝送が好まれる場合

以下の要因が当てはまる場合は、ステートレス・セッション *EJB* バインディングを検討してください。

- バインディングが、それ自体が *EJB* であるインポートされたサービス用である場合
- インポートされたサービスが疎結合されている場合
- *EJB* の状態が重要でない場合

---

## サービス実装タイプ

サービス実装タイプとは、サービス・コンポーネントの実装のことです。

このセクションでは、サービスの標準実装について説明します。これらの実装は、アセンブリー・エディターのサービス内または BPEL プロセス内で使用されます。

## Java オブジェクト

Java でのコンポーネントの実装は、Java オブジェクトと呼ばれます。

一般的な実装の 1 つとして、Java で作成されたコンポーネントがあります。この実装は、「従来の Java object (plain old Java object)」または *POJO* というニックネームで呼ばれることがあります。この実装は Java インターフェースを持つこともできますが、通常は *WSDL* インターフェース・タイプを持ちます。複数のインターフェースが指定されている場合には、*WSDL* インターフェースを Java インターフェースと混用することはできません。ただし、一連の *WSDL* インターフェースで作成されたアプリケーションを一連の Java インターフェースを持つアプリケーションと「結合」することは可能です。その方法は、「ようこそ」ビューのサンプル・ギャラリーにリストされたサンプルで示しています。

Java オブジェクトを操作している間、コードはユーザーに対して非表示のまま、エディターのコンテキスト内に維持されます。

Java オブジェクトはメディエーション・モジュールで 사용할 ことができます。これは、WebSphere Process Server または WebSphere Enterprise Service Bus サーバーのいずれかにデプロイできます。

## BPEL プロセス

*BPEL* プロセス・コンポーネントは、ビジネス・プロセスを実装します。

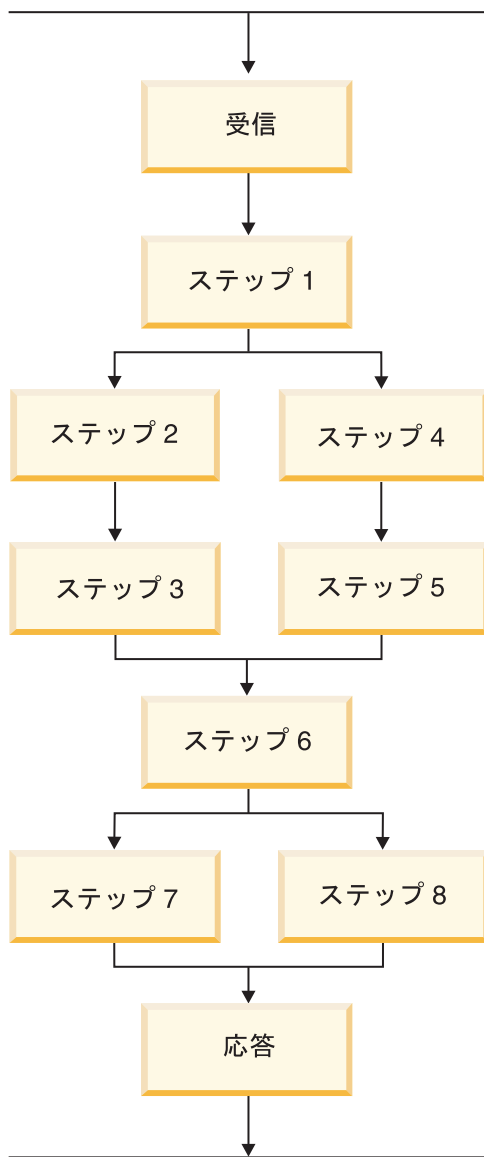
この実装言語は、業界標準の Business Process Execution Language for Web Services (BPEL4WS) およびその IBM 拡張機能です。BPEL プロセスでは、より単純なサービスを使用することによって、長期間実行される可能性のあるステートフル・サービスを実装します。プロセス・エディターで作成された BPEL プロセスは、以下を実行できます。

- コントロール・フロー・グラフを使用した、その他のサービスのオーケストレーションの記述
- 変数を使用した、プロセス状態の維持
- フォールト処理による高度エラー処理
- 非同期イベントのサポート
- インスタンスを識別する要求内のビジネス・データ (カスタマー ID など) にマークを付けるための関連セットを使用した、インバウンド要求と特定プロセスの正しいインスタンスとの関連付け
- 高度補正サポートによる、拡張トランザクションの提供

これらの標準 BPEL 項目の他に、WebSphere Integration Developer は BPEL を拡張して、ヒューマン・タスク・サポートによってプロセスに人を組み込むことができます。例えば、この拡張で、人がローンを承認するという要件をプロセスに追加することができます。

プロセス・エディターは、ユーザーがビジネス・プロセスを素早く簡単にビルドできるように、BPEL 構成のビジュアル表示を使用します。



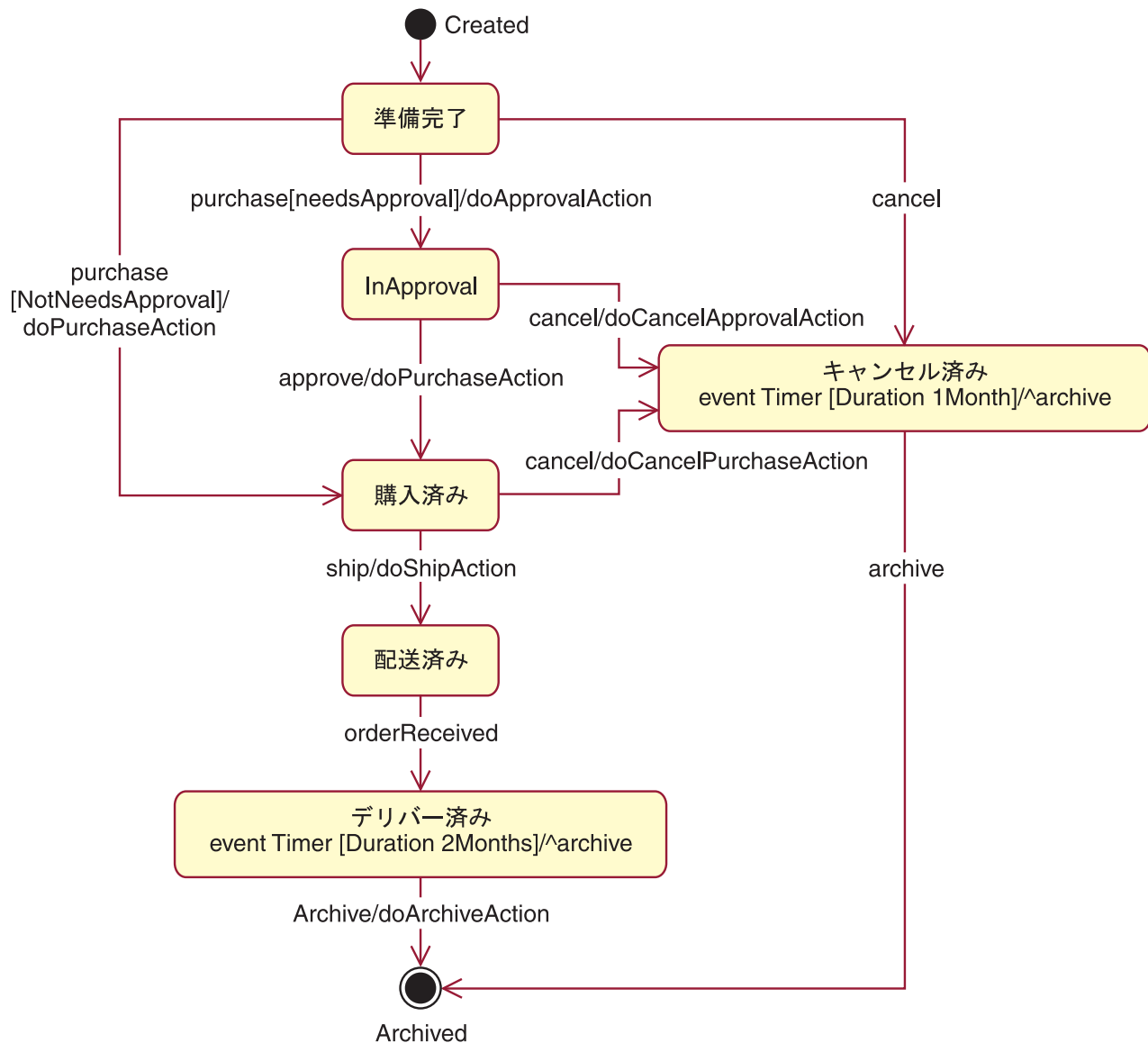


BPEL プロセスをメディエーション・モジュールで使用することはできません。これは、WebSphere Process Server にのみデプロイ可能です。

## ステート・マシン

ビジネス・プロセスを作成するための別の方法として、ステート・マシンがあります。ステート・マシンは、制御のフローではなく状態の変更に関連するプロセスに適しています。状態によって、成果物がその時点で何を行うことができるのかが定義されます。ステート・マシン とは、この状態セットの実装のことです。

ステート・マシンは、プロセスで相互に関連する状態のセットを示す一般的な方法です。よく知られているステート・マシンとして、ジュースの自動販売機があります。自動販売機に硬貨を入れると、目的のジュースとともに正確なお釣りが出てきます。これはステート・マシンが、投入された硬貨から、返却しなければならない硬貨を機械的に分析するためです。次の図に、ステート・マシン・エディターで作成された標準的なステート・マシンを示します。ステート・マシンで、品物が購入され、顧客に出荷されます。



ステート・マシンをメディエーション・モジュールで使用することはできません。これは、WebSphere Process Server にのみデプロイ可能です。

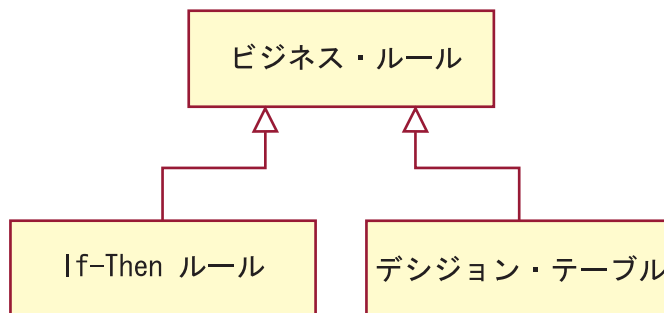
## ビジネス・ルール

ビジネス・ルールは、ビジネス・プロセスとステート・マシンを補完します。例えば、変数に関する条件がある場合、ビジネス・ルールによって、実行時に変数の値を変更できます。ビジネス・ルールはビジュアル・プログラミング言語によって作成され、コンテキストに基づいた決定を行います。決定には、単純なものもあれば、複雑なものもあります。ビジネス・ルールはプロシーチャー型ではないため、ルールはアプリケーションとは無関係に変更できます。

ビジネス・ルールは、コンテキストに基づいてプロセスの結果を決定します。ビジネス・ルールは毎日のビジネスの局面で使用され、具体的な一連の事情に基づいて決定を行います。この決定は、すべての事情をカバーするために多くのルールを必要とする場合があります。ビジネス・プロセス内のビジネス・ルールを使用して、アプリケーションはビジネス条件の変更に素早く対応することができます。例えば、ある保険

会社は、申込者に自動車保険を承認するためのビジネス・ルールとして、申込者が 25 歳を超える男性で、自動車のカテゴリーがスポーツ・カーで、過去 5 年間弊社の保険に入っていれば、毎月 100 ドルの料金で保険を承認する と定めることができます。

WebSphere Integration Developer は、ビジネス・ルールを作成するための多数のアプローチを提供します。If-Then ルールまたはデシジョン・テーブルを作成できます。これらのすべてがプロセスの結果を決定します。これらのルールはプロセス自体から独立していることに注意してください。つまり、プロセスを再実行せずに、いつでもルールを変更できます。例えば、ビジネスを行う場所によっては、日付が 12 月 26 日から 1 月 1 日の間である場合、20% 割引の歳末セールを行う というルールを設定できます。しかも、売り上げの不振が続いている場合は、いつでも割引率を 40% に変更できます。

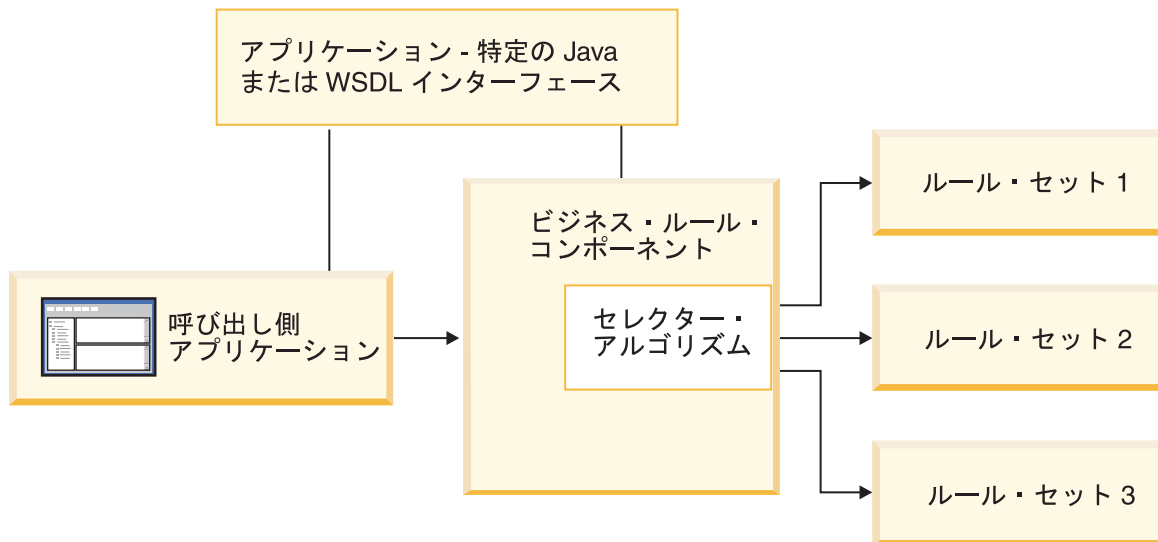


ビジネス・ルールをメディエーション・モジュールで使用することはできません。これは、WebSphere Process Server にのみデプロイ可能です。

## セレクター

統合アプリケーションには、相互作用のための多くの方法が備わっています。セレクター は、操作を、クライアント・アプリケーションから、実装で使用可能ないくつかのコンポーネントのいずれかに経路指定するために使用されます。

コンポーネントへの経路指定は日付に基づきます。日付に基づいた経路の例として、新学期が始まる 2 週間前に、学校関連の商品について新学期特別価格を提供する などがあります。ビジネスにおいては、日付に基づいたそのような経路が数多くある場合があります。セレクターは、日付に基づいて実行時に経路を選択するための判断を行います。例えば、新学期が始まる直前であれば、前述の新学期特別価格の提供を選択します。また、学校の終業式の季節であれば、子どものための夏休み準備特別価格を提供できます。

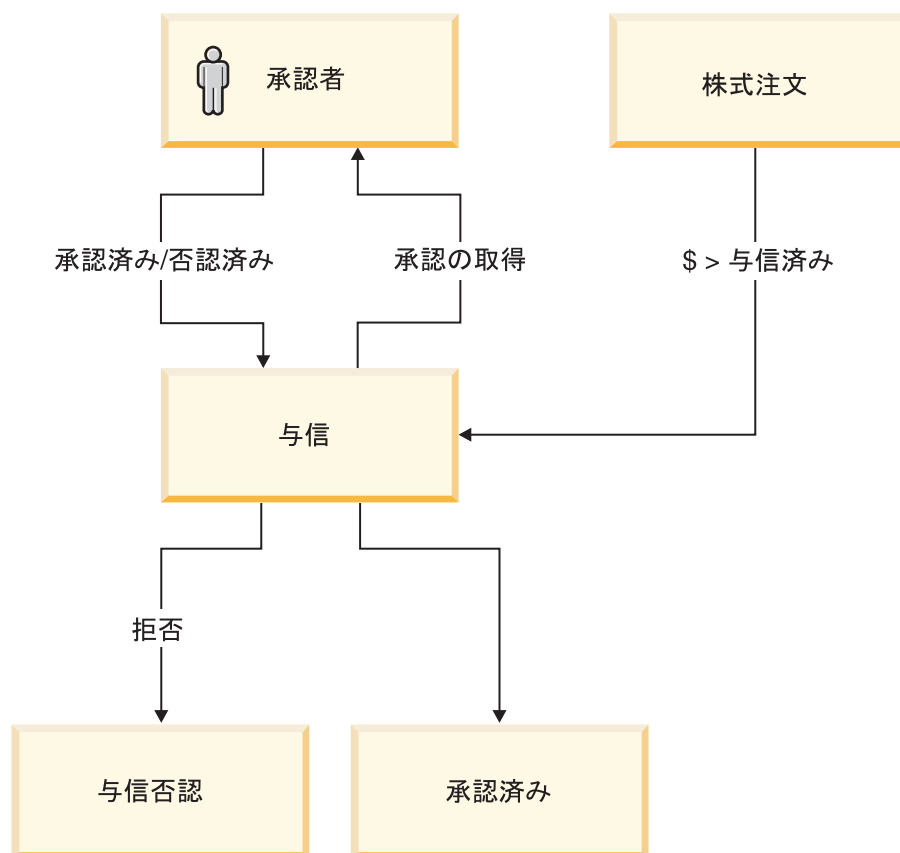


セレクトターをメディエーション・モジュールで使用することはできません。これは、WebSphere Process Server にのみデプロイ可能です。

## ヒューマン・タスク

ヒューマン・タスク・コンポーネントは、人が行うタスクを実装します。これは、ビジネス・プロセスでの人の関与を意味します。

ビジネス・プロセスに人の介入が必要となることもあります。例えば、顧客が自分のクレジット限度額を超える品物を購入したいとします。ヒューマン・タスクを使用して、人が介入して、顧客が購入できないようにしているビジネス・ルールをオーバーライドできます。ヒューマン・タスクには複数の属性があります。タスクの所有者の設定、および指定された人が不在の場合に備えたエスカレーション・プロセスの提供などです。ヒューマン・タスク・コンポーネントでは、多くのプロセスがレビュー、リサーチ、承認などのタスクで人の介入を必要とするという現実が認識されます。



ヒューマン・タスクをメディエーション・モジュールで使用することはできません。これは、WebSphere Process Server にのみデプロイ可能です。

## インターフェース・マップ

インターフェース・マップ は、相互作用するコンポーネントのインターフェース間の違いを解決します。

互いに相互作用する必要のあるコンポーネントのインターフェースに違いがあることはよくあります。WebSphere Integration Developer で、さまざまなアプリケーション用に作成したコンポーネントを組み立てることがよくありますが、これが原因でこうした違いが発生します。似たようなコンポーネントを再度コード化せずに済むため、新しいアプリケーションを作成するためにコンポーネントを再利用することは、WebSphere Integration Developer の長所の 1 つです。ただし、通常、いくつかの調整が必要になります。

例えば、2 つのコンポーネントが、基本的に同じアクションを実行するが、`getCredit` と `getCreditRating` などのように異なる名前を持つメソッドを持つことがあります。また、これらのメソッドが異なる操作名を持ち、操作がさまざまなパラメーター・タイプを持つこともあります。インターフェース・マップはこれらのメソッドの操作およびパラメーターをマップし、違いが解決され、2 つのコンポーネントが相互作用できるようにします。インターフェース・マップは、2 つのコンポーネントのインターフェース間にわたるブリッジのようなものであり、違いにかかわらず、2 つのコンポーネントを一緒にワイヤリングできるようにします。

インターフェース・マップは、それを使用するコンポーネントから独立して存在します。つまり、コンポーネント自体を変更する必要はありません。

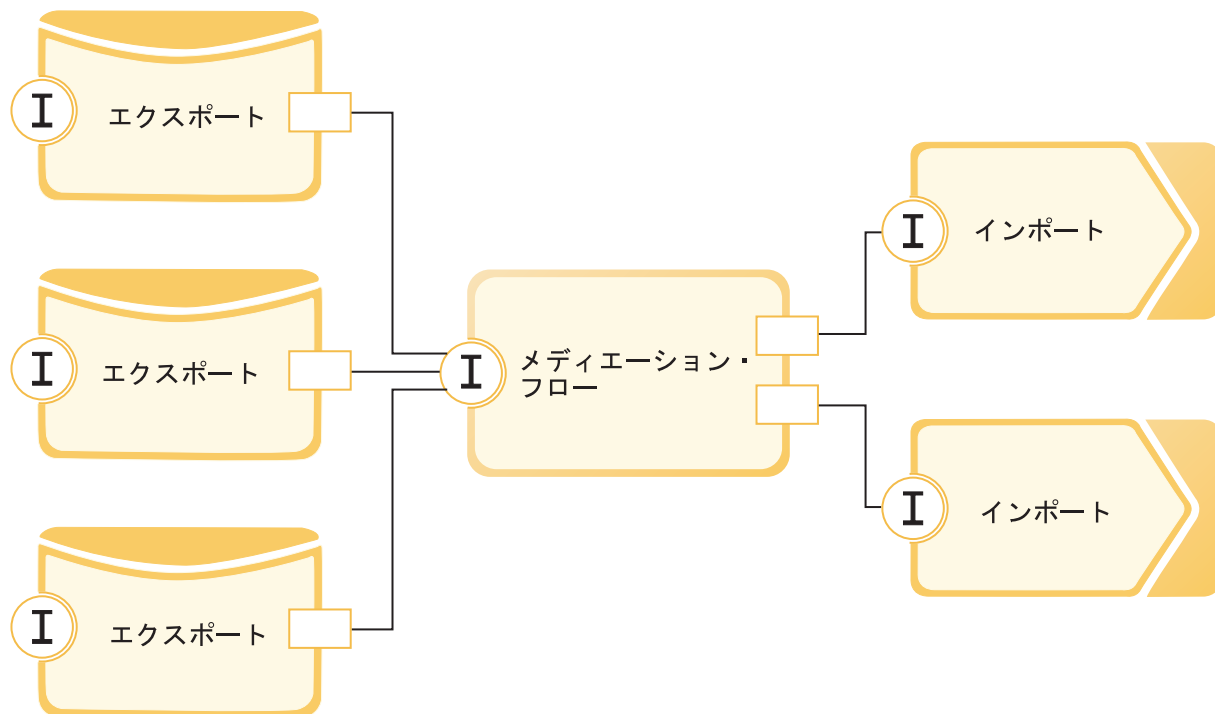
インターフェース・マップをメディエーション・モジュールで使用することはできません。これは、WebSphere Process Server にのみデプロイ可能です。

## メディエーション・フロー

メディエーションとは、サービス間を動的に仲介または介入する方法のことです。メディエーション・フローはメディエーションを実装します。

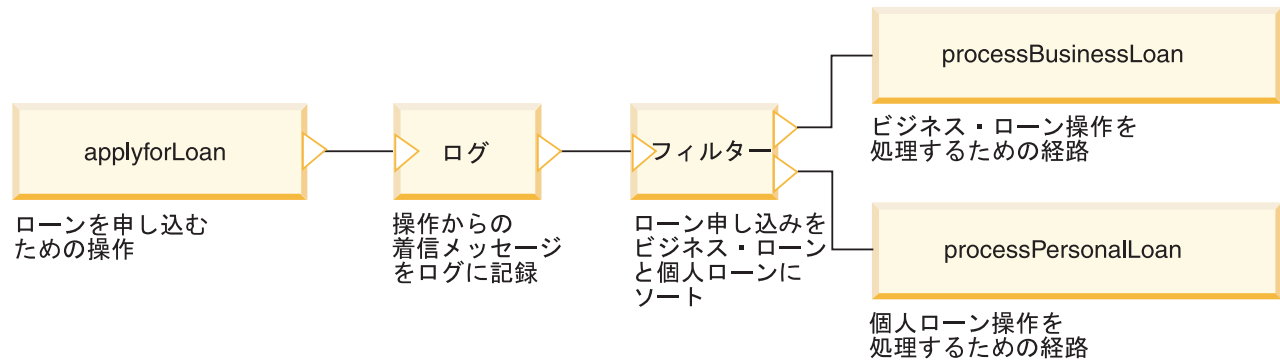
メディエーションにはいくつかの役立つ機能があります。例えば、ある 1 つのサービスからのデータを、後続のサービスが受け入れ可能なフォーマットに変換する必要がある場合に、メディエーションを使用できます。ロギングでは、あるサービスからのメッセージが次のサービスに送信される前に、そのメッセージをログに記録することができます。ルーティングでは、あるサービスからのデータを、メディエーション・フローによって決定された適切なサービスに経路指定することができます。メディエーションは、その接続先であるサービスから独立して動作します。アセンブリー・エディター内のメディエーションは、エクスポートとインポート間のメディエーション・フロー・コンポーネントとして示されます。

以下の図では、3 つのサービス要求元またはエクスポートが、その出力データをメディエーション・フロー・コンポーネントのインターフェースに送信します。すると、メディエーション・フロー・コンポーネントは、該当するデータを 2 つのサービス・プロバイダーまたはインポートに経路指定します。



メディエーション・フローは、メディエーション・フロー・エディターで作成されるフローのような構成です。アセンブリー・エディターでメディエーション・フロー・コンポーネントを選択すると、メディエーション・フロー・エディターが起動します。メディエーション・フロー・エディターでは、あるサービス、サービス要求元、またはエクスポートからの操作が、メディエーション・フロー・エディターで提供される機能と一緒に、別のサービス、サービス・プロバイダー、またはインポートの操作にマップされます。これらの機能はメディエーション・プリミティブと呼ばれ、下記の図に示されるようにメディエーション・フロー内でワイヤリングされます。メディエーション・プリミティブは IBM から提供されます。また、独自のカスタム・プリミティブを作成することもできます。メディエーション・プリミティブは、メッセージの内容とメッセージのコンテキストの両方に影響します。このコンテキストは、SOAP や JMS ヘッダーなどのバインディング特定情報またはユーザー定義プロパティです。

次の図では、操作 `applyforLoan` は、最初に、メッセージを、そのメッセージを記録するロギング・プリミティブ `Log` に送信します。 `Log` はメッセージを `Filter` プリミティブに送信し、 `Filter` プリミティブはメッセージに応じて、 `processBusinessLoan` 操作または `processPersonalLoan` 操作にそのメッセージを経路指定します。



『モジュール』のセクションで説明したように、メディエーション・フロー・コンポーネントのためのメディエーション・モジュールがあります。この中には、1 個のメディエーション・フロー・コンポーネントと、メディエーション・フロー・コンポーネントを増補する 0 個以上の Java コンポーネントが含まれます。メディエーション・モジュールは、WebSphere Process Server または WebSphere Enterprise Service Bus サーバーのいずれかにデプロイできます。

## スタンドアロン参照

スタンドアロン参照 は、Service Component Architecture コンポーネント (JavaServer Pages またはサーブレットなど) として定義されていないアプリケーションへの参照です。スタンドアロン参照によって、これらのアプリケーションが Service Component Architecture コンポーネントと相互作用できます。

スタンドアロン参照には、インターフェースも、実装也没有 (実装はモジュールのスコープ外であるため)。モジュールには、0 または 1 つのスタンドアロン参照成果物を入れることができます。スタンドアロン参照には、既存のアプリケーションを、WebSphere Integration Developer で作成された Service Component Architecture コンポーネントと一緒に使用できるようにするという実用的な価値があります。

スタンドアロン参照をメディエーション・モジュールで使用することはできません。これは、WebSphere Process Server または WebSphere Enterprise Service Bus サーバーのいずれかにデプロイできます。

---

## 関連情報

このセクションでは、アーキテクチャーに関連したいくつかのトピックについて説明します。

以下のトピックには、この製品のアーキテクチャーに関連するいくつかの追加情報があります。

- 『.NET サービスで使用するサービスの開発』
- 26 ページの『双方向言語サポート』

## .NET サービスで使用するサービスの開発

.NET サービスで使用するサービスの開発を行う場合は、いくつかの特別の考慮事項について認識しておく必要があります。これらの考慮事項には、.NET 開発環境で開発した WSDL ファイルを WebSphere Integration Developer 開発環境に取り込むこと、および .NET サービスでできるように WebSphere Integration Developer から WSDL ファイルをエクスポートすることという、2 つの局面があります。2 つ

の開発環境の主な相違点は、.NET 環境がインライン・スキーマを使用するのに対して、 WebSphere Integration Developer はインライン・スキーマを使用しないという点です。インライン・スキーマとは、スキーマを個別のファイルとしてインポートするよう指定するのではなく、 WSDL ファイルに組み込む方法のことです。 WebSphere Integration Developer には、この両方の場合に役立つ情報が提供されています。

『インターフェースの作成』のセクションの『インライン・スキーマ』では、.NET WSDL ファイルなどのインライン・スキーマを含む WSDL ファイルをインポートする方法を示しています。

『ビジネス・サービスの組み立て』のセクションの『.NET サービスで使用するプロキシの作成』では、.NET で使用できるように、サービスを外部に公開する方法を示しています。このトピックでは、特に、.NET サービスで使用するプロキシの作成を中心に説明しています。

## 双方向言語サポート

WebSphere Integration Developer は、マルチリンガル環境で動作します。このことは、さまざまな言語で表されるデータを表示したり操作したりできることを意味します。このサポートには、書き方が双方向である言語（例えば、アラビア語やヘブライ語）がいくつか含まれています。これらの言語は右から左に書かれますが、数字およびラテン語（またはキリル文字やギリシャ語など）のテキストの部分は、このテキストに左から右に組み込まれます。

『IBM WebSphere Integration Developer の双方向スクリプト・サポートの概説』では、必要な構成、このサポートを使用する場合のいくつかの特定の技術的ポイント、および制限など、双方向言語のサポートについて説明しています。



## 第 3 章 ツールについての学習

WebSphere Integration Developer では、サービス指向アーキテクチャーを基にした統合アプリケーションを作成するために、ビジュアル・ツールを使用します。

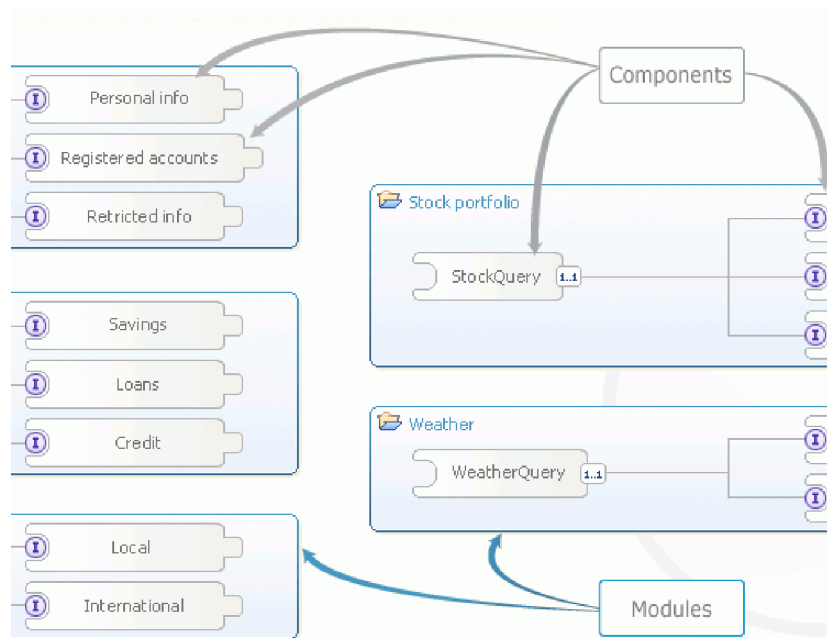
「ようこそ」ビューの「概要」セクションには、WebSphere Integration Developer のツールについての最上位レベルの知識を提供しています。ここでは、最上位レベルとは、ツールについての最も理論的なプレゼンテーションのことです。グラフィックスによってツールの機能を説明します。この製品を使用するのが初めてで、製品のツールで何ができるかを素早く理解したい場合は、「概要」セクションをお読みください。

「ようこそ」ビューには、チュートリアルセクションもあります。このセクションでは、ツールの実際の動作を見ることができます。サンプル・セクションでは、障害の危険がない方法でツールを使い始めることができます。最後に、WebSphere Integration Developer を使用してアプリケーションの開発を始めるときに、インフォメーション・センターを使用して各ツールの詳細な情報を検索することができます。この詳細な情報には、概念的な情報、実行できるタスク、および各ツールについての参照情報が含まれます。

### 「ようこそ」ビューの「概要」

「ようこそ」ビューの「概要」セクションは、最上位レベルでの製品のツールおよびフィーチャーの概説を提供します。「概要」では、重要なツールがこの製品において果たす役割がグラフィックスで示されています。

「ようこそ」ビューの「概要」セクションでは、実際に製品を使って作業を開始する前に、製品について素早く学習できます。この製品の多くの新規ユーザーは、ここから開始する必要があります。「概要」にあるアイコンを選択すると、ツールまたはトピックについてのグラフィックが表示されます。例えば、コンポーネントとモジュールの関係が次の図に示されます。製品のオーディオビジュアル・ツアーを起動することもできます。

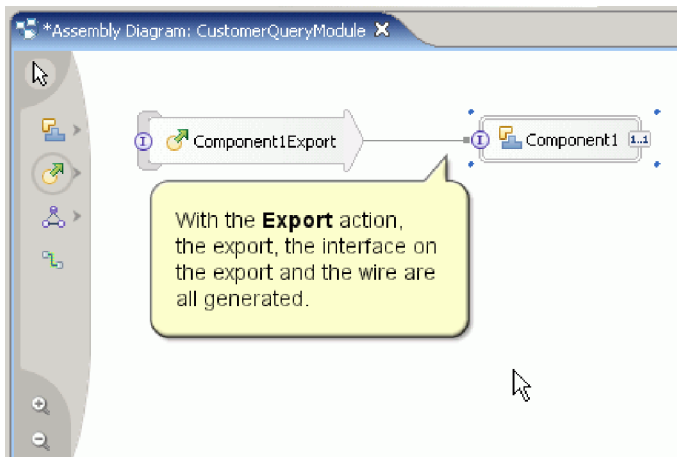


---

## 「ようこそ」ビュー内のチュートリアル

チュートリアルでは、障害の危険がない方法で、実際のツールの動作を見ることができます。チュートリアルを選択すると、チュートリアルが進行し、ユーザーが実行するとおりのタスクを見ることができます。

製品を概念的なレベルで理解した後は、チュートリアルを起動してツールを試してみることができます。それぞれのチュートリアルは、特定のツールを映画のように紹介しています。チュートリアルは見るだけでなく、表示されている内容が吹き出しテキストによって説明されます。WebSphere Integration Developer チュートリアルは、「ようこそ」ビューのチュートリアル・ギャラリーにある「実演学習 (Watch and Learn)」セクションにあります。



---

## 「ようこそ」ビュー内のサンプル

サンプルでは、障害の危険がない環境で WebSphere Integration Developer を使用してビジネス・サービス・ソリューションを開発する方法を実践的に示します。

サンプルは、「ようこそ」ビューのサンプル・ギャラリーにあります。サンプルを選択すると、複数のオプションが提供されます。段階的手順に従って自分でサンプルをビルドしたり、あるいは自動的にビルドさせることができます。

サンプルは、スコープの点で異なります。テクノロジー・サンプルは、特定のタスクを実行する特定のツールに重点を置きます。アプリケーション・サンプルは、複数のツールを使用してより複雑な結果を達成します。シナリオは、非常に長いサンプルで、多数のツールを使用して大規模なアプリケーションをビルドします。各サンプルでは、サンプルをビルドするための所要時間をリストしています。

## Business state machine (simple)

This sample demonstrates how a business state machine can be used to moderate a sales order transaction. Specifically, the state machine emulates an on-line brokerage that manages the selling of a share.

The following tools are used in this application:

- Business state machine editor
- Human task editor

To import the ready-made sample, click the **Import** link below and click **Finish** in the opened wizard. See **Running instructions** to run the imported code.

or

If you want to build the sample for yourself, click **Step-by-step instructions**.

### Ready-made sample

[Import](#) [Running instructions](#)

### Build it yourself

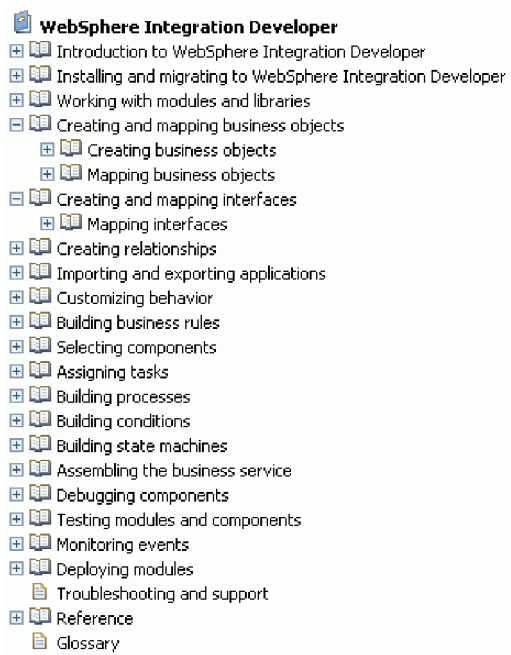
[Step-by-step instructions](#)

---

## インフォメーション・センター

インフォメーション・センターは、製品に関する詳細な情報を提供します。インフォメーション・センターには、チュートリアルおよびサンプルの他に、WebSphere Integration Developer のそれぞれのツールに関する追加の概念情報、タスク情報、および参照情報もあります。

目的の情報を素早く見つけるには、インフォメーション・センターのナビゲーションを使用します。インフォメーション・センター全体を検索することもできます。検索結果を選択すると、検索対象に関する詳細情報を含むさまざまなトピックと、関連情報へのリンクが表示されます。





---

## 特記事項

The XDoclet Documentation included in this IBM® product is used with permission and is covered under the following copyright attribution statement: Copyright (c) 2000-2004, XDoclet Team. All rights reserved.

Portions based on *Design Patterns: Elements of Reusable Object-Oriented Software*, by Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, Copyright (c) 1995 by Addison-Wesley Publishing Company, Inc. All rights reserved.

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

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

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

以下の保証は、国または地域の法律に沿わない場合は、適用されません。IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態を提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任または保証条件は適用されないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

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

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

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

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

Intellectual Property Dept. for Rational Software  
IBM Corporation  
20 Maguire Road  
Lexington, Massachusetts 02421-3112  
U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で 사용할 수 있습니다, 有償の場合もあります。

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

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

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

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

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

著作権使用許諾:

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

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次のように、著作権表示を入れていただく必要があります。

(C) (お客様の会社名) (年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 (C) Copyright IBM Corp. 2000, 2006. All rights reserved.

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

## プログラミング・インターフェース情報

プログラミング・インターフェース情報は、プログラムを使用してアプリケーション・ソフトウェアを作成する際に役立ちます。

一般使用プログラミング・インターフェースにより、お客様はこのプログラム・ツール・サービスを含むアプリケーション・ソフトウェアを作成することができます。

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

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

## 商標

<http://www.ibm.com/legal/copytrade.shtml> を参照してください。









Printed in Japan

SD88-6686-02



日本アイ・ビー・エム株式会社  
〒106-8711 東京都港区六本木3-2-12