IBM® WebSphere Commerce

ストア開発ガイド

バージョン 5.5

IBM® WebSphere Commerce

ストア開発ガイド

バージョン 5.5

ご注意!

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。

本書の内容は、新版で特に指定のない限り、IBM WebSphere Commerce Business Edition バージョン 5.5、 IBM WebSphere Commerce - Express バージョン 5.5、 IBM WebSphere Commerce Professional Edition バージョン 5.5 (プロダクト番号 5724-A18)、および以降のすべてのリリースとモディフィケーションに適用されます。

本書は、上記の製品、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。製品のレベルにあった版を使用していることをご確認ください。

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： IBM® WebSphere Commerce
Store Development Guide
Version 5.5

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 2003.10

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

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

目次

はじめに

「*IBM® WebSphere® Commerce ストア開発ガイド*」では、 WebSphere Commerce ストア・アーキテクチャーとストアの開発過程について説明しています。特に、以下のトピックについて詳しく説明しています。

- さまざまなストア開発プロセス
- WebSphere Commerce によってサポートされるビジネス・モデル
- WebSphere Commerce アーキテクチャー
- ストアフロントの開発
- ストア・データの開発
- ストア・データ・アーキテクチャー
- ストア・データの情報モデル
- ストアのグローバル化
- ストアへのアクセス制御の追加
- ストアのパッケージ化
- ストアの発行
- ストアへの WebSphere Commerce フィーチャーの追加

本書の表記規則と用語

本書では、以下のような強調表示の規則を使用しています。

太文字	コマンドまたはグラフィカル・ユーザー・インターフェース (GUI) のコントロール (フィールド、アイコン、またはメニュー選択項目の名前など) を表します。
モノスペース (Monospace)	表示どおりに入力するテキストの例、ファイル名、およびディレクトリーのパスと名前を表します。
イタリック	単語の強調のために使用されます。イタリックは、ご使用のシステムに該当する値に置き換える名前も表します。

このアイコンはヒントを表します。これは、タスクを完了するのに役立つ可能性のある追加情報です。

重要

この種の節は、特に重要な情報を強調します。

注

この種の節は、重要な情報を強調します。

Business — WebSphere Commerce Business Edition に固有の情報を示します。

Professional — WebSphere Commerce Professional Edition に固有の情報を示します。

Express — WebSphere Commerce - Express に固有の情報を示します。

Developer — Professional Business WebSphere Commerce 開発環境に固有の情報を示します。開発環境は WebSphere Commerce Studio バージョン 5.5 です。

Express WebSphere Commerce 開発環境に固有の情報を示します。開発環境は WebSphere Commerce - Express Developer Edition バージョン 5.5 です。

AIX — AIX® 上で実行しているプログラムに固有の情報を示します。

400 — OS/400® 上で実行しているプログラムに固有の情報を示します。

Linux — WebSphere Commerce for Linux for xSeries™ に固有の情報、 WebSphere Commerce for Linux for Eserver zSeries™ and S/390® に固有の情報、 WebSphere Commercefor Linux for Eserver iSeries™ に固有の情報、 WebSphere Commerce for Linux for Eserver pSeries™ に固有の情報を示します。

Solaris — Solaris オペレーティング環境で実行しているプログラムに固有の情報を示します。

2000 — Windows® 2000 上で実行しているプログラムに固有の情報を示します。

DB2 — DB2 Universal Database™ に固有の情報を示します。

Oracle — Oracle9i Database に固有の情報を示します。

本書の変数

本書の重要な変数の一部を以下に示します。

businessmodel
作業しているサンプル・ビジネス・モデルの名前 (消費者向けや B2B 向けなど)。

Professional Business *cell_name*
セルとは、WebSphere Application Server 分散ネットワーク中の 1 つ以上のノードの任意の論理グループのことです。この定義では、ノードとは単一の

WebSphere Application Server です。 WebSphere Application Server デプロイメント・マネージャーの単一オカレンスによって管理される 1 つ以上のセルは、 WebSphere Application Server デプロイメント・マネージャー・セルと呼ばれます。

Express *cell_name*

WebSphere Commerce - Express では、cell_name は host_name と同等です。

host_name

この変数は、WebSphere Commerce Server の完全修飾ホスト名を表します (たとえば、 `server.mydomain.ibm.com` は完全修飾されています)。

instance_name

この変数は、作業している WebSphere Commerce インスタンスの名前 (mall1 など) を表します。

storedir

この変数は、ご使用のストアがあるストア・ディレクトリーの名前を表します。

WAS_instance_name

この変数は、 WebSphere Commerce インスタンスと関連付けられている WebSphere Application Server セルの名前を表します。

パス変数

本書では、以下の変数を使用してディレクトリー・パスを表しています。

WC_installdir

これは WebSphere Commerce のインストール・ディレクトリーです。オペレーティング・システム別の WebSphere Commerce のデフォルト・インストール・ディレクトリーを以下に示します。

AIX /usr/WebSphere/CommerceServer55

400 /QIBM/ProdData/CommerceServer55

Linux /opt/WebSphere/CommerceServer55

Solaris /opt/WebSphere/CommerceServer55

Windows C:¥Program Files¥WebSphere¥CommerceServer55

WCDE_installdir

WebSphere Commerce 開発環境のインストール・ディレクトリー。 WebSphere Commerce Business Edition および WebSphere Commerce Professional Edition では、開発環境は WebSphere Commerce Studio バージョン 5.5 です。デフォルトのインストール・ディレクトリーは C:¥WebSphere¥CommerceStudio55 です。

WebSphere Commerce - Express の場合、開発環境は WebSphere Commerce - Express Developer Edition バージョン 5.5 です。デフォルトのインストール・ディレクトリーは C:¥WebSphere¥CommerceDev55 です。

400 *WC_userdir*

これは、WebSphere Commerce によって使用され、ユーザーが構成する必要があり変更を加えることができる、すべてのデータのディレクトリーです。この種のデータの一例としては、WebSphere Commerce インスタンス情報があります。このディレクトリーは、 OS/400 に固有です。

WC_userdir 変数は以下のディレクトリーを表します。

```
/QIBM/UserData/CommerceServer55
```

WAS_installdir

これは WebSphere Application Server のインストール・ディレクトリーです。オペレーティング・システム別の WebSphere Application Server のデフォルト・インストール・ディレクトリーを以下に示します。

AIX	/usr/WebSphere/AppServer
400	/QIBM/ProdData/WebAS5/Base
Linux	/opt/WebSphere/AppServer
Solaris	/opt/WebSphere/AppServer
Windows	C:¥Program Files¥WebSphere¥AppServer

400 *WAS_userdir*

これは、WebSphere Application Server によって使用され、ユーザーが構成する必要があり変更を加えることができる、すべてのデータのディレクトリーです。この種のデータの一例としては、WebSphere Application Server インスタンス情報があります。このディレクトリーは、 OS/400 に固有です。

WAS_userdir 変数は以下のディレクトリーを表します。
/QIBM/UserData/WebAS5/Base/*WAS_instance_name*

WC_userdir

WC_userdir 変数は以下のディレクトリーを表します。
/QIBM/UserData/WebAS5/Base/*WAS_instance_name*

workspace_dir

開発環境で使用されます。この変数は *drive*:¥WebSphere¥workspace_db2 を表します。

最新情報の入手先

本書は、将来、更新される可能性があります。以下の WebSphere Commerce Web サイトで更新情報をチェックしてください。

http://www.ibm.com/software/commerce/library/

更新では、新しい情報が含まれていることがあります。

第 1 部 概要

第 1 章 ストア開発の概要

この章では、WebSphere Commerce におけるサイトおよびストア開発プロセスを概説し、本書で論じられる概念の多くを紹介します。

注: 本書では、*ストア開発* という用語は、単一のストアの作成に関係するプロセスと複数のストアまたはサイト環境の作成に関係するプロセスの両方を指して使用されています。

WebSphere Commerce のストア開発について

WebSphere Commerce を使用してサイトまたはストアの開発を始める前に、以下のエレメントが開発プロセスにどのような影響を与えるかを理解する必要があります。これらの各エレメントについてはこの章で紹介しますが、たいていは本書の各所でより詳細に説明しています。さらに場合によっては、WebSphere Commerce ライブラリーの他の文書で詳細に説明されていることもあります。

WebSphere Commerce でストアを開発する方法をどのように選択するかは、以下のエレメントに応じて異なります。

- ストアの目的
- ストアを表現するビジネス・モデル
- 開発中のストアの数およびタイプ
- ストアの土台
- 必要なカスタマイズの程度

ストアの目的

ストアは通常、以下のいずれかの目的で開発されます。

- 実動: 実動ストアとは、顧客またはパートナーが使用できる状態になっている、実稼働環境中の十分な機能を備えたストアのことです。
- デモ: デモ・ストアは、セールスの目的で特定の機能を説明します。デモ・ストアは、一部の機能しか備えていない場合があります。
- サンプル: サンプル・ストアとは、オンライン・ストアを作成する際のベースとして使用するように設計されている、十分な機能を備えたストアのことです。

ストアを表現するビジネス・モデル

ストアを開発する前に、WebSphere Commerce によってサポートされるビジネス・モデルのどれがそのストアを最もよく表現するかを理解する必要があります。WebSphere Commerce は、以下のビジネス・モデルの 1 つのインスタンスであるサイトまたはストアをサポートします。

注: ビジネス・モデルの詳細については 17 ページの『第 2 章 WebSphere Commerce でサポートされるビジネス・モデル』で説明しますが、ここでは各ビジネス・モデルの概要を示します。

- ダイレクト・セールス・ビジネス・モデル: 旧リリースと同様に、 WebSphere Commerce はダイレクト・セールスのビジネス・モデルをサポートします。 WebSphere Commerce を使用して、商品、サービス、または情報を含む、ビジネスと消費者との間、または 2 つのビジネスやパーティーの間での直接の商取引をサポートするサイトまたはストアを作成できます。 WebSphere Commerce は、以下のタイプのダイレクト・セールス・ビジネス・モデルをサポートします。
  - 消費者向けビジネス・モデル: 消費者向けは、商品、サービス、または情報を含む、ビジネスと消費者との間での商取引をサポートします。消費者は通常、消費者向けのシナリオに基づいて、商品またはサービスをビジネスから直接購入します。
  - Business B2B 向けビジネス・モデル: B2B 向けは、商品、サービス、または情報を含む、2 つのビジネス間やパーティーの間での商取引をサポートします。典型的な B2B 向けの取引は、バイヤー、サプライヤー、製造メーカー、販売店、ディストリビューター、および取引先の間で実施されます。
- Business ホスティングビジネス・モデル: WebSphere Commerce は、インターネット・サービス・プロバイダーまたは他のホスティング・プロバイダーによって、マーチャントまたは他のビジネスのホスティングもサポートします。
- Business バリュー・チェーン・ビジネス・モデル: バリュー・チェーンは、複数の企業やパーティーが関係する取り引きをサポートします。製品、商品、サービス、または情報は、バリュー・チェーンのパーティーを介して、生産者からエンド・ユーザーに提供されます。バリュー・チェーンには、関係および管理という局面もあります。つまり、バリュー・チェーン内のパートナーや企業の関係を管理したり、それらのパーティーに管理サービスを提供することができます。 WebSphere Commerce は以下の 2 つのタイプのバリュー・チェーンについて、取り引きと関係管理をサポートします。
  - デマンド・チェーン: デマンド・チェーンは、間接セールス・チャネルおよびダイレクト・セールス・チャネルの両方をサポートします。
  - サプライ・チェーン: サプライ・チェーンは、商品の調達およびソーシングをサポートします。 WebSphere Commerce は、プライベート・マーケットプレイスを介して商品の調達をサポートします。プライベート・マーケットプレイスは、契約関係にあるバイヤーに対して商品およびサービスを販売するためのフォーラムを取引先に提供します。

注: ビジネス・モデルの詳細については 17 ページの『第 2 章 WebSphere Commerce でサポートされるビジネス・モデル』で説明します。

開発中のストアの数

ビジネスによっては、複数のストアまたは複数のタイプのストアを開発することが必要になる場合もあります。たとえば、顧客に直接販売するビジネスの場合、必要なのは顧客がアクセスして商品を購入する 1 つのストアのみです。ただし、デマンド・チェーンをサポートしている場合、ビジネス用のメインとなる 1 つのハブ・ストアと、チャネルに接続したり管理するためのいくつかのストアが必要となる場合があります。さらに、チャネル内の組織またはビジネス用のホスト・ストアを選択することもできます。デマンド・チェーンの詳細については、 17 ページの『第 2 章 WebSphere Commerce でサポートされるビジネス・モデル』を参照してください。

マーチャントまたは他のビジネスに対してストアをホスティングするビジネスであれば、マーチャントを管理したり登録要求を処理したりするためのハブ・ストアを開発することと、ホスティングするストア用のサイトを開発するメソッドを持つことも必要です。ホスティング・ビジネス・モデルの詳細については、17 ページの『第 2 章 WebSphere Commerce でサポートされるビジネス・モデル』を参照してください。

WebSphere Commerce を使用して、多数のタイプのストアおよびサイトごとに複数のストアを開発できます。ストアのタイプの詳細については、71 ページの『第 7 章 ストア・アーキテクチャー』を参照してください。

ストアの土台

WebSphere Commerce を使用してサイトまたはストアを作成する前に、どこから開発を始めるかを決定する必要があります。 WebSphere Commerce には、開発の開始点として使用できるいくつかのサンプルが備えられています。または、開発を最初から始めることもできます。 WebSphere Commerce に備えられているサンプルの詳細については、「*WebSphere Commerce サンプル・ストア・ガイド*」を参照してください。

サンプルからの開始

WebSphere Commerce に備えられているサンプルは、ストア・アーカイブとしてパッケージ化されています。

ストア・アーカイブ: ストア・アーカイブ・ファイル (.sar) は、サイトまたはストアの作成に必要なすべての資産が含まれている ZIP アーカイブ・ファイルです。これは、主にストアをパッケージ化して送付するための手段として使用されています。ストア・アーカイブを WebSphere Commerce Server に発行するだけで、表示、ブラウズ、および買い物を実行できる機能ストアを作成できます。

一般に、ストア・アーカイブは以下のファイルで構成されます。

- Web 資産: HTML ファイル、JSP ファイル、イメージ、グラフィックス、および組み込みファイルなど、ストア・ページの作成に使用するファイル。
- プロパティー・リソース・バンドル: ストア・ページのテキストが入っています。ストアが複数の言語をサポートする場合、ストア・アーカイブには、サポートされる言語ごとに複数のリソース・バンドル、さらにデフォルトのリソース・バンドル (ロケールを含まない) が入っています。たとえば、`AddressText_en_US.properties` および `AddressText.properties` などです。
- ストア・データ資産: データベースにロードされるデータ。ストア・データ資産には、キャンペーン、カタログ・エントリー、通貨、フルフィルメント情報、価格設定、配送、ストア、税情報などのデータが含まれます。ストア・データ資産の詳細なリストについては、123 ページの『第 6 部 ストア・データの開発』を参照してください。

  WebSphere Commerce で提供されるサンプルのストア・アーカイブにあるストア・データ資産は、ローダー・パッケージに有効な、整形式の XML ファイルです。ストア・アーカイブ XML ファイルは移植可能であることが意図されており、データベースの特定のインスタンスに固有な、生成された基本キーを含めるべきではありません。その代わりに、ストア発行時に ID リゾルバーによって解決される内部別名が使用されます。これらの規則を使用すると、サンプルのスト

ア・アーカイブを移植可能にすることができます。詳細については、 355 ページの『第 9 部 ストアのパッケージ化』を参照してください。

ローダー・パッケージの詳細については、 379 ページの『第 37 章 ストア・データのロードの概要』を参照してください。

注: ストア・データ資産には、契約を作成するための情報も含まれます。契約情報はローダー・パッケージによってロードされません。この情報により契約を作成するコマンドに対する入力データが備えられます。

- 支払資産: WebSphere Commerce Payments の構成情報です。支払い情報はローダー・パッケージによってロードされません。この情報により WebSphere Commerce Payments を構成するコマンドに対する入力データが備えられます。
- 記述子: ストア・アーカイブおよびその発行方法について記述する XML ファイルです。これらのファイルには、`store-refs.xml`、`ibm-wc-load.xml`、`unpack.xml`、および `ForeignKeys.dtd` が含まれます。

ストア・アーカイブの詳細については、 357 ページの『第 35 章 ストアのパッケージ化』を参照してください。

ストア・アーカイブの発行: 管理コンソールの発行ユーティリティーを使用するか、またはコマンド行から、ストア・アーカイブを発行することができます。ストア・アーカイブを発行する方法の詳細については、 WebSphere Commerce オンライン・ヘルプのトピック『ストア・アーカイブの発行』を参照してください。

サンプルのタイプ: WebSphere Commerce に備えられているサンプルは、以下のように分類されます。

- 複合ストア・アーカイブ
- Business コンポーネント・ストア・アーカイブ
- 基本ストア・アーカイブ

複合ストア・アーカイブ: 複合ストア・アーカイブには、作業サイトを作成するために必要なすべての資産が含まれています。 WebSphere Commerce に備えられているサンプルの複合ストア・アーカイブには、通常は対応するビジネス・モデルに適した環境を作成するための、組織構造、事前定義のユーザー役割、および必要なアクセス制御ポリシーが含まれています。複合ストア・アーカイブには、必要なストアまたはサイトを作成するために必要な資産も含まれています。たとえば、デマンド・チェーンのサンプルの複合ストア・アーカイブには、チャネル・ハブ・サイト、共用カタログ、および販売店とディストリビューターのストアのサンプルが含まれます。

WebSphere Commerce には、独自のストアを作成するための基礎として活用できる、十分な機能を備えたサンプルのオンライン・サイトを含む、いくつかの複合ストア・アーカイブが組み込まれています。それらのサンプルには、ダイレクト・セールス・ストア (消費者向けおよび Business B2B 向けの両方)、 Business デマンド・チェーン・ビジネス、 Business サプライヤー・ビジネス、および Business ホスティング・サイトが組み込まれています。それらには、今日の代表的な電子商取引サイトで現在一般的に使用されている機能のほとんどがインプリメントされており、必要なすべてのストア資産が備えられています。 WebSphere

Commerce に備えられているサンプルの詳細については、「*WebSphere Commerce サンプル・ストア・ガイド*」を参照してください。

サンプルの複合ストア・アーカイブから開始する理由: サンプルの複合ストア・アーカイブを使用して開始すると、必要なすべてのデータが WebSphere Commerce Server にロードされて、十分な機能を備えたサイトが作成されます。

WebSphere Commerce では、機能的なサイトを作成するために特定のデータを WebSphere Commerce Server データベースにロードし、スキーマによって決定されるオーダーにそのデータをロードする必要があります。サンプルのコンポーネント・ストア・アーカイブには WebSphere Commerce Server データベースで必要なオーダーや構造のデータがすべて含まれているため、独自に作成するサイトのベースとしてその 1 つを使用するなら、作成作業の初期に必要な時間を大幅に節約できます。

サンプルの複合ストア・アーカイブを発行してから、それをストアの必要に応じて編集することができます。この編集作業はかなりのものになる場合も、ごく簡単な場合もあります。たとえば、WebSphere Commerce で利用できるツールを使用してデータを編集したり、開発環境を使用してストア・ページのルック・アンド・フィールを変更したりするだけで十分かもしれません。あるいはデータに対して広範囲に渡る変更を加えるために XML ファイルまたはデータベースを直接編集したり、ストア・フローとフィーチャーを変更するためにストア・ページを再作成したりする必要がある場合もあります。または、ストア開発にサンプル・ストア資産と開発する新規ストア資産の組み合わせを使用すると、最も有効なストア開発の方法になる場合があります。たとえば、あるサンプル・ストアの中のデータベース資産のいくつかが実際のストアのニーズによく似ているが、実際のストアのページのフローについてはそうではない場合、そのストアからコピーしたデータベース資産をカスタマイズし、しかもまったく新規の Web 資産を開発することができます。ストア・データの編集の詳細については、123 ページの『第 6 部 ストア・データの開発』を参照してください。

コンポーネント・ストア・アーカイブ: Business 複合ストア・アーカイブを形成する各部分は、個別のストア・アーカイブとしても使用できます。これらのストア・アーカイブは、コンポーネント・ストア・アーカイブとして知られています。コンポーネント・ストア・アーカイブは、組織構造と定義済みユーザー役割を含む組織構造ストア・アーカイブか、または他のタイプのストアがリソースとして使用できるファイルやデータ資産の集合です。 WebSphere Commerce に備えられているサンプルの詳細については、「*WebSphere Commerce サンプル・ストア・ガイド*」を参照してください。

サンプルのコンポーネント・ストア・アーカイブから開始する理由: サンプルのコンポーネント・ストア・アーカイブ、またはそれらの組み合わせから開始すると、サンプルの複合ストア・アーカイブから開始するよりも柔軟性が高くなります。それは、複合ストア・アーカイブの発行によって、十分に機能的なサイトが作成されるためです。このサイトの一部は必要に適しており、他の部分は必要に適していない場合があります。たとえば、ストア・ページのフローが、提供されているサンプルのいずれのものともかなり異なる場合、または WebSphere Commerce Server データベース・スキーマを大幅にカスタマイズする予定の場合は、提供されているサンプルの全体ではなく、その特定の一部だけを発行することもできます。たとえば、

サンプルの組織構造だけを発行してから、すべての資産を開発して、サイトにストアを作成することができます。または、サンプルの組織構造に加えて、組織構造内にストアを作成するかまたは他のストアが使用できるリソースを提供する、1 つ以上のサンプルのコンポーネント・アーカイブを発行することもできます。

注: バリュー・チェーン・ビジネス・モデルのインスタンスを作成する場合、複数のエンティティーを含むサイトに必要な組織構造はとても複雑なので、サンプルの組織構造の発行から開始することをお勧めします。 WebSphere Commerce で組織構造がどのように機能するかの詳細については、27 ページの『WebSphere Commerce 組織構造について』を参照してください。

基本ストア・アーカイブ: WebSphere Commerce には、WebSphere Commerce Server 内にストアを作成するために必要な資産の最小セットを提供する、サンプルの基本ストアも備えられています。

サンプルの基本ストアから開始する理由: サンプルの基本ストアから開始すると、ストア ID を使用して JSP ファイルを呼び出せるストア・エンティティーを、Commerce サーバー内に確立することができます。 WebSphere Commerce に備えられているサンプル・ストアのいずれともかなり異なるストアを作成する場合、サンプルの基本ストアから開始することができます。この基本ストアから開始すると、開発者は必要に応じて資産を追加することができ、ストアに適用されない資産を除去または変更する必要がありません。 WebSphere Commerce に備えられている基本ストアの詳細については、「*WebSphere Commerce サンプル・ストア・ガイド*」を参照してください。

注: サンプルの基本ストアを、提供されているサンプルの組織構造と組み合わせて使用することもできます。

最初からの開始

最初から開始する、つまり WebSphere Commerce に備えられているどのサンプルも使用せずに開始することもできます。

必要なカスタマイズの程度

ストアの土台を、サンプルのストア、サンプルの組織構造、またはサンプルの基本ストアのいずれにするか、あるいは最初から作成するかを決めた後、そのベースに対してどのような変更を加えるかを決める必要があります。通常、WebSphere Commerce でのたいていのストア開発は、以下のいずれかのカテゴリーに分類されます。

- 新機能の追加やストア・フローの変更を含む、機能の追加または変更。
- ストアのルック・アンド・フィールの作成または変更。
- ストア・データの作成または変更。

多くの場合、ストア開発作業には、これら 3 つのすべてが関係してきます。

ストア機能の追加または変更

ストアのフローの変更やストアへの新規機能の追加など、ストア機能の追加および変更を行うと、通常はビジネス・ロジックの変更が必要となります。ビジネス・ロジック開発用のツール (コマンドの作成と拡張、カスタマイズされたコードの作

成、ビジネス・ロジックのインプリメントなどのツール) については、「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」に説明されています。

注: ビジネス・ロジックを作成または変更する開発者は、 Java™、 Enterprise JavaBeans™、 WebSphere Studio Application Developer、および J2EE のプログラミング・スキルがなければならず、 WebSphere Commerce のプログラミング・モデルとオブジェクト・モデルに精通している必要があります。

WebSphere Commerce アクセラレーターによって、消費者向けおよび B2B 向けのサンプル・ストアが提供する一部の機能およびストアのフローを変更することができます。変更可能なフローおよび機能とその変更方法については、WebSphere Commerce オンライン・ヘルプを参照してください。

ストアのルック・アンド・フィールの作成または変更

ストアのルック・アンド・フィールの変更には、通常はストアフロントの変更が関係します。ストアフロント資産には、HTML ページ、JSP ファイル、スタイルシート、イメージ、グラフィックス、およびその他のマルチメディア・ファイル・タイプなどの Web 資産が含まれます。ストアフロント資産の開発には、サンプル・ストア・ページのカスタマイズ、そのページの独自の既存のページへの置き換え、新規ページの作成、またはこれらの 3 つすべてを組み合わせて実行することが含まれます。

WebSphere Commerce には、ストアフロント資産を作成または編集するための以下のツールが用意されています。

- WebSphere Studio Application Developer

  WebSphere Studio Application Developer (WebSphere Commerce Studio に同梱されている) には、ストアフロント資産の作成と編集に必要なツールが含まれています。ストアフロント資産には、HTML、グラフィックス、マルチメディア、および JavaServer Pages (JSP) の各ファイルが含まれます。 Page Designer (WebSphere Studio Application Developer に付属) を使用すると、アニメーション・イメージに加えて、HTML または JSP ファイルを作成できるようになります。また、自分で選んだ別の Web 開発ツールを使用するよう WebSphere Studio Application Developer を構成することもできます。独自のツールを登録する方法の詳細については、 WebSphere Studio オンライン・ヘルプを参照してください。

  WebSphere Commerce Studio のツールを使用してストアフロント資産を作成および編集する方法の詳細については、 WebSphere Commerce Studio オンライン・ヘルプを参照してください。 WebSphere Commerce にストアフロントを作成する方法の詳細については、 85 ページの『第 8 章 ストアフロントの開発』を参照してください。

  注: ストアフロントを作成または変更する開発者は、 Java、 JavaScript™、 HTML、JSP テクノロジーのプログラミング・スキルがなければならず、 WebSphere Commerce のストア・アーキテクチャーに精通している必要があります。

- WebSphere Commerce アクセラレーター

  WebSphere Commerce アクセラレーターには、ストアのルック・アンド・フィールに変更を加える以下のツールが含まれます。

  – 「ページの変更」ノートブック

- 「ロゴのアップロード」ノートブック
- 「スタイルの変更」ウィザード
- 「ファイルの管理」ノートブック
- 「ストア・プロファイル」ノートブック

これらのツールの詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

注: 上記のツールは、消費者向けサンプル・ストアを基にしたストアと、「ストア作成 (Store Creation)」ウィザードを使用して作成した (ホスティング・モデルおよびデマンド・チェーン・モデルの) Business ホストされるストアのみ処理します。

ストア・データの作成または変更

ストアのデータベース資産を開発および編集するための選択肢がいくつかあります。

- WebSphere Commerce ローダー・パッケージ

  WebSphere Commerce ローダー・パッケージは、データを準備して WebSphere Commerce データベースにロードするユーティリティーで主に構成されます。WebSphere Commerce データベースに大量のデータをロードしたり、データベースのデータを更新したりする場合には、ローダー・パッケージを使用してください。このパッケージ内のローダー・ユーティリティーは、有効かつ形式化された XML をデータベースにデータをロードするための入力として使用します。 XML 文書のエレメントはデータベースのテーブル名にマップされ、エレメント属性は列にマップされます。

  ローダー・パッケージを使用してデータ資産を開発およびロードする方法の詳細については、 379 ページの『第 37 章 ストア・データのロードの概要』を参照してください。

  いつ WebSphere Commerce ローダー・パッケージを使用するか: データベース資産を初めて WebSphere Commerce データベース資産にロードするとき、およびそれらを更新するときに WebSphere Commerce ローダー・パッケージを使用します。ローダー・パッケージを使用して、バック・エンド・システムからの定期的なデータの供給を自動化することもできます。

- WebSphere Commerce アクセラレーター

  WebSphere Commerce アクセラレーターは、さまざまなストア操作でオンライン・ストアを保守するために使用される、オンライン・ツールのワークベンチです。ただし、WebSphere Commerce アクセラレーターによってデータの編集が可能になるので、少量のデータを変更するときには特に、それをストア開発ツールとして使用することもできます。 WebSphere Commerce アクセラレーターを使用して編集できるデータベース資産のリストについては、 123 ページの『第 6 部 ストア・データの開発』を参照してください。

  WebSphere Commerce アクセラレーターをいつ使用するか: WebSphere Commerce アクセラレーターは、データの作成または更新の際に使用します。

- データベースを直接編集する

  SQL の挿入、更新、または削除を使用してデータベースを直接編集することも、いつでも可能です。

注: SQL は、データベース固有のものです。 Oracle で必要とされる SQL 構文は、異なっています。 SQL ステートメントには必然的にデータベース固有の値が含まれることになり、そのような SQL ステートメントは、別の WebSphere Commerce Server インスタンスで再使用できないことがあります。

ストア・データを作成または変更する開発者は、 WebSphere Commerce のストア・アーキテクチャー、ストア・データ、およびストア・アーカイブに精通していなければなりません。カスタマイズされたストア機能をインプリメントしたり既存のデータベース情報と統合したりするために、 WebSphere Commerce のデータベース・スキーマの変更や拡張を実行するには、開発者が DB2® または Oracle のデータベース管理者スキルを有している必要があります。

シナリオ: 実動ストアの開発およびデプロイ

このセクションでは、WebSphere Commerce によって実動ストアを開発するための、推奨されるシナリオを概説します。

表 1. シナリオ: 実動ストアの開発およびデプロイ

タスク	サブタスク	参照
ビジネスを反映しているのは、サポートされているどのビジネス・モデルであるかを判別します。		17 ページの『第 2 章 WebSphere Commerce でサポートされるビジネス・モデル』
ストア・フローを判別します。		83 ページの『第 4 部 ストアフロントの開発』
ユース・ケースを作成します。		83 ページの『第 4 部 ストアフロントの開発』
WebSphere Commerce に備えられているサンプル・ストアを分析します。		「*WebSphere Commerce サンプル・ストア・ガイド*」
どのサンプル・ストアまたは他のサンプルを開始点として使用するかを決めます。		「*WebSphere Commerce サンプル・ストア・ガイド*」

表 1. シナリオ: 実動ストアの開発およびデプロイ (続き)

ストア資産のベースライン・セットを作成します。	開発環境でプロジェクトを作成します (WebSphere Commerce 開発環境に固有)。	WebSphere Studio 製品資料
	開発環境にサンプルのストア・アーカイブの 1 つを発行します。	WebSphere Commerce オンライン・ヘルプのヘルプ・トピック『ストア・アーカイブの発行』
	可能であれば、 WebSphere Commerce アクセラレーターでフロー変更ツールを使用してストアを構成します。	WebSphere Commerce オンライン・ヘルプのヘルプ・トピック『WebSphere Commerce Accelerator を使用したストア・フローの変更』
	必要に応じてデータベース・スキーマを変更します。	「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」
	ソース制御システム内のストア資産を検査して、マスター・コピーを作成します。	「*WebSphere Commerce V5.5 Customization and Deployment Handbook*」(SG24-6969) レッドブック
ストアをベースライン資産から作成するために必要な開発を判別します (ストアフロント、データ、およびサーバー開発)。	ストアのルック・アンド・フィールの変更を決めます。	「*WebSphere Commerce サンプル・ストア・ガイド* 83 ページの『第 4 部 ストアフロントの開発』」
	ストア・ページのキャッシング・ストラテジーを決めます。	83 ページの『第 4 部 ストアフロントの開発』
	ストア・データの変更を決めます。	「*WebSphere Commerce サンプル・ストア・ガイド* 123 ページの『第 6 部 ストア・データの開発』」
	サンプル・ストアで使用されているインプリメンテーションについて理解します。	「*WebSphere Commerce サンプル・ストア・ガイド*」
	既存のサーバー機能を分析して、強化またはカスタマイズが必要な箇所を判別します。	「WebSphere Commerce オンライン・ヘルプ」
	バック・エンド・システムとの統合がどの程度必要かを判別します。	「WebSphere Commerce オンライン・ヘルプ」
チーム環境をセットアップします。	各開発者は開発者プロジェクトを IDE にセットアップして、ソース制御システム内のマスター・コピーにある資産と、製品に同梱されているサーバー資産をプロジェクトに移植します。	「*WebSphere Commerce V5.5 Customization and Deployment Handbook*」(SG24-6969) レッドブック

表 1. シナリオ: 実動ストアの開発およびデプロイ (続き)

	開発者は資産のベースライン・セットを実行します。	
	チームは顧客および管理者の視点から、ストアの既存の機能に精通するようにします。	「*WebSphere Commerce サンプル・ストア・ガイド*」
ストア資産を開発します。	ストアフロント資産を変更および強化するか、または新規のストアフロント資産を作成します。	83 ページの『第 4 部 ストアフロントの開発』
	追加のサーバー機能を開発します (新規のコマンド、EJB の記述、バック・エンド・システムとの統合)	「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」
	データを変更して、追加のデータを作成します。	123 ページの『第 6 部 ストア・データの開発』
実動のための利用可能なデータを作成します。		123 ページの『第 6 部 ストア・データの開発』
開発された資産を実動へと展開します。		「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」 追加情報は、「*WebSphere Commerce V5.5 Customization and Deployment Handbook*」 (SG24-6969) レッドブックにもあります。

第 2 部 WebSphere Commerce によってサポートされるビジネス・モデル

第 2 章 WebSphere Commerce でサポートされるビジネス・モデル

WebSphere Commerce を使用してストアまたはサイトの開発を始める前に、WebSphere Commerce がサポートするビジネス・モデルについて理解する必要があります。 WebSphere Commerce を使用して作成するストアのほとんどは、これらのビジネス・モデルの 1 つのインスタンスとなります。

注: WebSphere Commerce を使用して、この章で説明するビジネス・モデルに準拠しないストアを作成することもできます。

WebSphere Commerce でサポートされるビジネス・モデルについて

WebSphere Commerce は、以下のビジネス・モデルの 1 つに該当するビジネスをオンラインで展開するための、構造的なインフラストラクチャーを提供します。

- ダイレクト・セールス
- Business ホスティング
- Business バリュー・チェーン

ダイレクト・セールス

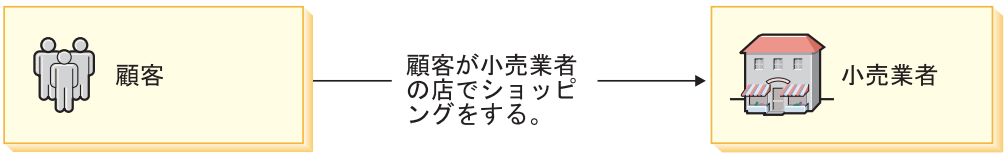
ダイレクト・セールスは、商品、サービス、または情報を含む、ビジネスと消費者との間または 2 つのビジネス間やパーティー間での、直接の商取引をサポートします。 WebSphere Commerce は、以下のタイプのダイレクト・セールス・ビジネス・モデルをサポートします。

- 消費者向け
- Business B2B 向け

消費者向け

消費者向けは、商品、サービス、または情報を含む、ビジネスと消費者との間での商取引をサポートします。消費者は通常、消費者向けのシナリオに基づいて、商品またはサービスをビジネスから直接購入します。

以下の図は、典型的な消費者向けビジネスを示しています。



この図に示されているように、典型的な消費者向けビジネスでは、消費者はビジネス (通常は小売業者) から直接購入します。このビジネスとは、小売業者、独自の小売アウトレットを通じて消費者に商品を直接販売する製造メーカー、または消費者

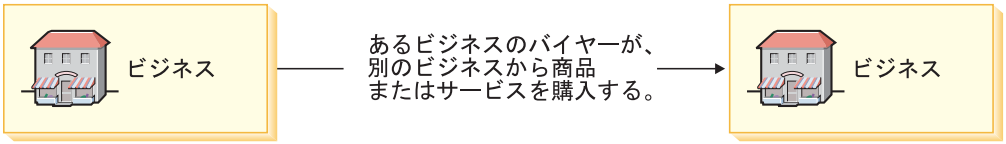
に対して商品の販売またはサービスの提供を直接行う他のビジネスなどです。たとえば、カタログを通じて消費者に直接販売するビジネスは、消費者向けビジネスといえます。

従来はビジネスと見なされなかった政府などの組織も、消費者向けビジネスと見なすことができます。政府は商品やサービスを顧客に直接提供することがあります。

B2B 向け

Business B2B 向けは、商品、サービス、または情報を含む、2 つのビジネス間やパーティー間での商取引をサポートします。典型的な B2B 向けの取引は、バイヤー、サプライヤー、製造メーカー、販売店、ディストリビューター、および取引先との間で実施されます。

以下の図は、典型的な B2B 向けビジネスを示しています。



典型的な B2B 向けビジネスでは、ビジネスが他のビジネスから商品またはサービスを直接購入します。売り手側のビジネスは、他のビジネスのバイヤーに販売する、卸売業者、ディストリビューター、製造メーカー、または小売業者などです。

従来はビジネスと見なされなかった政府やメディアなどの組織も、 B2B 向けビジネスと見なすことができます。政府は商品やサービスをビジネスに直接提供することがあります。

ホスティング

Business ホスティング・モデルは、インターネット・サービス・プロバイダー (ISP) または他のホスティング・プロバイダーによって、マーチャントまたは他のビジネスのホスティングをサポートします。

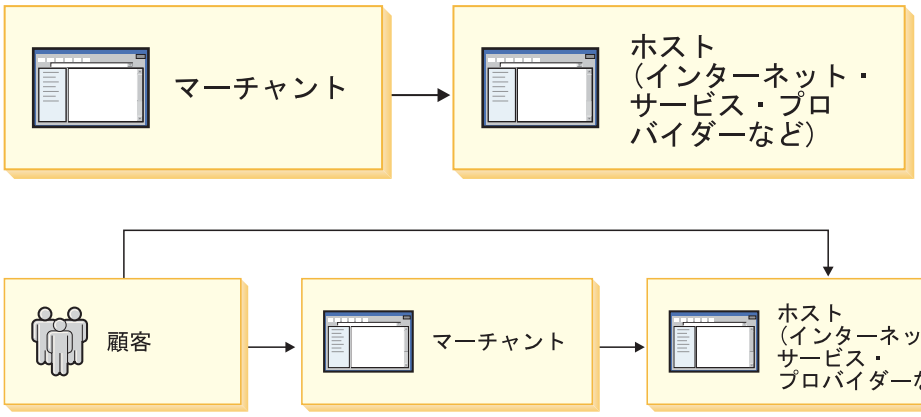
ホスティング・ビジネスには、以下の 2 つの側があります。

- ホストされるストア
- (オプション) プロバイダーによってホストされるストアを顧客が検索できるサイト

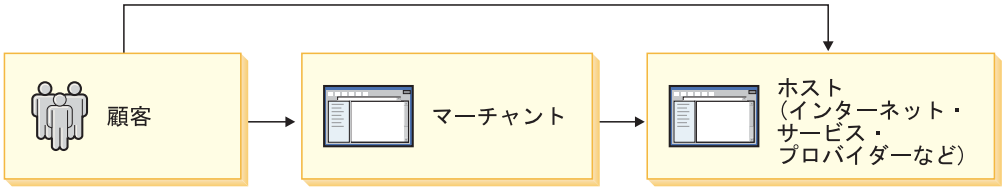
ホストされるストアとの関係を管理するために、ホスティング・モデルには、普通はハブ (WebSphere Commerce ではハブ・ストア) が組み込まれています。このハブには、マーチャントがストアの作成や管理を行える自己供給ツールと、ホスティング・プロバイダーがすべてのホストされるストアを管理できるツールが備えられています。

普通ホスティング・プロバイダーには、プロバイダーによってホストされるストアを顧客が検索してアクセスできるストアも組み込まれています。

以下の図は、ホスティングの例を示しています。



この例で、マーチャントはホストのサイトに入り、そのサイトによってホストされるサイトを作成します。ホストされるストアをマーチャントが管理できる単純な自己供給ツールが、ホスティング・プロバイダーに備えられている場合がよくあります。ホストされるストアがビジネス用にオープンしている場合に、顧客はホストのサイトを介したり、直接ホストされるストアに入ったりして、ストアにアクセスできます。



この例では、顧客はホストされるストアまたはビジネスに直接入るか、またはホストのサイトをブラウズしてからホストされるストアまたはビジネスに転送されるかを選択できます。

ホストされるストアは消費者向けストアに非常に似ています。 WebSphere Commerce サンプル・ストアにインプリメントされている両方のストアの特定の違いについては、「*WebSphere Commerce サンプル・ストア・ガイド*」を参照してください。

バリュー・チェーン

Business WebSphere Commerce バージョン 5.5 の新機能として、複数の企業が関係するオンライン商取引を可能にする機能があります。バリュー・チェーンは、複数の企業やパーティーが関係する取り引きをサポートします。製品、商品、サービス、または情報は、バリュー・チェーンのパーティーを介して、生産者からエンド・ユーザーに提供されます。バリュー・チェーンには、関係および管理という局面もあります。つまり、バリュー・チェーン内のパートナーや企業の関係を管理したり、それらのパーティーに管理サービスを提供することができます。

結果として、バリュー・チェーンはビジネスの 2 つの側を管理しなければなりません。それは顧客とダイレクト・セールス、およびチャネル・パートナーとサプライヤーです。個々の側に独自の管理チャネルと業務が必要です。

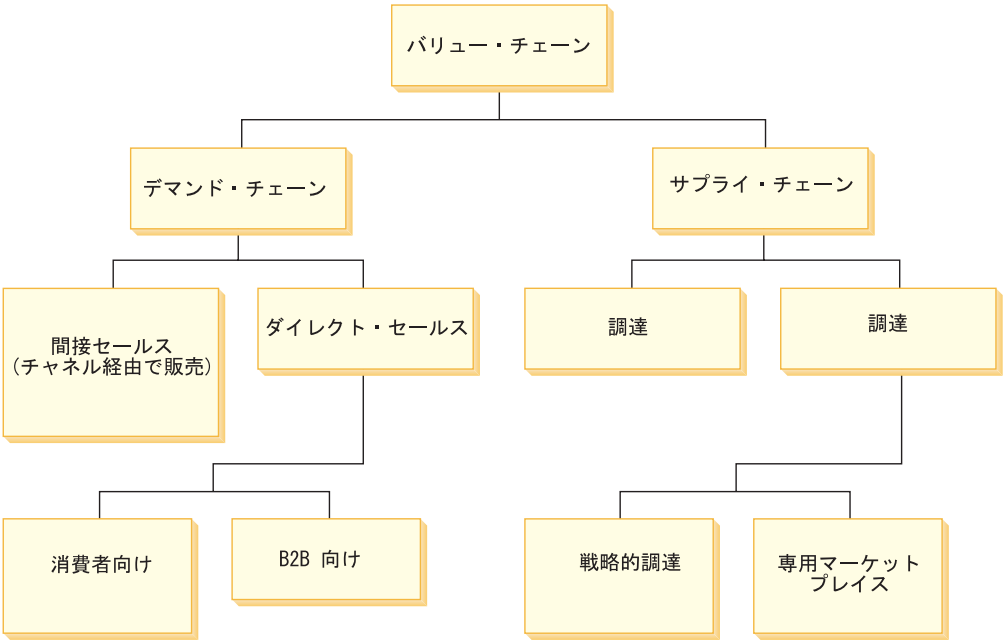
パートナーまたはサプライヤーとの関係を管理するために、バリュー・チェーン・ビジネス・モデルには、普通はハブ (WebSphere Commerce ではハブ・ストア) が組み込まれています。バリュー・チェーンの管理者は、ハブ・ストア中のバリュー・チェーンの操作可能な局面を管理できます。その中には、パートナーやサプライヤーがバリュー・チェーンに参加できるようにすること、つまりパートナーやサプライヤーを登録し、セットアップして、コラボレーションを作成することが含まれます。パートナーやサプライヤーは、ハブ・ストアにアクセスして、ユーザー登録などの管理用タスクをすべて行うこともできます。

顧客に直接に販売する (ダイレクト・セールス) ために、バリュー・チェーンには普通ストアフロントが組み込まれています。顧客は商品やサービスを直接ストアフロントで購入できます。

WebSphere Commerce は以下の 2 つのタイプのバリュー・チェーンについて、それを介した取り引き、およびその関係管理をサポートします。

- デマンド・チェーン
- サプライ・チェーン

以下の図は、バリュー・チェーンでサポートされるパートナーおよび関係についての概要を示しています。



デマンド・チェーン

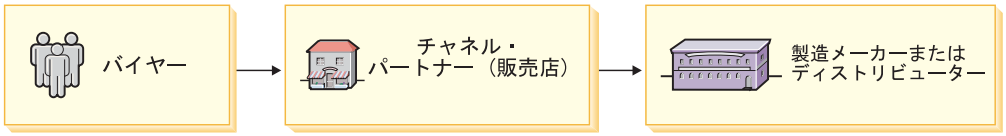
Business デマンド・チェーンは、ビジネスの商品またはサービスを販売する企業で構成されます。例として、あるデマンド・チェーンの例は、販売取引を開始するバイヤー、製造メーカーの商品を販売する販売店、および商品を製造する製造メーカーで構成されます。また、別のデマンド・チェーンは、製造メーカーの商品を販売する販売店、商品を製造する製造メーカー、および製造メーカーの商品を販売店に供給するディストリビューターから構成されます。デマンド・チェーンは、デマンド・チェーンの所有者が顧客やパートナーに直接販売するダイレクト・セールス・

チャネルもサポートします。ダイレクト・セールスの詳細については、 17 ページの『ダイレクト・セールス』を参照してください。

デマンド・チェーンのホスティング: デマンド・チェーンの所有者は、販売店やディストリビューターなどのチャネル・パートナー向けのストアをホスティングすることができます。

以下の図は、WebSphere Commerce によってサポートされるいくつかのデマンド・チェーンを示しています。

バイヤー、チャネル・パートナー (販売店)、および製造メーカー:



この例では、バイヤーは製造メーカーの販売店 (チャネル・パートナー) から商品を購入します。さらに販売店は、製造メーカーのハブを介して、製造メーカーから商品を入手します。

注: 販売店は、製造メーカーによってホスティングすることも、リモートにしておくこともできます。

販売店、製造メーカー、およびディストリビューター:

この例では、製造メーカーは販売店を含むチャネル・パートナーのためにハブを提供します。販売店および他のチャネル・パートナーは、製造メーカーの商品のディストリビューターを見つけることなどを含め、このハブでいくつかの機能を実行できます。

サプライヤーを見つけるために、販売店は専用ハブで商品カタログを参照することができます。目的の商品が複数のディストリビューターから購入可能である場合、販売店はさまざまなディストリビューターの商品の納期、ディストリビューターの場所、および価格を調べることができます。その後、販売店は複数のディストリビューターにオーダーを分けて出すこともできます。次いでオーダーはディストリビューターに送られ、ディストリビューターは取り引きを完了して商品またはサービスを販売店に引き渡します。次に販売店は、商品またはサービスをコンシューマーに直接販売します。

デマンド・チェーンのサンプル・サイトである Commerce プラザは、この販売店、製造メーカー、およびディストリビューターのシナリオのサンプルです。

注: 販売店は、製造メーカーによってホスティングすることも、リモートにしておくこともできます。

その他のシナリオ: このセクションで説明しているのは、デマンド・チェーンの 2、3 の例にすぎません。シナリオの詳細は、実践されるビジネスのタイプに応じて変わります。たとえば企業が製造メーカーであれば、ハブの目的は、製造メーカー

の販売店が、いくつかのディストリビューターから製造メーカーの商品を見つけることの支援になります。企業かディストリビューターであれば、ハブの目的は、ディストリビューターの販売店が、いくつかの異なるサプライヤーから商品またはサービスを検索することの支援になります。

サプライ・チェーン

Business

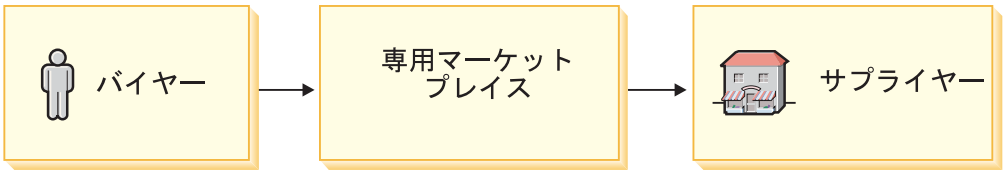
サプライ・チェーンは、ビジネスにサービスを提供する複数の企業で構成されます。 WebSphere Commerce には、プライベート・マーケットプレイスの形式でサプライ・チェーンをサポートする、構造的なインフラストラクチャーを備えています。

プライベート・マーケットプレイスは、商品を販売するためのフォーラムを取引先に提供します。バイヤーはこのフォーラムに入って、選択可能なオプションに目を通した後に、適切な商品またはサービスを選択します。

注: プライベート・マーケットプレイスは、競争入札や秘密入札または他の方法による競争をサポートしません。

サプライ・チェーンのホスティング: サプライ・チェーンの所有者は、そのサプライヤー向けのストアをホスティングすることができます。

以下の図は、サプライヤーのビジネスの例を示しています。



この例のサプライ・チェーンで、バイヤーはサプライヤーのハブに入って対話し、複数のサプライヤーからの商品とオファーが示された集合カタログをブラウズします。それからバイヤーは、複数のサプライヤーからご希望のオファーを選択したり見積を依頼したりできます。またバイヤーには、ビジネスを実践したり、直接オンライン・サプライヤーから調達したりするオプションもあります。

WebSphere Commerce でのサンプル・ストア

WebSphere Commerce には、いくつかのサンプル・ストアが備えられており、この章にリストされているさまざまなビジネス・モデルを WebSphere Commerce がどのようにサポートするかを学ぶことができます。使用できるサンプル (および対応するストア・アーカイブ・ファイル) は以下のとおりです。

表 2.

消費者向け	Business **B2B 向け**	Business **ホスティング**	Business **デマンド・チェーン**	Business **サプライ・チェーン**
FashionFlow (ConsumerDirect.sar) Express Express Store (ExpressStore.sar)	ToolTech (B2BDirect.sar)	**注:** 以下にリストされているすべてのストアは、複合ストア・アーカイブ Hosting.sar 中でも使用できます。Hosting.sar を発行して、ホスティングのサンプル全体を参照することをお勧めします。	**注:** 以下にリストされているすべてのストアは、複合ストア・アーカイブ DemandChain.sar 中でも使用できます。DemandChain.sar を発行して、デマンド・チェーンのサンプル全体を参照することをお勧めします。	**注:** 以下にリストされているすべてのストアは、複合ストア・アーカイブ SupplyChain.sar 中でも使用できます。SupplyChain.sar を発行して、サプライ・チェーンのサンプル全体を参照することをお勧めします。
		Commerce ホスティング・ハブ (Hosting Hub.sar)	Commerce プラザ (ChannelHub.sar)	Commerce サプライヤー・ハブ (SupplierHub.sar)
		ストア・ディレクトリー (Store Directory.sar)	カタログ資産ストア (CatalogAsset Store.sar)	カタログ資産ストア (Catalog AssetStore.sar)
		カタログ資産ストア (CatalogAsset Store.sar)	販売店ストアフロント資産ストア (Resellerstore frontAsset Store.sar)	サプライヤー資産ストア (Supplier AssetStore.sar)
		ホスティング・ストアフロント資産ストア (HostedStore FrontAsset Store.sar)	ディストリビューター資産ストア (DistributorAsset Store.sar)	サプライヤー
		ホストされるストア	ホストされる販売店ストア	
			ディストリビューター・ストア	

これらのサンプル中のストアのタイプについて詳しくは、 74 ページの『ストア・アーキテクチャーがビジネス・モデルをサポートする方法について』を参照してください。サンプル・ストアの詳細については、「*WebSphere Commerce サンプル・ストア・ガイド*」を参照してください。

注: 個々のサンプルには、ビジネス・モデルの組織構造があるコンポーネント・ストア・アーカイブも含まれています。

これらのサンプルは各ビジネス・モデル中のストアの特定のインスタンスを表現したものであり、そのビジネス・モデルで使用できる、すべてのバリエーションを示してはいないことに注意してください。ただし、特定の場合のビジネスがここに示したサンプルとかなり異なっていても、これらのサンプルを開始点として、サイトを構築したり、サンプルの一部を使用してサイトを構築できる場合があります。WebSphere Commerce に備えられているサンプルの詳細については、「*WebSphere Commerce サンプル・ストア・ガイド*」を参照してください。

第 3 部 WebSphere Commerce アーキテクチャー

この部分では、WebSphere Commerce アーキテクチャーがビジネスのオンライン化をどのようにサポートしているのかについて、その概要を扱います。特に、WebSphere Commerce アーキテクチャーのコンポーネントによって、ビジネスのさまざまなパーティー (たとえば、顧客、ビジネス・パートナー、ディストリビューター、販売店、またはサプライヤー) とのオンラインでの対話がどのように可能になっているのかについて説明します。

ビジネスで関わるさまざまなパーティー (たとえば、顧客、ビジネス・パートナー、取引先、サプライヤー、メーカー、ディストリビューター、および管理者) とオンラインでビジネス面のやり取りをしたり相互に対話できるようにするために、WebSphere Commerce には次のアーキテクチャー・コンポーネントが組み込まれています。

- 組織構造
- アクセス制御モデル
- ビジネス・ポリシー・フレームワーク
- インスタンス・アーキテクチャー
- ストア・アーキテクチャー

これらのコンポーネントが組み合わされてアーキテクチャーが作成され、このアーキテクチャーによってさまざまなビジネス・パートナーと相互に対話することができます。

第 3 章 WebSphere Commerce 組織構造

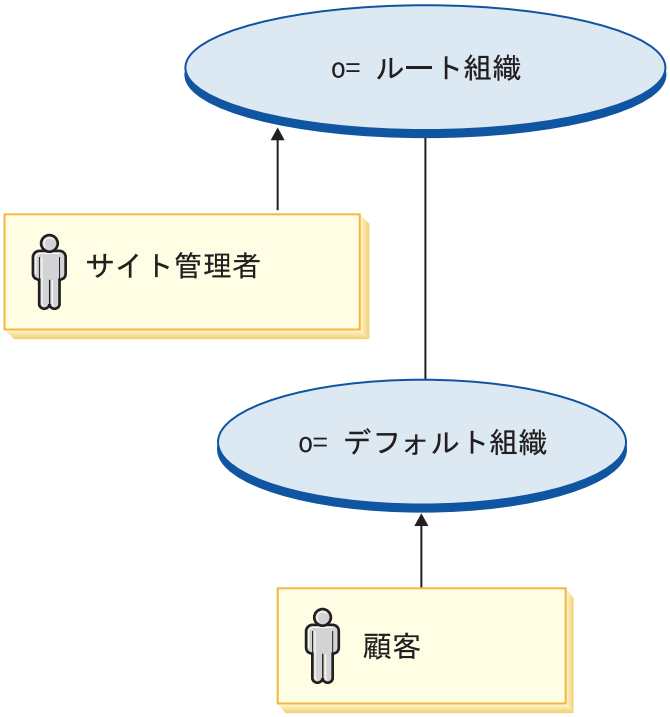
顧客またはバイヤーがサイトへのアクセス、カタログのブラウズ、およびオーダーの発行を行えるようにするため、従業員がカタログの更新、新しい販売促進の作成、またはオーダーの管理を含むサイトの管理を行えるようにするため、または販売店その他のビジネス・パートナーがサイトでの取引を完了できるようにするために、ビジネス・シナリオでのすべての役者に WebSphere Commerce 組織構造での役割を割り当てる必要があります。

WebSphere Commerce 組織構造について

WebSphere Commerce 組織構造は、ビジネス・シナリオでの役者つまりエンティティーのためのフレームワークを提供します。このフレームワークは、組織、組織単位、およびユーザーのエントリーを持つ典型的な組織階層に類似した、階層構造で編成されています。フレームワーク内の組織および組織単位は、ビジネスの各部分の所有者となります。顧客、管理者、ストア、カタログ、およびディストリビューターを含むビジネスのすべての部分は、組織または組織単位によって所有される必要があります。

組織構造とアクセス制御モデル (39 ページの『第 4 章 WebSphere Commerce でのアクセス制御』を参照) は密接に関連しており、アクセス制御モデルは個々のエンティティー (ストア、顧客、管理者など) ではなく組織にアクセス制御ポリシーを適用します。エンティティー (またはリソース) に適用されるポリシーは、そのエンティティーまたはリソースを所有する組織に適用されます。

以下の図は、基本的な WebSphere Commerce 組織構造を示しています。基本的な組織構造は、ビジネス・モデルには関係なく、インスタンスの作成時にインストールされます。



- **ルート組織**: ルート組織は最上位の組織であり、それ自身の親です。 WebSphere Commerce 組織構造内のすべての組織は、ルート組織の下層となります。サイト管理者は、ルート組織によって所有されます。
- **デフォルト組織**: デフォルト組織は、ルート組織によって所有されます。すべてのゲスト顧客および消費者向けシナリオ内のすべての顧客は、デフォルト組織に属します。 B2B 向けおよびバリュー・チェーンのシナリオ内の顧客は、デフォルト組織またはその他の組織のいずれかに属することができます。

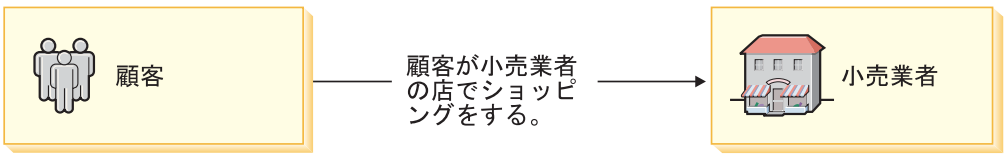
組織エンティティーの 1 つ以上のレベルが、親の組織エンティティーの下に存在することがあります。ビジネスをサポートするために必要な数の子の組織エンティティーを追加できます。

組織構造がビジネス・モデルをサポートする方法

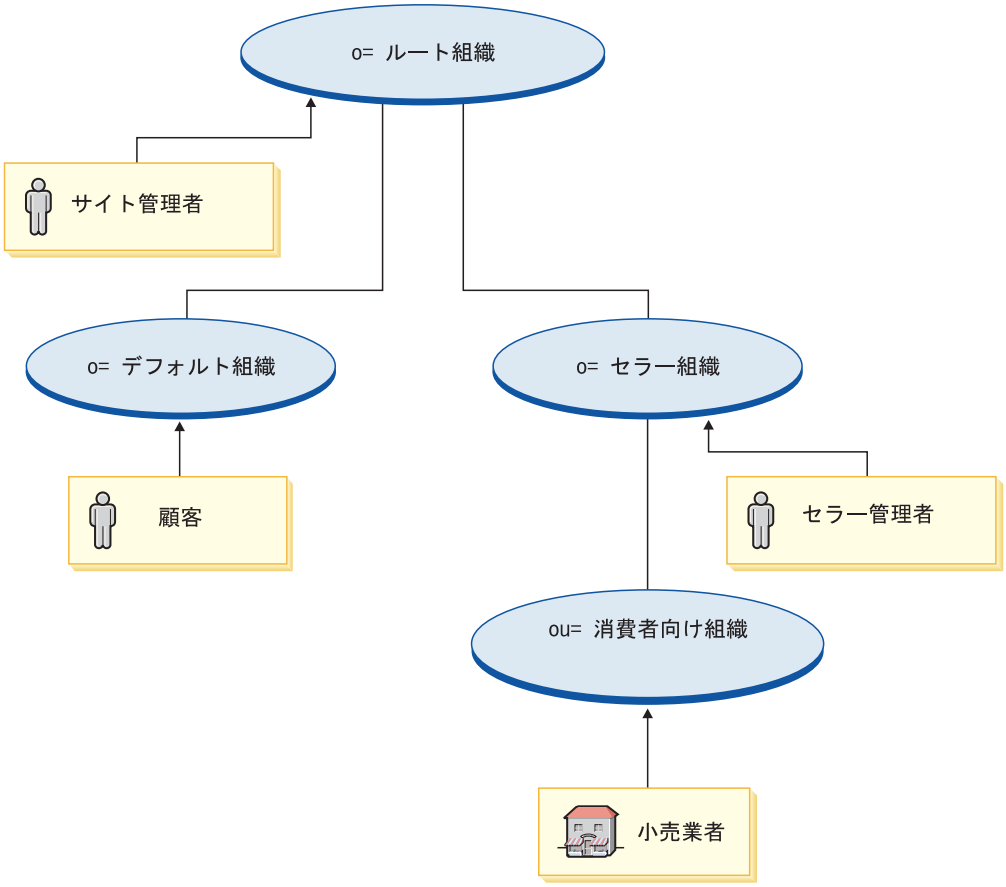
WebSphere Commerce 組織構造は、サポートされるビジネス・モデルのすべてのエンティティーをサポートできる十分な柔軟性を備えています。次のセクションの図は、各ビジネス・モデルの典型的な例が、WebSphere Commerce 組織構造にマップされる方法を示しています。

消費者向け

以下の図は、典型的な消費者向けビジネスを示しています。

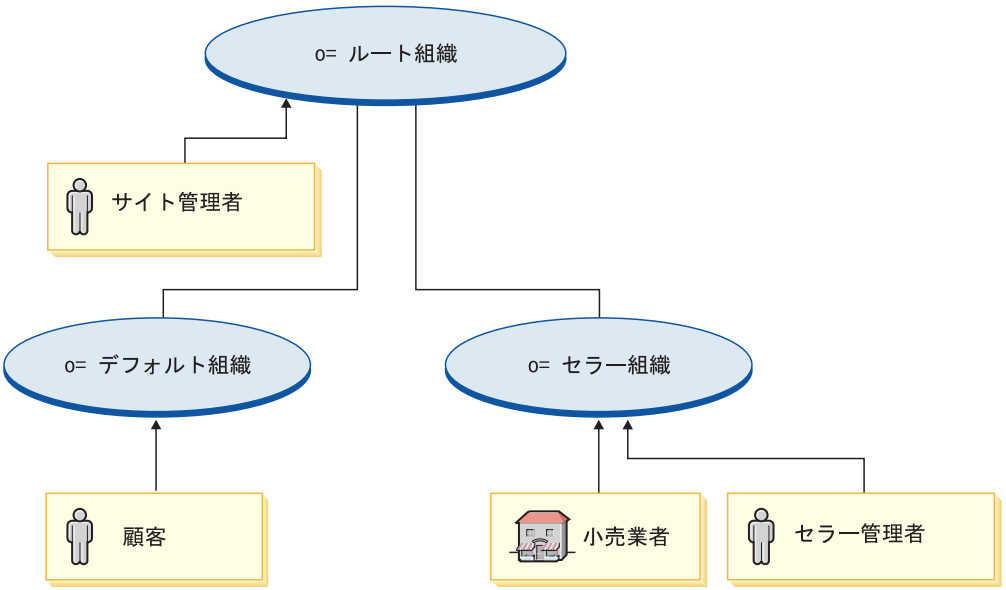


WebSphere Commerce を使用してこのビジネスをオンラインで実施するために、前述の図に含まれるエンティティーを以下の組織に割り当てる必要があります。



- **ルート組織**: ビジネス内のすべての組織は、ルート組織の下層となります。オンライン・サイトを保守するサイト管理者は、ルートによって所有されます。
  - **デフォルト組織**: ビジネスの顧客はすべて、デフォルト組織によって所有されます。
  - **セラー組織**: すべてのセラー組織 (ストアおよびストアを保守するための管理者を含む) を所有するためのセラー組織が作成されます。ストアの機能を保守する管理者 (顧客サービス担当者やカタログ・マネージャーおよびプロダクト・マネージャーなど) は、セラー管理者と呼ばれ、セラー組織によって直接所有されます。
    - 子の組織単位 (ou) である消費者向け組織は、セラー組織の下に作成されて、ストア (小売業者) を所有します。

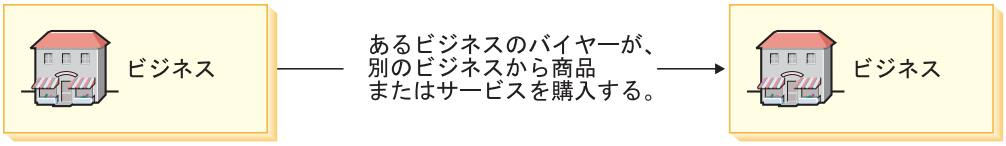
Express WebSphere Commerce - Express の組織構造は、前述の消費者向け組織とは少し異なります。 WebSphere Commerce - Express を使用してこのビジネスをオンラインで実施するために、前述の消費者向けの図に含まれるエンティティーを以下の組織に割り当てる必要があります。



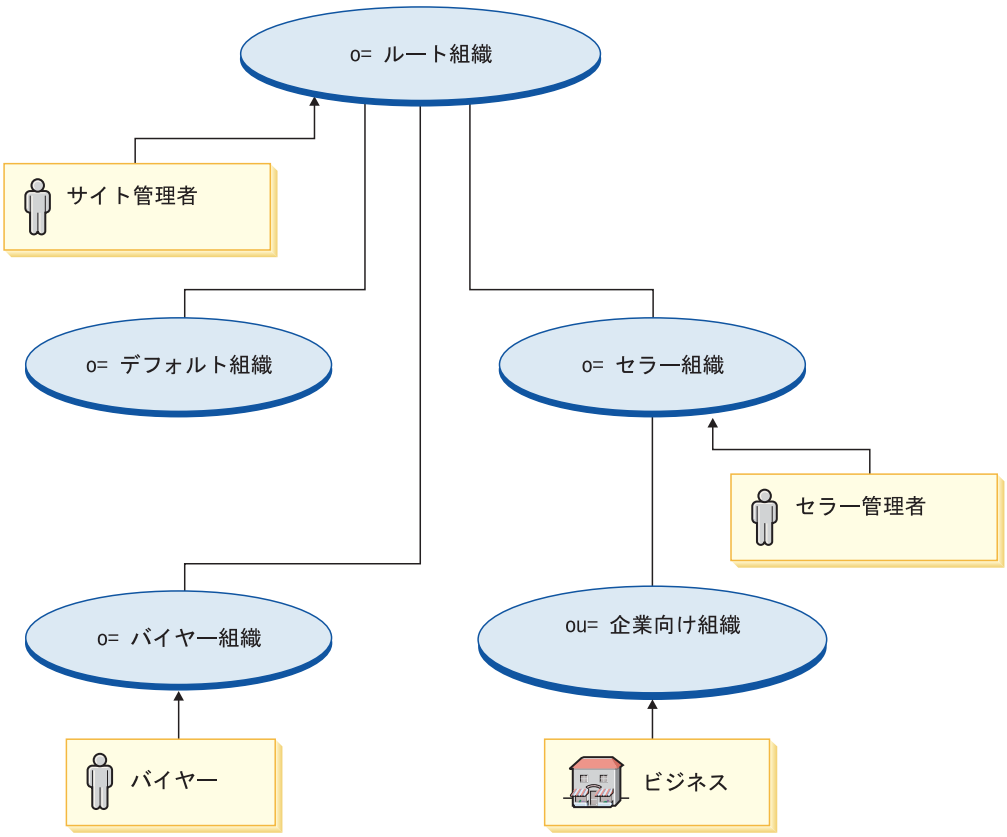
- **ルート組織**: ビジネス内のすべての組織は、ルート組織の下層となります。オンライン・サイトを保守するサイト管理者は、ルートによって所有されます。
  - **デフォルト組織**: ビジネスの顧客はすべて、デフォルト組織によって所有されます。
  - **セラー組織**: すべてのストア (小売業者) およびそのストアを保守する管理者を所有するために作成されます。ストアの機能を保守する管理者 (顧客サービス担当者やカタログ・マネージャーおよびプロダクト・マネージャーなど) は、セラー管理者と呼ばれ、セラー組織によって直接所有されます。

B2B 向け

Business 以下の図は、典型的な B2B 向けビジネスを示しています。



このビジネスをオンラインで実施するために、前述の図に含まれるエンティティーを以下の組織に割り当てる必要があります。

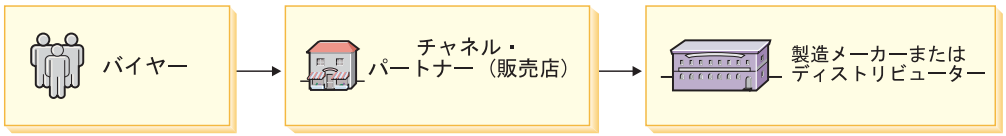


- **ルート組織**: ビジネス内のすべての組織は、ルート組織の下層となります。オンライン・サイトを保守するサイト管理者は、ルートによって所有されます。
  - **デフォルト組織**: 消費者向け編成構造とは異なり、顧客はデフォルト組織によって所有されません。代わりに、顧客はバイヤー組織によって所有されるバイヤーになります。
  - **バイヤー組織**: B2B 向けビジネスでバイヤーとして知られる顧客は、 B2B 向け組織構造内に独自の組織が割り当てられます。
  - **セラー組織**: ストアを所有するすべての組織を所有するためのセラー組織が作成されます。ストアの機能を保守する管理者 (顧客サービス担当者やカタログ・マネージャーおよびプロダクト・マネージャーなど) は、セラー管理者と呼ばれ、セラー組織によって直接所有されます。
    - 子の組織単位 (ou) である B2B 向け組織は、セラー組織の下に作成されて、ストア (ビジネス) を所有します。

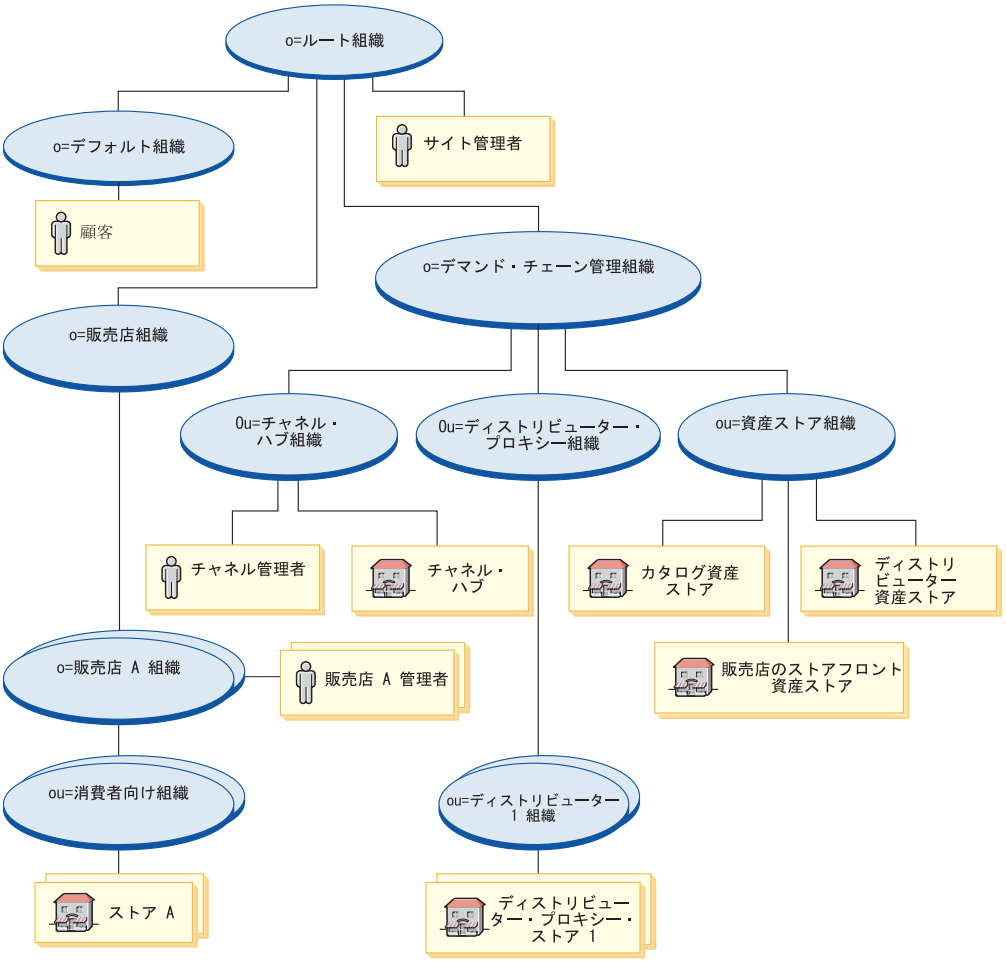
デマンド・チェーン

Business

以下の図は、デマンド・チェーンのビジネスの例を示しています。



このビジネスをオンラインで実施するために、前述の図に含まれるエンティティーを以下の組織に割り当てる必要があります。



- **ルート組織**: ビジネス内のすべての組織は、ルート組織の下層となります。さらに、オンライン・サイトを保守する管理者であるサイト管理者は、ルートの下に直接追加されます。
  - **デフォルト組織**: デフォルトでは、デフォルト組織の下には何も存在しません。販売店ストアの顧客がこの組織の下に入れられる場合があります。
  - **デマンド・チェーン・マネージメント組織**: デマンド・チェーン・マネージメント組織は、チャネル関連の組織すべて (販売店を所有する組織を除く) を所有するために作成されます。デマンド・チェーン・マネージメント組織は、以下の子の組織単位を所有します。
    - **チャネル・ハブ組織**: チャネル・ハブ組織は、チャネル・ハブを所有するために作成されます。チャネル・ハブ機能の保守および販売店組織の管理を行う管理者は、チャネル管理者と呼ばれて、チャネル・ハブ組織によって直接所有されます。

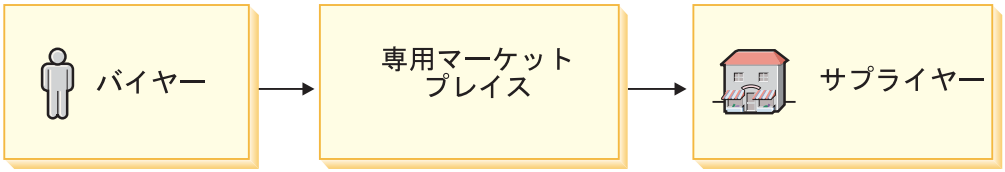
- **ディストリビューター・プロキシー組織**: ディストリビューター・プロキシー組織は、ディストリビューターへのすべての接続を所有するために作成されます。組織内のディストリビューター・プロキシーごとに、子の組織単位が作成されます。
  - **ディストリビューター組織**: サイト内のディストリビューター・プロキシーごとに、新しいディストリビューター組織が作成されます。
- **資産ストア組織**: 資産ストア組織は、チャネル・パートナー (販売店とディストリビューター) のためのストアの作成に使用されるすべての資産を所有するために作成されます。

– **販売店組織**: 販売店組織は、デマンド・チェーン内のすべての販売店を所有するために作成されます。販売店ごとに子組織が作成されます。
  - **販売店組織 A、B、C**: 販売店ストアごとに、親の販売店組織の下に新しい販売店組織が作成されます。ストアの機能を保守する管理者 (顧客サービス担当者やカタログ・マネージャーおよびプロダクト・マネージャーなど) は、販売店管理者と呼ばれ、対応する販売店組織によって直接所有されます。

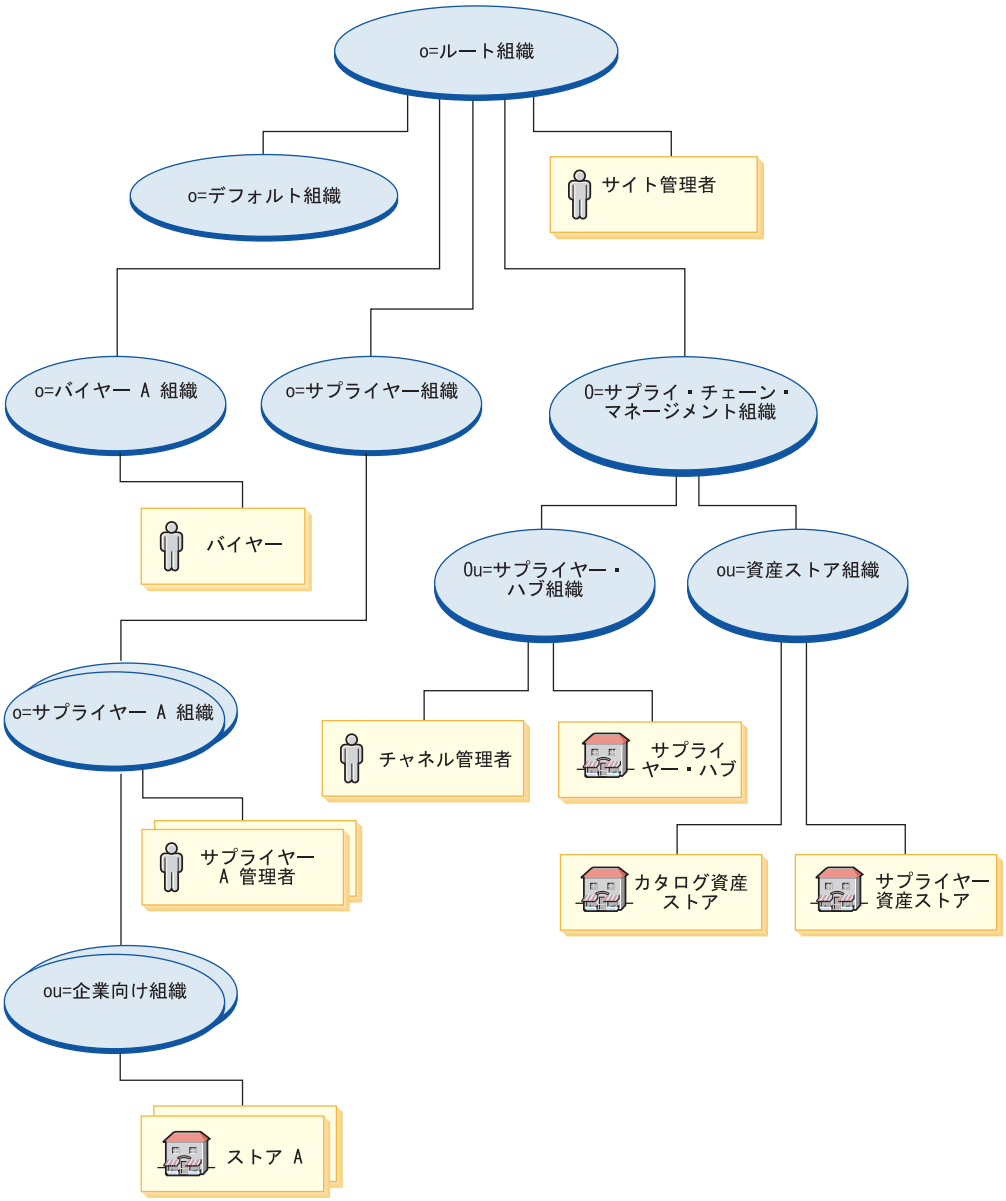
サプライ・チェーン

Business

以下の図は、典型的なサプライ・チェーン・ビジネスを示しています。



このビジネスをオンラインで実施するために、前述の図に含まれるエンティティーを以下の組織に割り当てる必要があります。



- **ルート組織**: ビジネス内のすべての組織は、ルート組織の下層となります。さらに、オンライン・サイトを保守する管理者であるサイト管理者は、ルートの下に直接追加されます。
  - **デフォルト組織**: デフォルトでは、デフォルト組織の下には何も存在しません。
  - **サプライ・チェーン・マネージメント組織**: サプライ・チェーン・マネージメント組織は、サプライ・チェーン関連の組織すべて (サプライヤーを所有する組織を除く) を所有するために作成されます。サプライ・チェーン・マネージメント組織は、以下の子の組織単位を所有します。
    - **サプライヤー・ハブ組織**: サプライヤー・ハブ組織は、サプライヤー・ハブを所有するために作成されます。サプライヤー・ハブ機能の保守およびサプ

ライヤー組織の管理を行う管理者は、チャネル管理者と呼ばれて、サプライヤー・ハブ組織によって直接所有されます。

- **資産ストア組織**: 資産ストア組織は、サプライヤーのためのストアの作成に使用されるすべての資産を所有するために作成されます。

– **サプライヤー組織**: サプライヤー組織は、チェーン内のすべてのサプライヤーを所有するために作成されます。サプライヤーごとに子組織が作成されます。

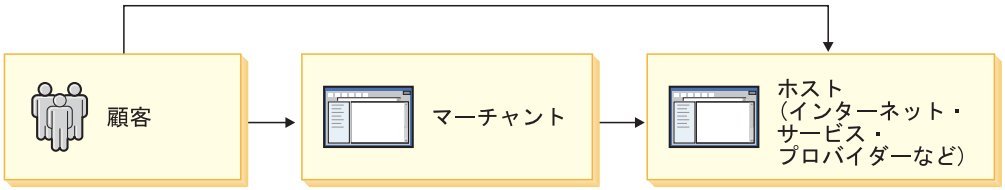
- **サプライヤー組織 A、B、C**: サプライヤー・ストアごとに、親のサプライヤー組織の下に新しいサプライヤー組織が作成されます。ストアの機能を保守する管理者は、サプライヤー管理者と呼ばれ、対応するサプライヤー組織によって直接所有されます。

– **バイヤー組織**: バイヤーは、ルートの下に独自の組織を持っています。すべてのバイヤーは対応するバイヤー組織によって所有されます。

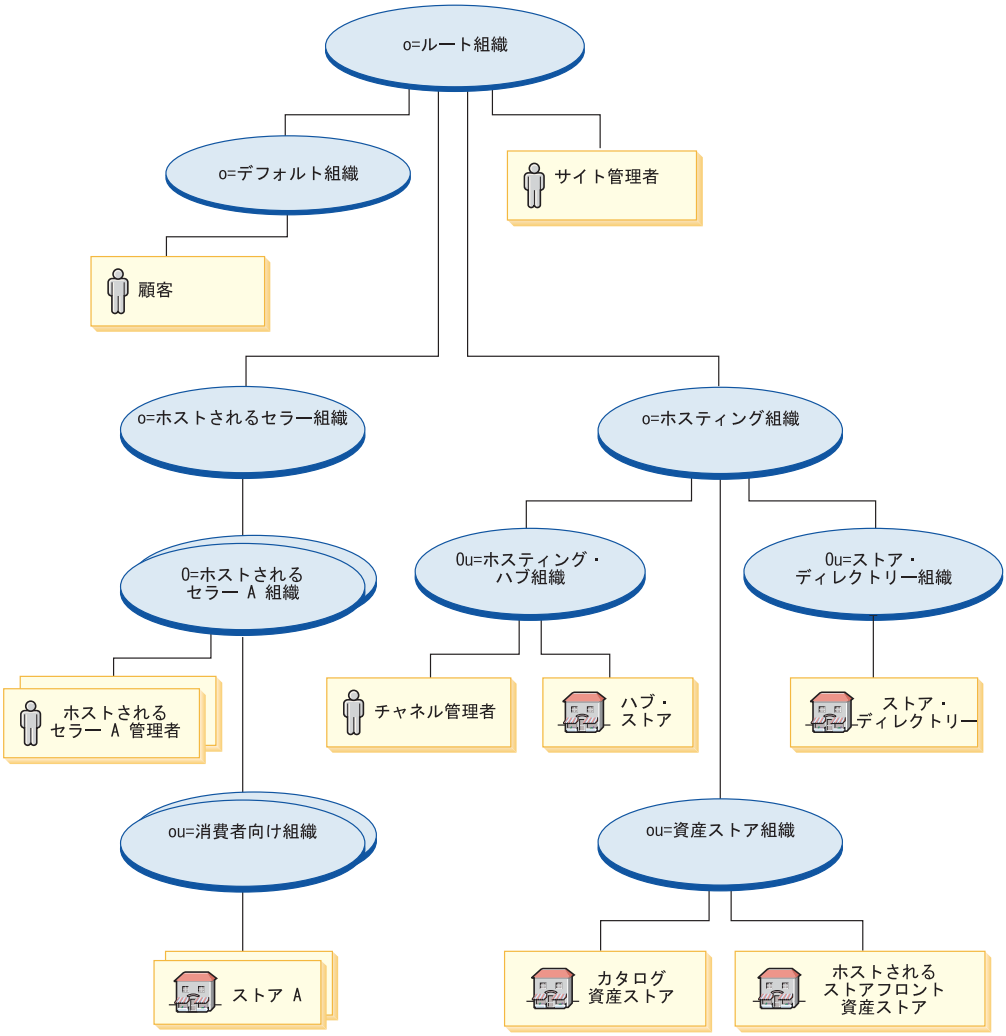
ホスティング

Business

以下の図は、典型的なホスティング・ビジネスを示しています。



このビジネスをオンラインで実施するために、前述の図に含まれるエンティティーを以下の組織に割り当てる必要があります。



- **ルート組織**: ビジネス内のすべての組織は、ルート組織の下層となります。さらに、オンライン・サイトを保守する管理者であるサイト管理者は、ルートの下に直接追加されます。
  - **デフォルト組織**: ビジネスの顧客はすべて、デフォルト組織によって所有されます。
  - **ホスティング組織**: ホスティング組織は、ホスティング関連の組織すべて (ホストされるストアを所有する組織を除く) を所有するために作成されます。ホスティング組織は、以下の子の組織単位を所有します。
    - **ホスティング・ハブ組織**: ホスティング・ハブ組織は、ホスティング・ハブを所有するために作成されます。ホスティング・ハブの機能の保守およびホスティング組織の管理を行う管理者は、チャネル管理者と呼ばれて、ホスティング・ハブ組織によって直接所有されます。
    - **ストア・ディレクトリー組織**: ストア・ディレクトリー組織は、ストア・ディレクトリーを所有するために作成されます。
    - **資産ストア組織**: 資産ストア組織は、ホストされるストアの作成に使用されるすべての資産を所有するために作成されます。

– **ホストされるセラー組織**: ホストされるセラー組織は、ホストされるストアを所有するために作成されます。ホストされるストアごとに、子の組織単位が作成されます。
  - **ホストされるストア組織 A、B、C**: ホストされるストアごとに、親のホスティング組織の下にホストされるストア組織が新しく作成されます。ストアの機能を保守する管理者は、ホストされるセラーの管理者と呼ばれ、対応するホストされるストア組織によって直接所有されます。

組織構造のサンプル

WebSphere Commerce では、サポートされるビジネス・モデルごとに組織構造のサンプルが提供されています。これらの組織構造のサンプルは (コンポーネント・ストア・アーカイブとして) それ自体で使用可能なので、組織構造のサンプルを独自のサイトの開始点として、またはサンプル・ビジネスの一部として使用できます。組織構造のサンプルの詳細については、「*WebSphere Commerce サンプル・ストア・ガイド*」を参照してください。

組織構造の作成

サイト用の新しい組織構造を作成する代わりに、 WebSphere Commerce で提供されている組織構造のサンプルの 1 つを発行することから始めて、その後、必要に応じてその組織構造を変更することをお勧めします。組織データの編集の詳細については、 133 ページの『WebSphere Commerce のメンバー資産について』を参照してください。

第 4 章 WebSphere Commerce でのアクセス制御

WebSphere Commerce では、顧客、バイヤー、管理者、ディストリビューター、メーカー、またはサプライヤーであれ、特定のユーザーがビジネスに関連して実行できるタスクをアクセス制御によって決定できます。

WebSphere Commerce のアクセス制御モデルについては、「*WebSphere Commerce セキュリティー・ガイド*」で詳しく説明されています。ただし、本書でも、アクセス制御がサイトおよびストア開発にどのように影響するかについての概要を示しています。

WebSphere Commerce のアクセス制御について

WebSphere Commerce のアクセス制御は、以下のエレメントで構成されています。すなわち、ユーザー、アクション、リソース、および関係です。

- ユーザーは、システムを使用する人のことです。アクセス制御を行うには、ユーザーを関係のあるアクセス・グループに分類しなければなりません。アクセス・グループのメンバーシップを決定するために使用される 1 つの共通属性は、役割です。役割は、組織単位のベースでユーザーに割り当てられます。役割の詳細については、135 ページの『役割』を参照してくださいアクセス・グループの例としては、登録済み顧客、ゲスト顧客、または顧客サービス担当者のような管理者グループなどがあります。
- アクションは、ユーザーがリソースに対して実行できる活動です。アクセス制御のためには、アクションも関係のあるアクション・グループに分類しなければなりません。たとえば、ストアで使用される一般的なアクションの 1 つに表示があります。表示は、ストア・ページを顧客に表示するときに行われます。ストアで使用する表示は、使用する前にアクションとして宣言して特定のアクション・グループに割り当てなければなりません。
- リソースは、保護されているエンティティーです。たとえば、アクションが表示の場合、保護されるリソースはその表示を呼び出すコマンド (com.ibm.commerce.command.ViewCommand など) になります。アクセス制御のために、リソースはリソース・グループにグループ化されます。
- 関係は、ユーザーとリソースの間の関係です。アクセス制御ポリシーでは、ユーザーとリソースの間の関係を満たさなければならない場合があります。たとえば、ユーザーは、作成したオーダーだけの表示しか許可されない場合があります。

アクセス制御ポリシー

アクセス制御ポリシーは、アクセス・グループのユーザーがリソースに関する特定の関係を満たしている限り、そのアクセス・グループが WebSphere Commerce のリソースに対して特定のアクションを実行することを許可します。

WebSphere Commerce インスタンス作成時にロードされる、 300 以上のデフォルト・アクセス制御ポリシーが用意されています。これらのポリシーは、オーダーの

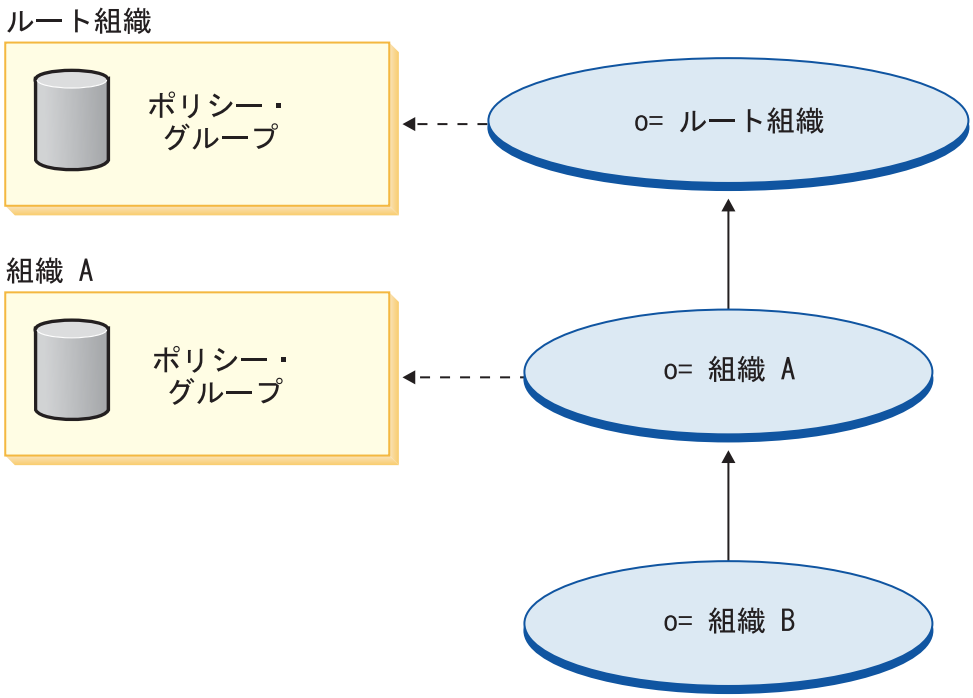
作成と処理や、Business 見積依頼および Business 契約などの商取引を含む、広い範囲の一般的なビジネス活動を網羅しています。デフォルト・ポリシーは、「*WebSphere Commerce セキュリティー・ガイド*」で説明されています。

アクセス制御ポリシー・グループ

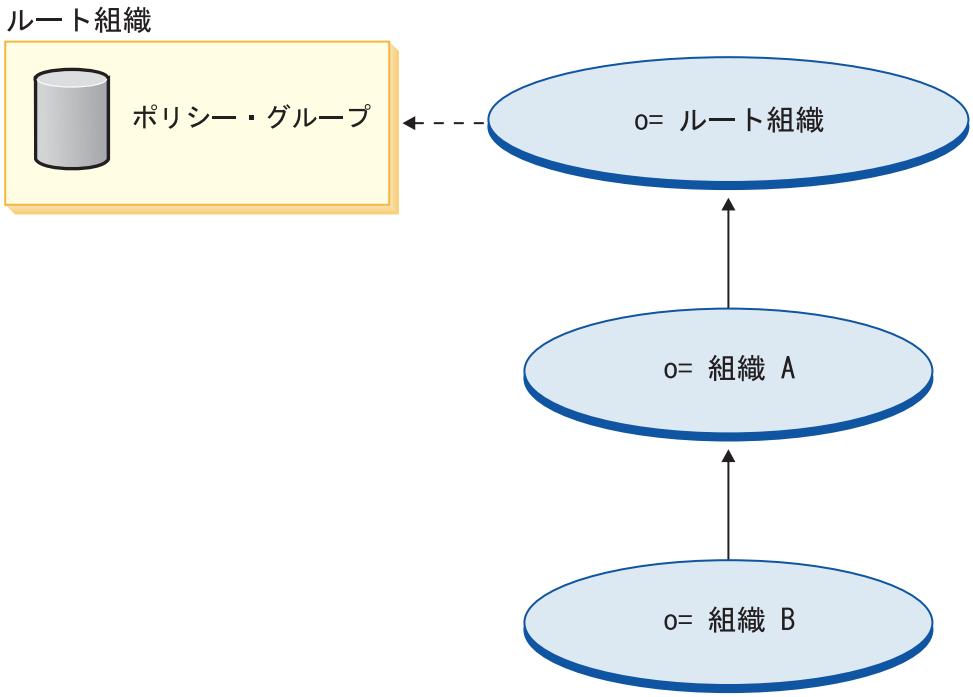
アクセス制御ポリシーをストアまたはサイトに適用するには、それがアクセス制御ポリシー・グループに属している必要があり、ポリシー・グループは、リソースを所有する組織によってサブスクライブされることが必要です。デフォルトでは、WebSphere Commerce で提供されるすべてのアクセス制御ポリシーはポリシー・グループに割り当てられます。 WebSphere Commerce が提供するデフォルト・ポリシーのリストについては、「*WebSphere Commerce セキュリティー・ガイド*」を参照してください。

アクセス制御ポリシー・グループは、組織によって所有されていても、自動的に組織に適用されるわけではありません。組織は、アクセス制御ポリシーを組織に適用するため、ポリシー・グループにサブスクライブすることが必要です。組織に子組織がある場合、親がサブスクライブするすべてのポリシー・グループは、自動的に子組織に適用されます。ただし、子組織がポリシー・グループに直接サブスクライブする場合、親組織がサブスクライブするポリシー・グループは子には適用されなくなります。

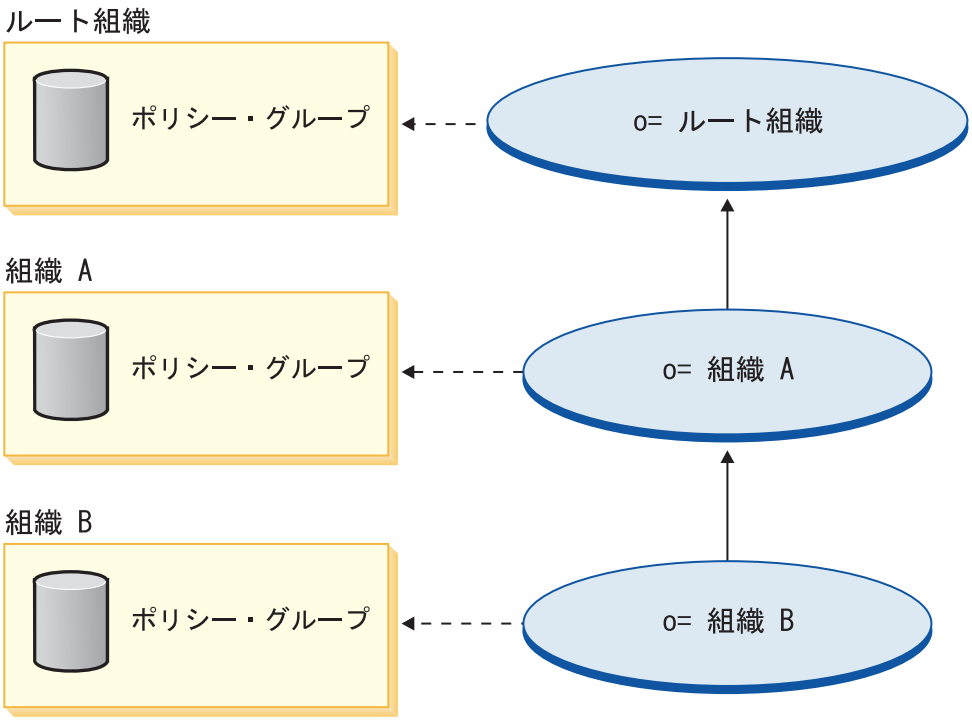
WebSphere Commerce の前のバージョンでは、ポリシーは、そのポリシーの所有者組織の子孫が所有するすべてのリソースに適用しました。たとえば、組織 A が特定のポリシーを持っており、組織 B の親である場合、組織 B は暗黙にそのポリシーを持つことになります。 WebSphere Commerce 5.5 では、組織はポリシー・グループをサブスクライブできます。 WebSphere Commerce 5.5 の場合、組織 B がどのポリシー・グループもサブスクライブしていない場合、アクセス制御フレームワークは、組織階層の検索を開始し、最低 1 つのポリシー・グループをサブスクライブする組織を検出するまでそれを続けます。組織 B の直接の親組織である組織 A がポリシー・グループをサブスクライブしていると、検索は停止し、組織 A のポリシー・グループ中のポリシーは組織 A と B に適用されます。これを次の図で示します。



組織 A がポリシー・グループをサブスクライブしていない場合、サブスクリプションを持つ組織に達するまで組織階層の検索は続行します。次の図では、ルート組織がポリシー・グループをサブスクライブするところを示します。組織 B と組織 A は、それらのグループ中のポリシーを継承します。



組織 B がポリシー・グループをサブスクライブすると、検索は組織 B で停止し、組織 B はサブスクライブ先のそれらのポリシーにのみ適用されます。これを次の図で示します。



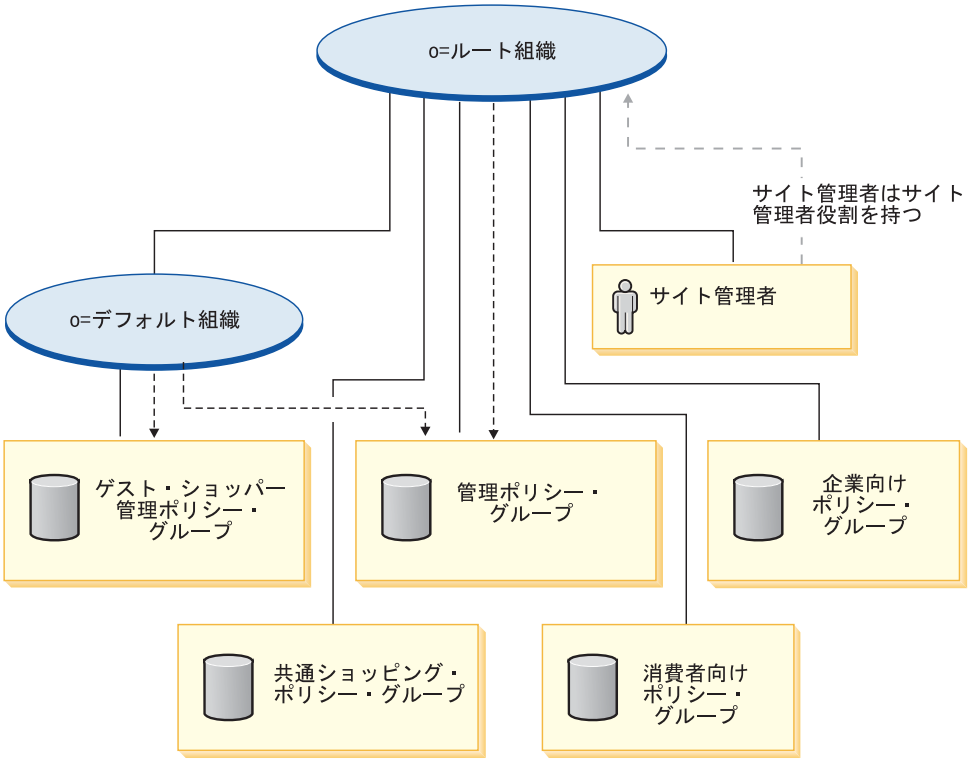
注: アクセス制御の面から言うと、リソースの所有権には特別な意味があります。すべてのリソースは、com.ibm.commerce.security.Protectable インターフェースをインプリメントする必要があります。このインターフェースのメソッドの 1 つは getOwner() で、これはリソースの所有者のメンバー ID を戻します。たとえば、Order Entity Bean は、リモート・インターフェースが Protectable インターフェースを拡張することで保護されるリソースです。 Order で getOwner() をインプリメントすると、それは特定の Order リソースが、 オーダーの発生したストアの所有者を戻すようなものとなります。リソースがコマンドであるポリシー (たとえば com.ibm.commerce.command.ViewCommand) なら、 getOwner() のデフォルトのインプリメントは、現在コマンドのコンテキストにあるストアの所有者を戻すことになります。コマンド・コンテキストにストアがなければ、ルートの組織が所有者として使用されます。詳細については、「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」を参照してください。

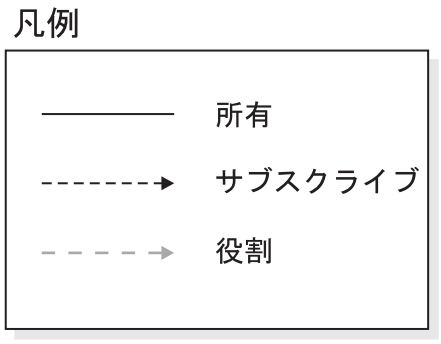
ビジネス・モデルのアクセス制御について

WebSphere Commerce アクセス制御構造は、サポートされるビジネス・モデルのすべてのエンティティーをサポートできる十分な柔軟性を備えています。次のセクションの図は、各ビジネス・モデルの典型的な例に、アクセス制御が適用される方法を示しています。

基本のアクセス制御構造

基本のアクセス制御構造は、ビジネス・モデルに関係なく、インスタンス作成中にインストールされます。





ルート組織は次のデフォルト・ポリシー・グループを所有します。

- 管理
- 共通ショッピング
- B2C
- B2B

ただし、ルート組織は管理ポリシー・グループしかサブスクライブしません。その結果、これらのポリシーは、ルートのすぐ下のサイト管理者に適用されます。

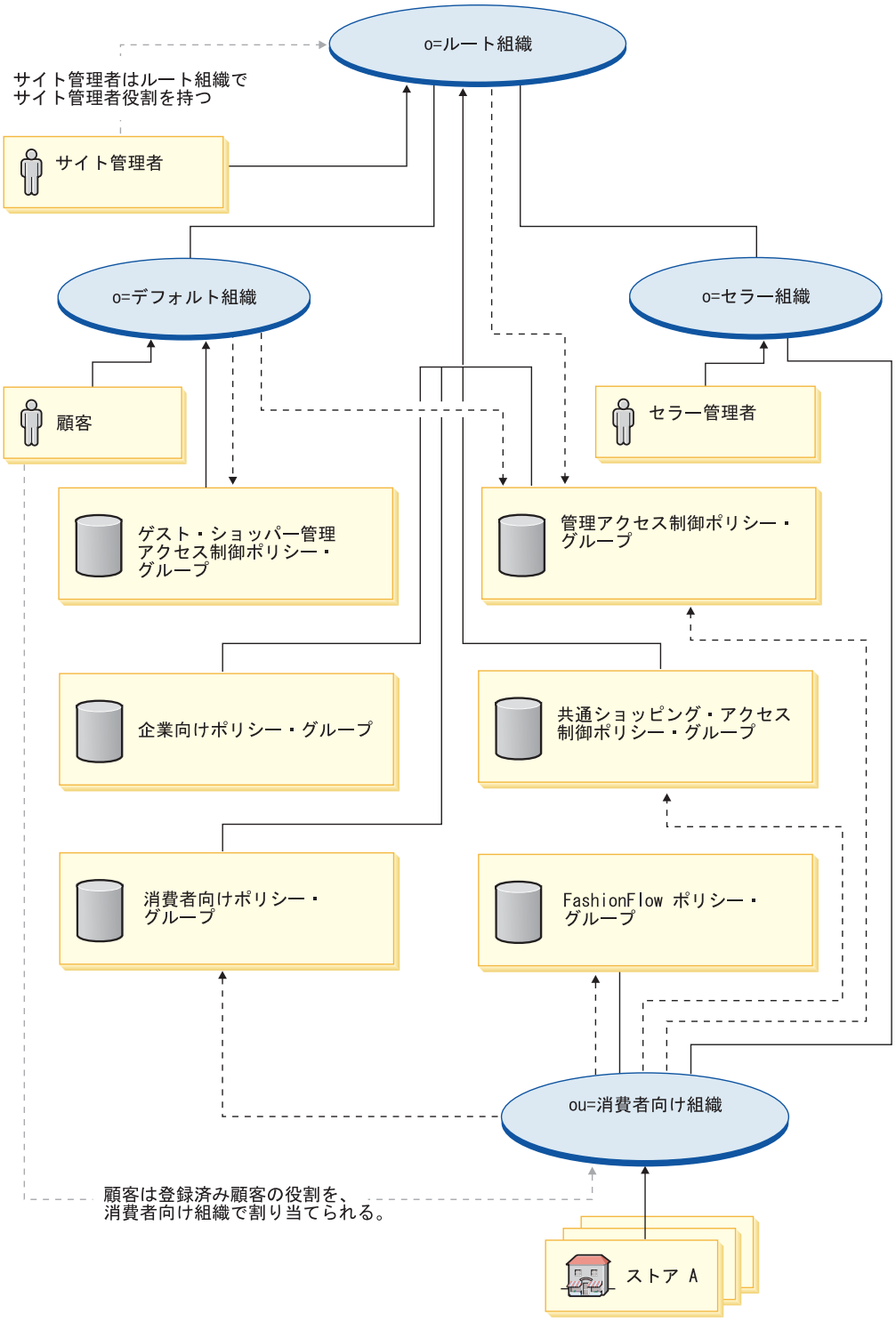
継承により管理ポリシー・グループ中のポリシーをデフォルト組織に適用することはできません。これは、デフォルト組織はゲスト・ショッパー管理のポリシー・グループをサブスクライブするためです。管理ポリシーを適用するには、デフォルト組織は管理ポリシー・グループを明示的にサブスクライブしなければなれません。

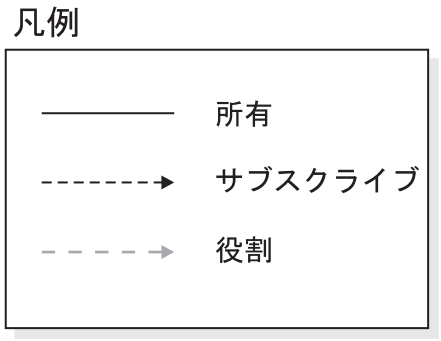
デフォルト組織はゲスト・ショッパー管理のポリシー・グループを所有します。

注: デフォルト・ポリシー・グループの詳細については、「*WebSphere Commerce セキュリティー・ガイド*」の付録を参照してください。

消費者向け

次の図は、基本的な消費者向け組織およびアクセス制御構造を説明しています。





この図では、基本的な消費者向け組織を説明しており、 43 ページの『基本のアクセス制御構造』で説明されているように、ルート組織がデフォルト・ポリシー・グループを所有してサブスクライブします。

消費者向け組織は、消費者向けアクセス制御ポリシー、管理ポリシー・グループ、および共通ショッピング・ポリシー・グループを直接サブスクライブします。

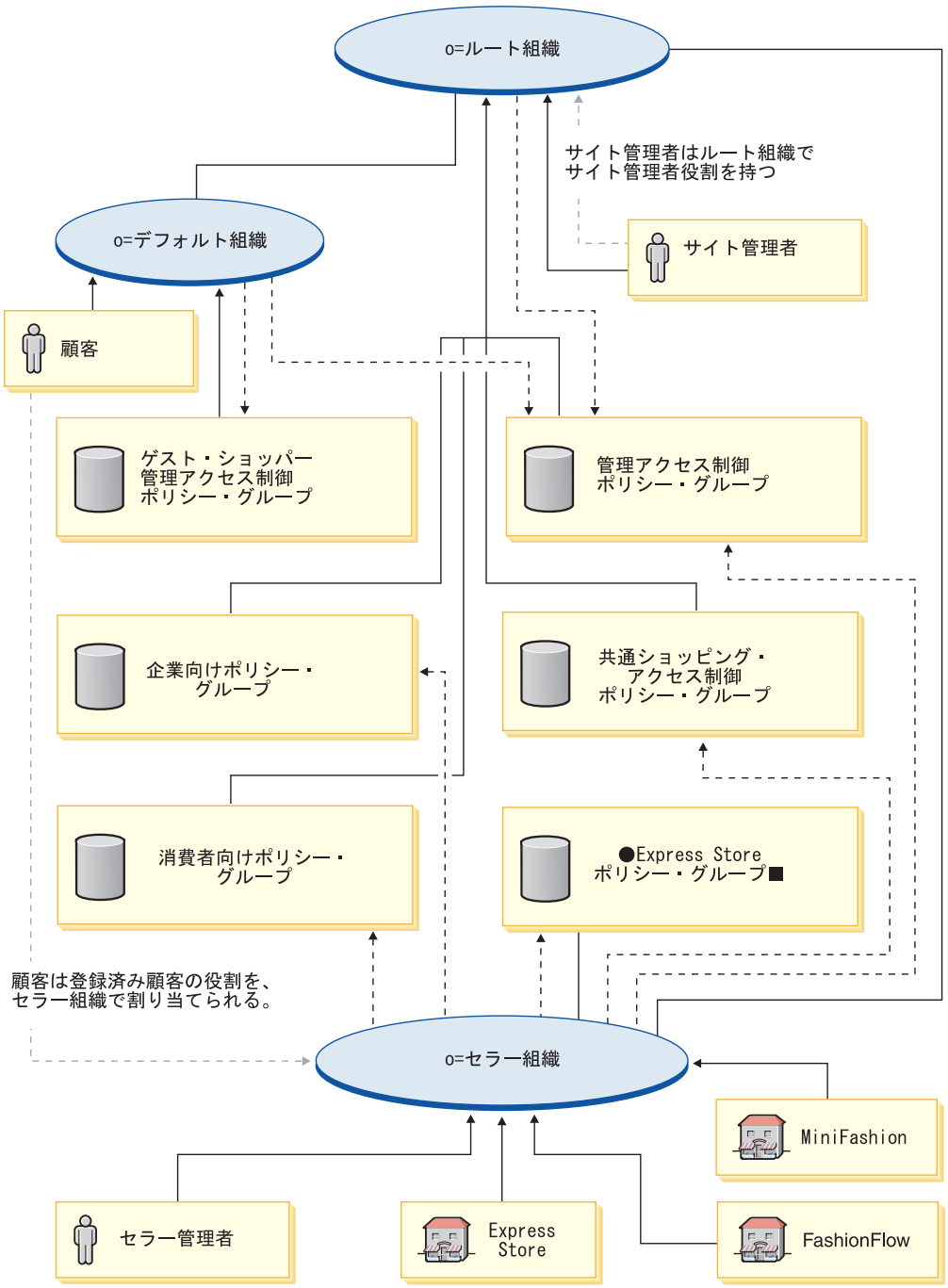
消費者向け組織は、FashionFlow ポリシー・グループも所有し、このグループをサブスクライブします。 FashionFlow ポリシー・グループには、以下のポリシーが含まれます。

AllUsersExecuteFashionAllUsersViews

アクセス制御ポリシー・グループは、組織エンティティーによってサブスクライブされるため、サイトで複数のストアを作成していて、個々のストアに異なるアクセス制御ポリシー・グループを適用したいのであれば、それぞれのストアに対して個別の組織を作成する必要があります。

Express WebSphere Commerce - Express のアクセス制御構造は、前述の消費者向けアクセス制御構造とは少し異なります。

以下の図は、WebSphere Commerce - Express のアクセス制御構造を説明しています。



この図では、WebSphere Commerce - Express での消費者向け組織を説明しており、43 ページの『基本のアクセス制御構造』で説明されているように、ルート組織がデフォルト・ポリシー・グループを所有してサブスクライブします。

セラー組織は、消費者向けアクセス制御ポリシー、B2B 向けアクセス制御ポリシー、管理ポリシー・グループ、および共通ショッピング・ポリシー・グループを直接サブスクライブします。

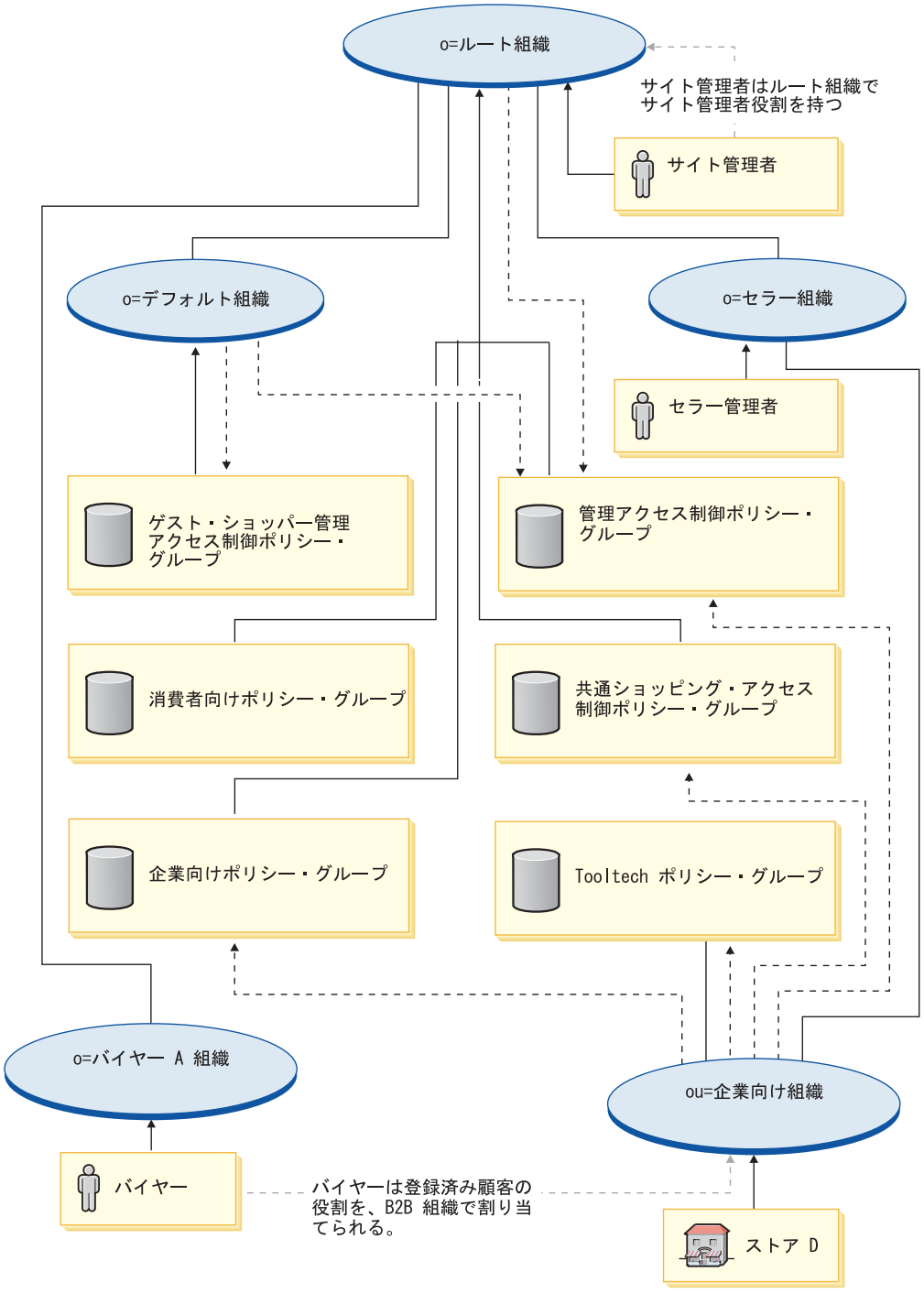
セラー組織は、Express ポリシー・グループも所有し、このグループをサブスクライブします。 Express ポリシー・グループには、以下のポリシーが含まれます。

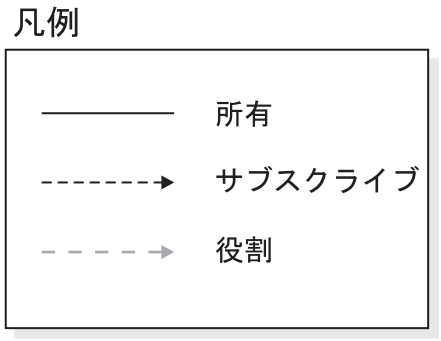
- AllUsersExecuteExpressAllUsersViews
- RegisteredUsersExecuteExpressAllUsersViews

B2B 向け

Business

次の図は、基本的な B2B 向け組織およびアクセス制御構造を説明します。





この図では、基本的な B2B 向け組織構造を説明しており、 43 ページの『基本のアクセス制御構造』で説明されているように、ルート組織がデフォルト・ポリシー・グループを所有してサブスクライブします。

B2B 向け組織は、 B2B、管理、および共通ショッピング・ポリシー・グループを直接サブスクライブします。

B2B 向け組織は、ToolTech ポリシー・グループも所有し、このグループをサブスクライブします。ToolTech ポリシー・グループには、以下のポリシーが含まれます。

- AllUsersForToolTechExecuteToolTechAllUsersViews
- RegisteredCustomersForOrgForToolTechExecuteToolTech RegisteredCustomerViews

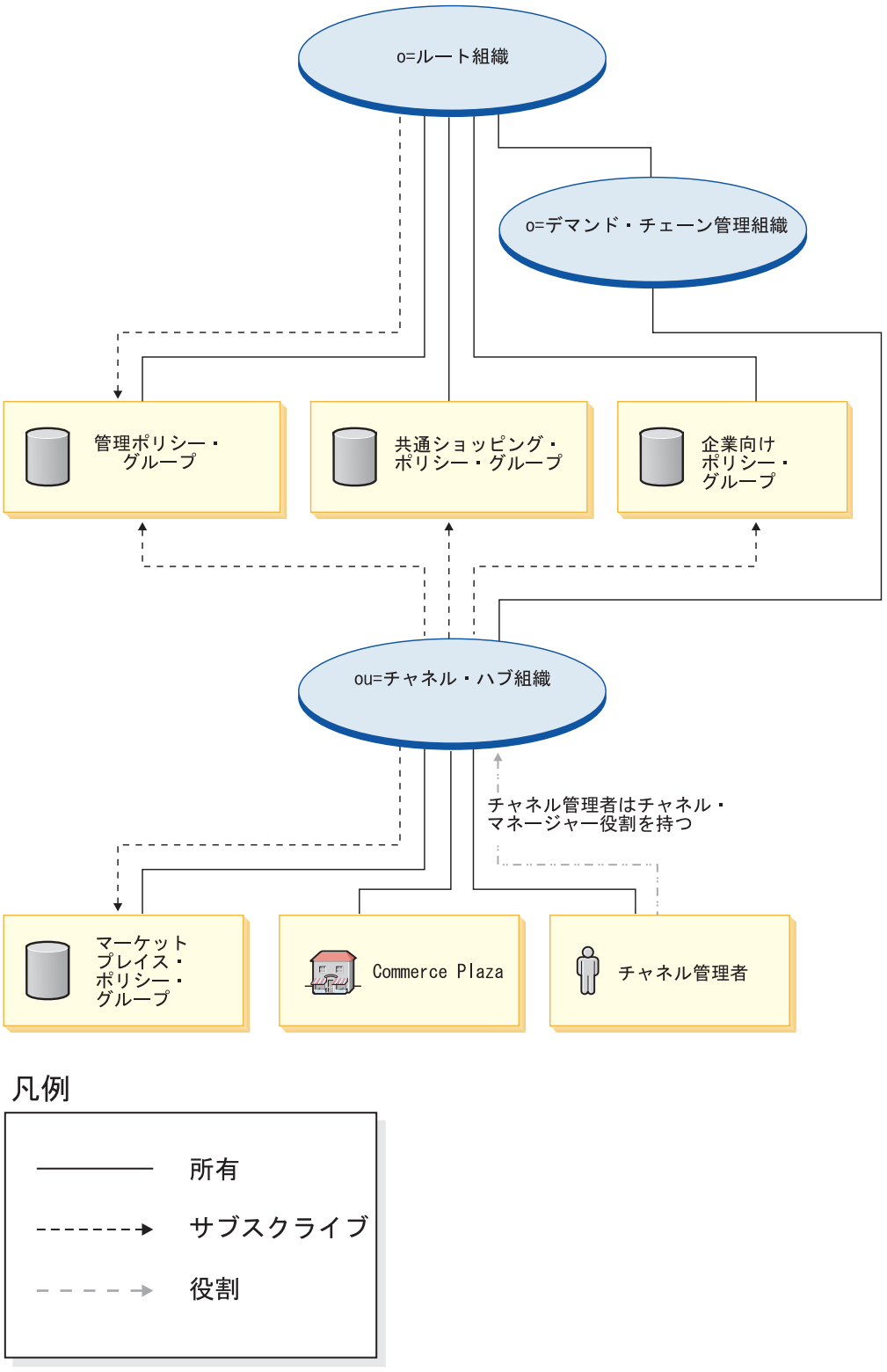
バイヤーとは、B2B 向けストア中でオーダーを発行する顧客のことです。すべてのバイヤーはバイヤー組織によって所有されなければなりません。ルート組織から継承される管理ポリシーは重要なので、通常バイヤー組織はポリシー・グループをサブスクライブしません。

アクセス制御ポリシー・グループは組織エンティティーによってサブスクライブされるため、サイトで複数のストアを作成していて、個々のストアに異なるアクセス制御ポリシー・グループを適用する予定であれば、それぞれのストアに対して個別の組織を作成する必要があります。

デマンド・チェーン

Business

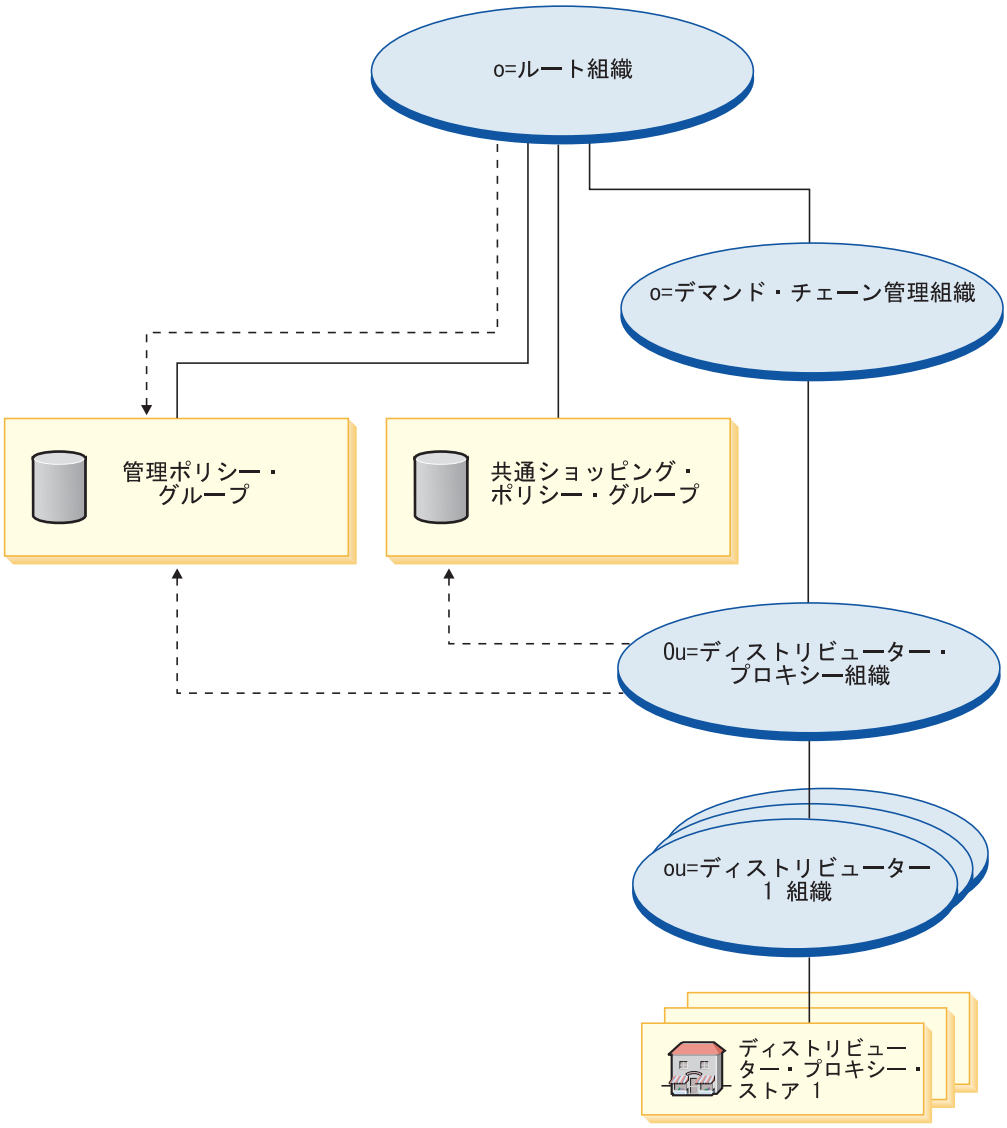
これらの図では、デマンド・チェーン組織構造を説明しており、 43 ページの『基本のアクセス制御構造』で説明されているように、ルート組織がデフォルト・ポリシー・グループを所有してサブスクライブします。

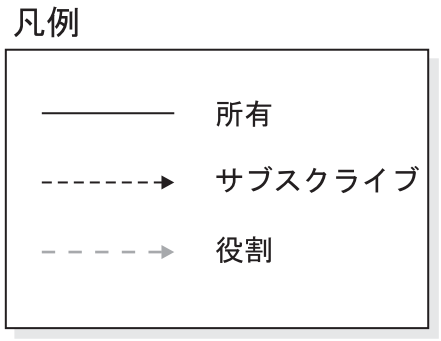


チャネル・ハブ組織は、管理ポリシー・グループ、共通ショッピング・ポリシー・グループ、 B2B ポリシー・グループを直接サブスクライブし、マーケットプレイス・ポリシー・グループを所有してサブスクライブします。その結果これらのポリシーは、チャネル・ハブ (Commerce プラザ) だけでなく、チャネル・ハブ組織のすぐ下のチャネル管理者にも適用されます。

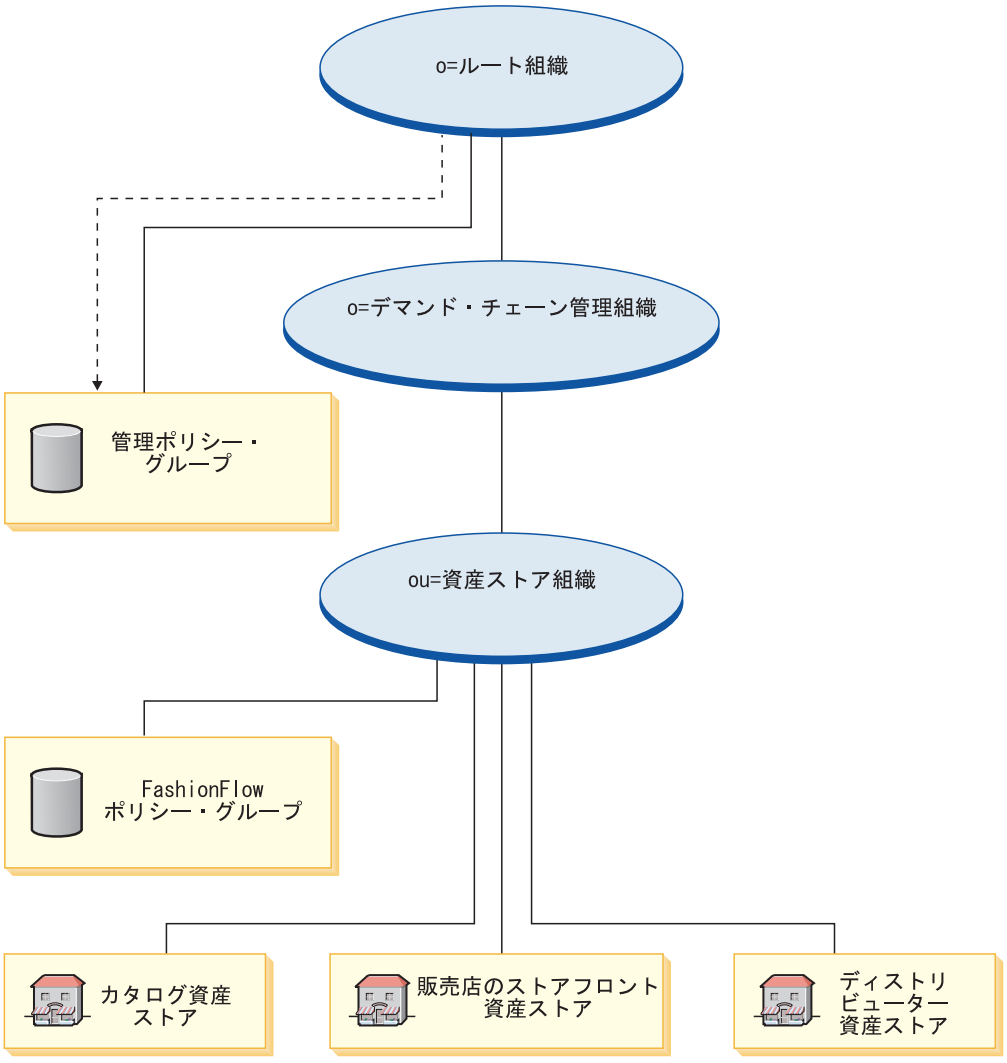
マーケットプレイス・ポリシー・グループには、以下のポリシーが含まれます。

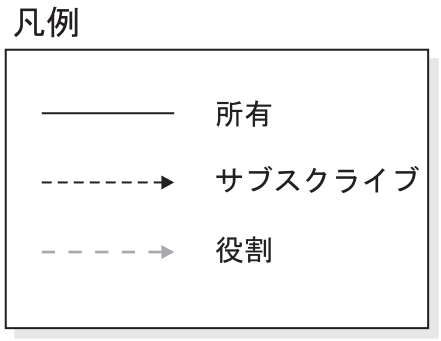
- AllUsersExecuteMarketplaceAllUserViews
- RegisteredCustomersForOrgExecuteMarketplaceRegistered CustomerViews
- ContractAdministratorsForChannelOrgExecuteCreate CommandsOnMemberResource
- ContractAdministratorsForChannelOrgExecuteContract DeployCommandsOnContractResource
- ContractAdministratorsForChannelOrgDisplayContract DatabeanResourceGroup





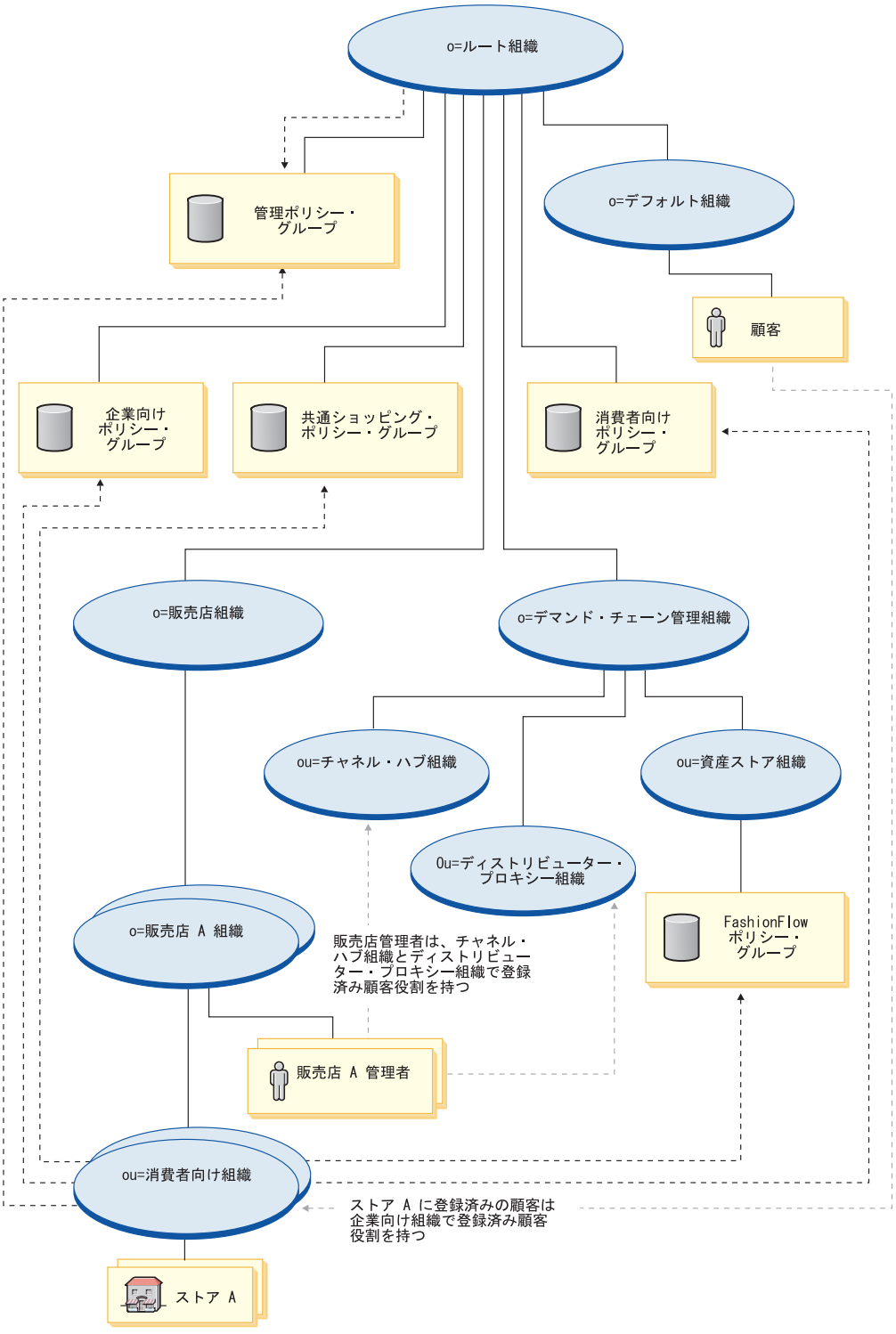
ディストリビューター・プロキシー組織は、管理ポリシー・グループおよび共通ショッピング・ポリシー・グループをサブスクライブします。その結果、これらのポリシーは、ディストリビューター・プロキシー組織のすぐ下のディストリビューター組織に適用されます。

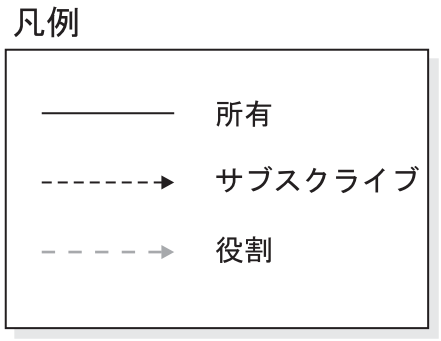




ポリシー・グループを直接サブスクライブしません。その結果、ルート組織から管理ポリシー・グループを継承します。これらのポリシーは、資産ストア組織と、その組織が所有する資産ストアに適用されます。資産ストア組織は FashionFlow ポリシー・グループを所有しますがサブスクライブしません。

注: 個々の販売店消費者向け組織は、販売店ストアの作成時に FashionFlow ポリシー・グループをサブスクライブします。





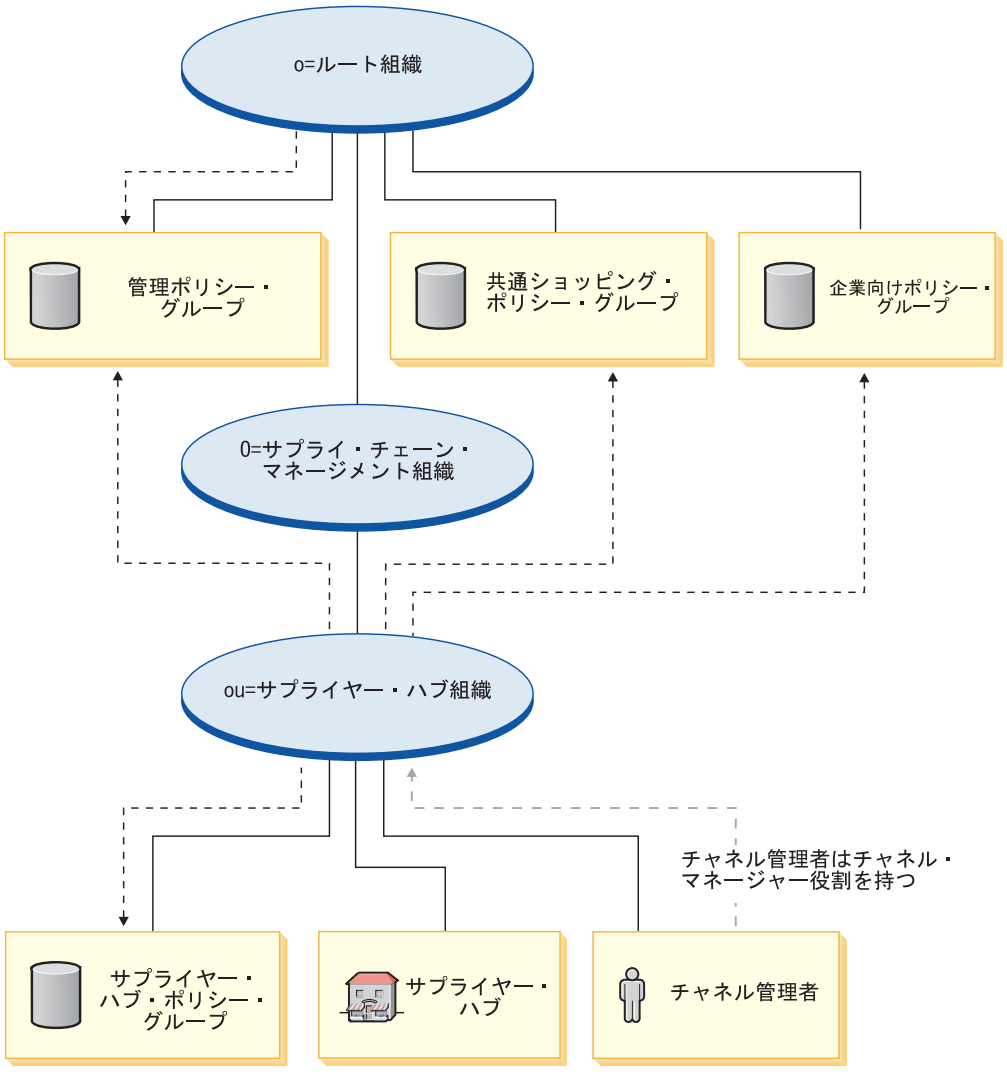
販売店組織は、ポリシー・グループを直接サブスクライブしません。その結果、ルート組織から管理ポリシー・グループを継承します。これらのポリシーは、販売店組織、その組織が所有する販売店 A の組織、および販売店 A の管理者に適用されます。

消費者向け組織は、管理ポリシー・グループ、共通ショッピング・ポリシー・グループ、 B2C および B2B ポリシー・グループ、FashionFlow ポリシー・グループを直接サブスクライブします。これらのポリシーは、消費者向け組織が所有するすべてのストアに適用されます。

サプライ・チェーン

Business

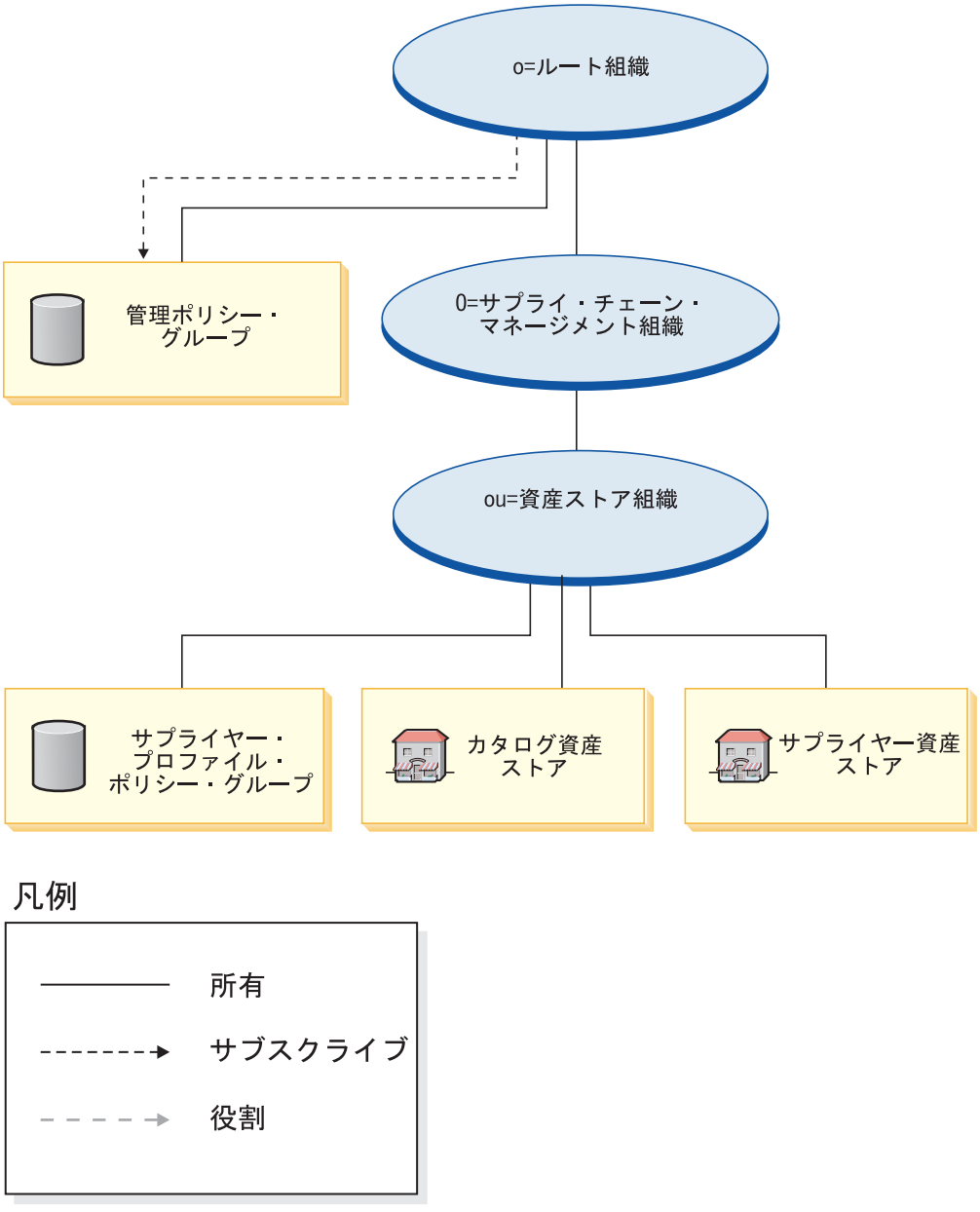
これらの図では、基本的なサプライ・チェーン組織構造を説明しており、 43 ページの『基本のアクセス制御構造』で説明されているように、ルート組織がデフォルト・ポリシー・グループを所有してサブスクライブします。



サプライヤー・ハブ組織は、管理ポリシー・グループ、共通ショッピング・ポリシー・グループ、 B2B ポリシー・グループを直接サブスクライブし、サプライヤー・ハブ・ポリシー・グループを所有してサブスクライブします。その結果、これらのポリシーは、サプライヤー・ハブだけでなく、サプライヤー・ハブ組織のすぐ下のチャネル管理者にも適用されます。

サプライヤー・ハブ・ポリシー・グループには、以下のポリシーが含まれます。

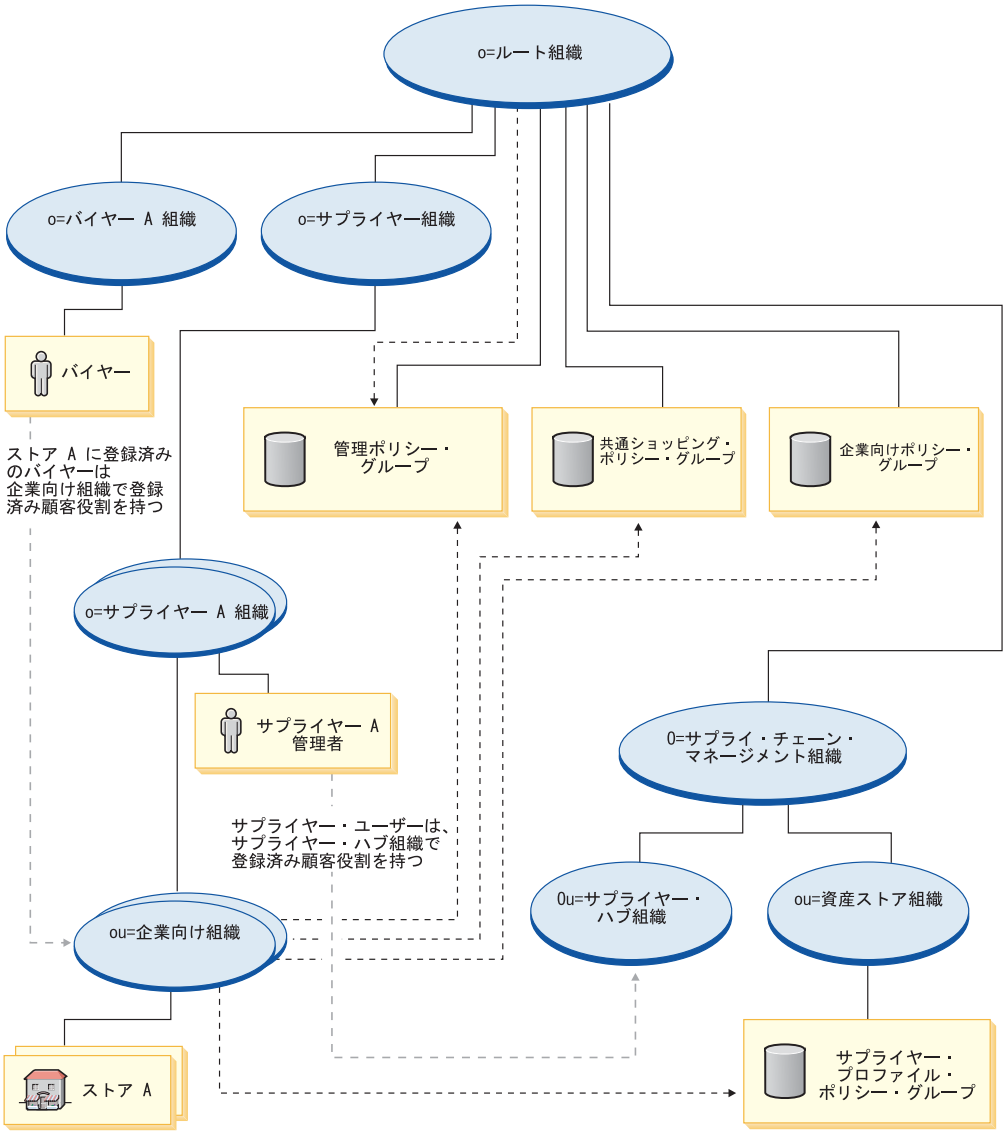
- AllUsersForSupplierHubExecuteSupplierHubAllUsersViews
- RegisteredCustomersForOrgForSupplierHubExecuteSupplierHub RegisteredCustomerViews
- ContractAdministratorsForChannelOrgExecuteCreateCommands OnMemberResource
- ContractAdministratorsForChannelOrgExecuteContractDeploy CommandsOnContractResource
- ContractAdministratorsForChannelOrgDisplayContract DatabeanResourceGroup

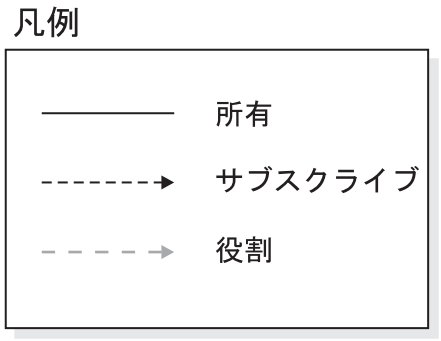


資産ストア組織は、ポリシー・グループを直接サブスクライブしません。その結果、ルート組織から管理ポリシー・グループを継承します。これらのポリシーは、

資産ストア組織と、その組織が所有する資産ストアに適用されます。資産ストア組織はサプライヤー・プロファイル・ポリシー・グループを所有しますがサブスクライブしません。

注: 個々のサプライヤーの B2B 向け組織は、サプライヤー・ストアの作成時にサプライヤー・プロファイル・ポリシー・グループをサブスクライブします。





サプライヤー組織は、ポリシー・グループを直接サブスクライブしません。その結果、ルート組織から管理ポリシー・グループを継承します。これらのポリシーは、サプライヤー組織、その組織が所有するサプライヤー A の組織、およびサプライヤー A の管理者に適用されます。

B2B 向け組織は、管理、共通ショッピング、B2B、およびサプライヤー・プロファイル・ポリシー・グループを直接サブスクライブします。これらのポリシーは、B2B 向け組織が所有するすべてのストアに適用されます。

サプライヤー・プロファイル・グループには、以下のポリシーが含まれます。

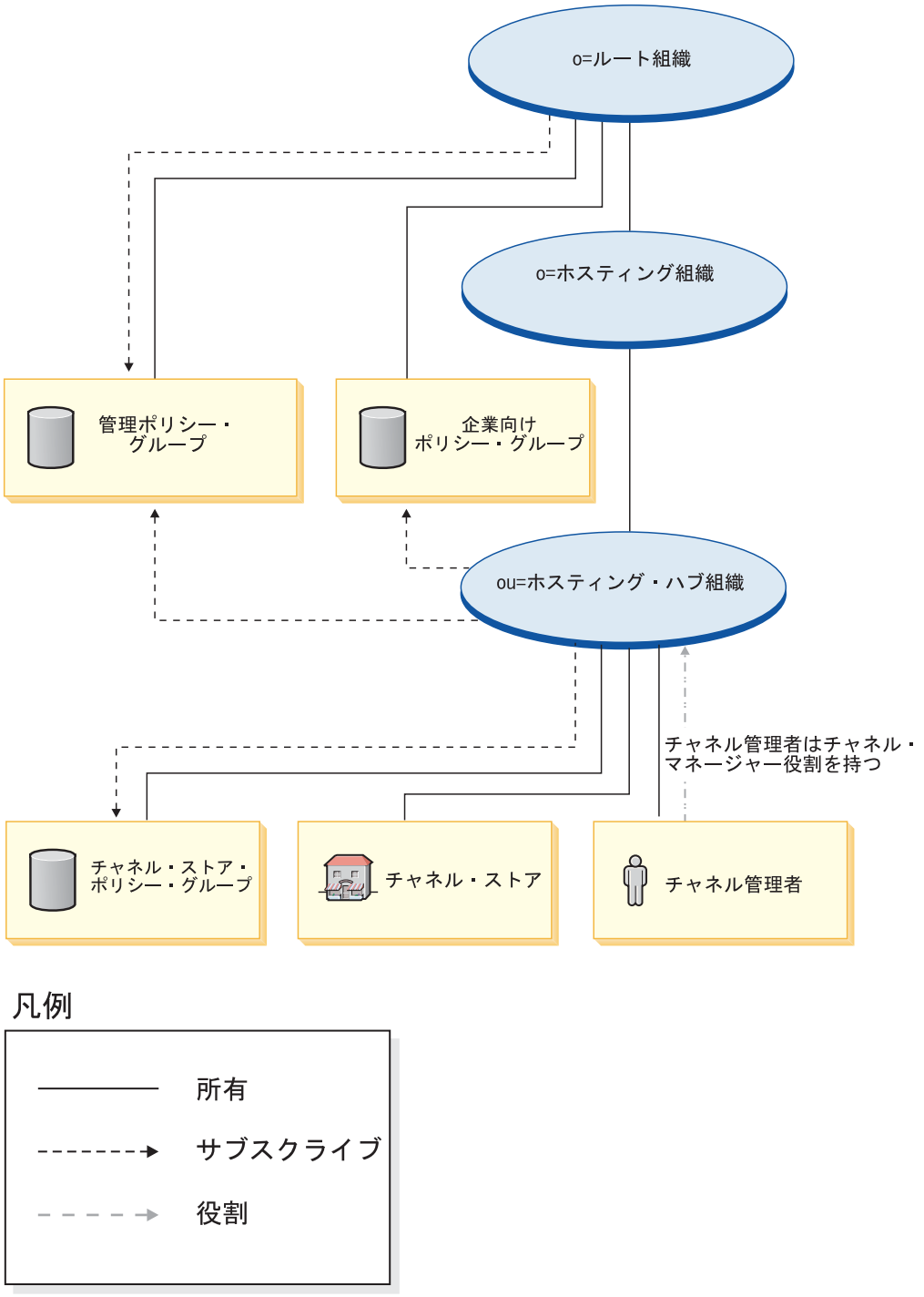
- AllUsersForSupplierExecuteSupplierAllUsersViews
- RegisteredCustomersForOrgForSupplierExecuteSupplierRegisteredCustomerViews

バイヤーとは、B2B ストア中でオーダーを発行する顧客のことです。すべてのバイヤーはバイヤー組織によって所有されなければなりません。ルート組織から継承される管理ポリシーは重要なので、通常バイヤー組織はポリシー・グループをサブスクライブしません。

ホスティング

Business

これらの図では、基本的なホスティング組織構造を説明しており、 43 ページの『基本のアクセス制御構造』で説明されているように、ルート組織がデフォルト・ポリシー・グループを所有してサブスクライブします。

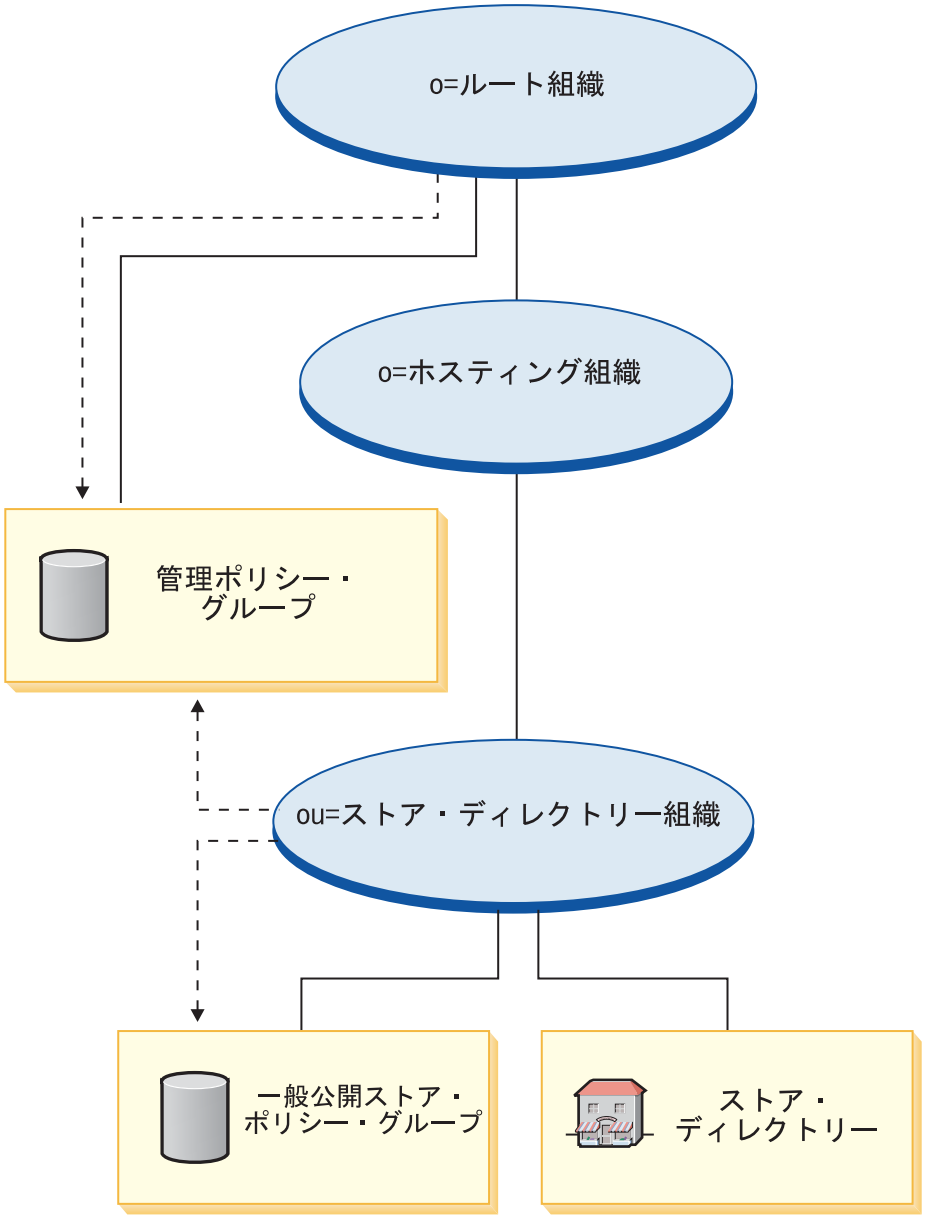


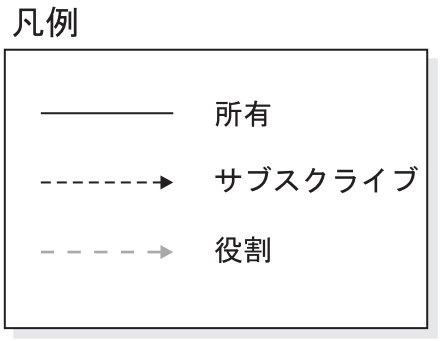
ホスティング・ハブは、管理ポリシー・グループ、 B2B ポリシー・グループを直接サブスクライブし、チャネル・ストア・ポリシー・グループを所有してサブスクライブします。その結果、これらのポリシーは、チャネル・ストア (ホスティング・ハブ) だけでなく、ホスティング・ハブ組織のすぐ下のチャネル管理者にも適用されます。

ホスティング・ハブ・グループには、以下のポリシーが含まれます。

- AllUsersExecuteChannelStoreAllUsersViews

- ContractAdministratorsForChannelOrgExecuteCreate CommandsOnMemberResource
- ContractAdministratorsForChannelOrgExecuteContract DeployCommandsOnContractResource
- ContractAdministratorsForChannelOrgDisplayContract DatabeanResourceGroup

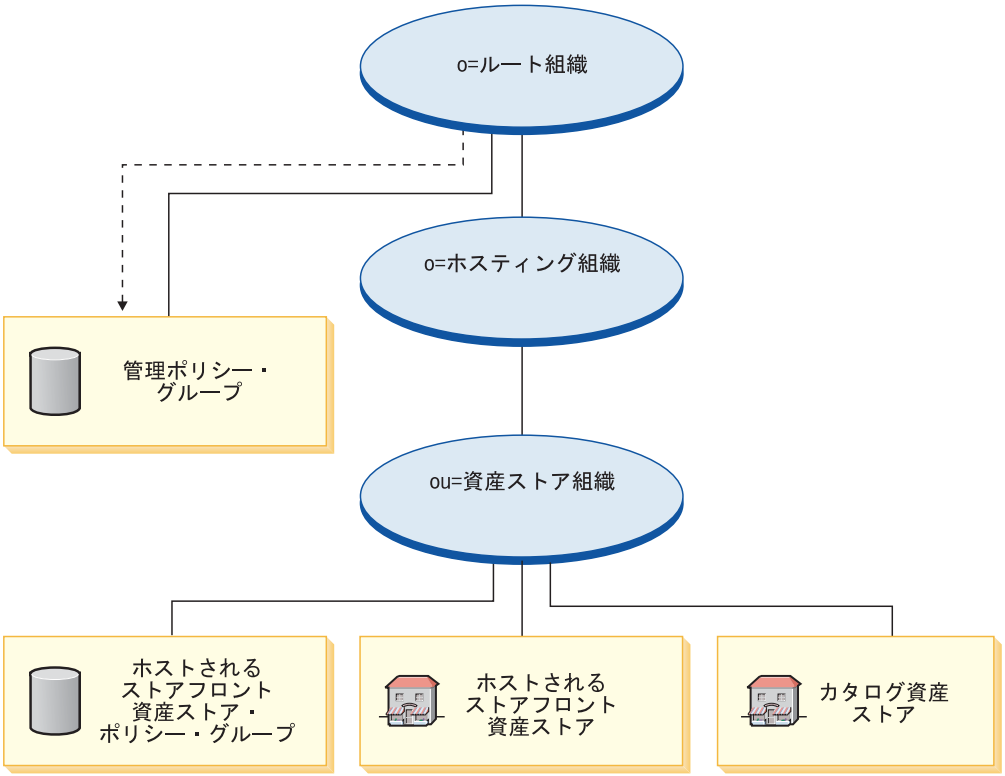


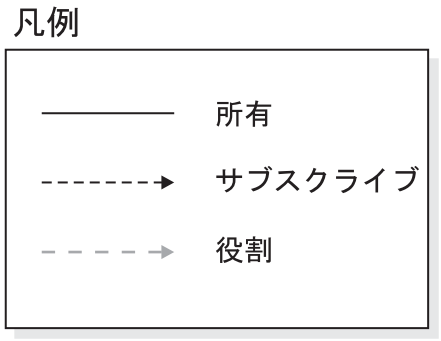


ストア・ディレクトリー組織は、管理ポリシー・グループを直接サブスクライブし、ストア・ディレクトリー・ポリシー・グループを所有してサブスクライブします。その結果、これらのポリシーは、ストア・ディレクトリー組織のすぐ下のストア・ディレクトリーに適用されます。

ストア・ディレクトリー・ポリシー・グループには、以下のポリシーが含まれます。

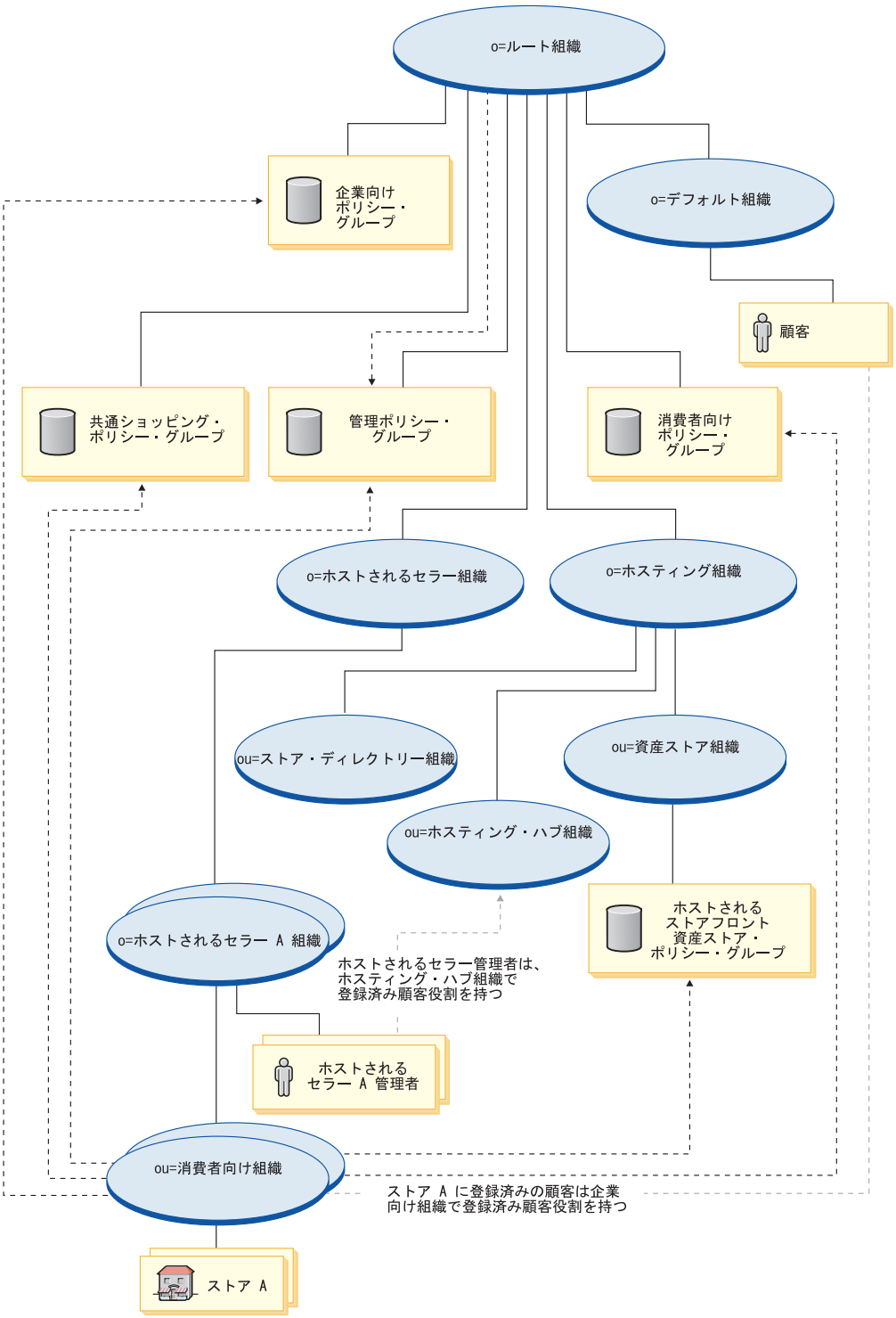
- AllUsersExecutePublicStoreAllUsersViews

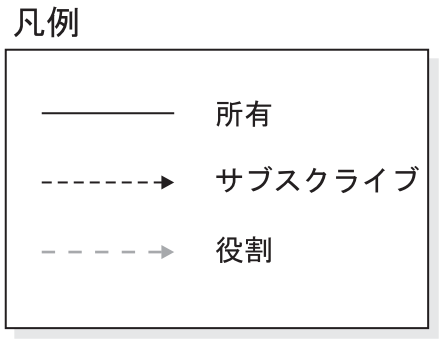




資産ストア組織は、ポリシー・グループを直接サブスクライブしません。その結果、ルート組織から管理ポリシー・グループを継承します。これらのポリシーは、資産ストア組織と、その組織が所有する資産ストアに適用されます。資産ストア組織はホストされるストアフロント資産ストア・ポリシー・グループを所有しますがサブスクライブしません。

注: 個々のホストされるセラー組織は、ホストされるストアの作成時に、ホストされるストアフロント資産ストア・ポリシー・グループをサブスクライブします。


o=ルート組織
企業向け
ポリシー・
グループ
o=デフォルト組織
顧客
共通ショッピング・
ポリシー・グループ
管理ポリシー・
グループ
消費者向け
ポリシー・
グループ
o=ホストされるセラー組織
o=ホスティング組織
ou=ストア・ディレクトリー組織
ou=資産ストア組織
ou=ホスティング・ハブ組織
o=ホストされるセラー A 組織
ホストされる
ストアフロント
資産ストア・
ポリシー・グループ
ホストされるセラー管理者は、
ホスティング・ハブ組織で
登録済み顧客役割を持つ
ホストされる
セラー A 管理者
ou=消費者向け組織
ストア A に登録済みの顧客は企業
向け組織で登録済み顧客役割を持つ
ストア A



ホストされるセラー組織は、ポリシー・グループを直接サブスクライブしません。その結果、ルート組織から管理ポリシー・グループを継承します。これらのポリシーは、ホストされるセラー組織、その組織が所有するホストされるセラー A の組織、およびホストされるセラー A の管理者に適用されます。

消費者向け組織は、管理、共通ショッピング、B2B、B2C、およびホストされるストアフロント資産ストア・ポリシー・グループを直接サブスクライブします。これらのポリシーは、消費者向け組織が所有するすべてのストアに適用されます。

サンプル・ビジネスでのアクセス制御

WebSphere Commerce の各サンプル・ビジネスには、アクセス制御フレームワークが入っています。アクセス制御フレームワークがこれらのビジネスでインプリメントされる方法の詳細については、「*WebSphere Commerce サンプル・ストア・ガイド*」を参照してください。

ストアへのアクセス制御の追加

ストアにアクセス制御を追加するための詳細については、 327 ページの『第 33 章 ストアでのアクセス制御』を参照してください。

第 5 章 WebSphere Commerce ビジネス・ポリシー・フレームワーク

Business ビジネス・ポリシーとは、ストアまたはストアのグループが従う一連のルールであり、ビジネス・プロセス、業界慣例、およびストアやストアのグループのオファリングの範囲や特性、およびストアまたはサイトが顧客や他のビジネス・パートナーと対話する方法を定義するものです。たとえば、サイトには、顧客によるストアへの製品の返品を許可する時と方法を決めるビジネス・ポリシーや、ストアが受け入れる支払いメソッドを決めるビジネス・ポリシーなどがあります。

WebSphere Commerce ビジネス・ポリシー・フレームワークについて

WebSphere Commerce は、オンライン・ストアまたはサイトで、ストアのビジネス・ポリシーをインプリメントできるフレームワークを提供しています。ビジネス・ポリシー・フレームワークは、以下の部分で構成されています。

- ビジネス・ポリシー
- Business ビジネス・アカウント
- 契約および Business サービス契約
- 契約条件

ビジネス・ポリシー

たいていの場合、オンライン・ストアまたはサイトでインプリメントする必要のあるビジネスには、ビジネス・ポリシーを事前定義しておきます。 WebSphere Commerce は、そのまま使用したり、必要に応じて変更できるビジネス・ポリシーのセットを提供しています。 WebSphere Commerce が提供するデフォルト・ビジネス・ポリシーの詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。これらのビジネス・ポリシーの編集方法の詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

ビジネス・アカウント

Business

ビジネス・アカウントは、顧客とビジネスとの間の関係を定義します。ビジネス・アカウントは、顧客組織の契約とオーダーを追跡し、顧客組織からのバイヤーが、ストアで購入する方法を構成します。

契約およびサービス契約

顧客またはビジネス・パートナー (たとえば販売店やディストリビューター) がストアにアクセスする前に、顧客またはビジネス・パートナーによるストアへのアクセスを定義する契約またはサービス契約を作成することが必要です。 WebSphere Commerce ビジネス・ポリシー・フレームワークでは、顧客には契約を、他のタイプのビジネス・パートナーにはサービス契約を作成します。

- 契約: 顧客との契約は、顧客がアクセスできるストアの領域、顧客に表示される価格、およびサイトとその価格を利用できる期間を定義します。すべてのストアには、内部管理者だけがストアにアクセスできる契約を別として、最低 1 つの契約が必要です。 WebSphere Commerce は、ストアでショッピングをするすべての顧客に適用されるデフォルトの契約を提供しています。 WebSphere Commerce Professional Edition では、デフォルトの契約が、サポートされている唯一の契約です。
- Business サービス契約: ビジネス・パートナー (販売店、ディストリビューター、製造メーカー、サプライヤー、または他のパートナー) とのサービス契約は、ビジネス・パートナーとの取り決めを定義します。たとえば、販売店とのサービス契約は、販売店からサイトへのアクセス方法、カタログを共用できるかどうか、または販売店のためにストアをホストできるかどうかなどを定義します。ディストリビューターとのサービス契約は、サイトの顧客がディストリビューターから見積もりを受け取る方法や、顧客がサイトからディストリビューターのサイトにアクセスする方法を定義します。

契約条件

契約条件は、契約およびサービス契約を、特定の顧客またはビジネス・パートナー向けにインプリメントする方法を定義します。契約では、契約で販売されるもの、販売されるアイテムの価格、顧客へのアイテムの配送方法、およびオーダーの支払方法などが定義されます。ビジネス・パートナーとのサービス契約では、契約条件は、ビジネス・パートナーが販売可能な商品を制限する場合があります。

サイトまたはストア操作のほとんどの局面がビジネス・ポリシーによって定義されるので、契約条件は通常、ビジネス・ポリシーを参照します。契約条件は、自分が参照するビジネス・ポリシーに対して標準パラメーターを提供します。ビジネス・ポリシーにパラメーターを提供することによって、各契約ごとにビジネス・ポリシーの振る舞いを変更することができます。

サンプル・ビジネスでのビジネス・ポリシー

WebSphere Commerce の各サンプル・ビジネスには、ビジネス・ポリシー・フレームワークが入っています。ビジネス・ポリシー・フレームワークがこれらのビジネスでインプリメントされる方法の詳細については、「*WebSphere Commerce サンプル・ストア・ガイド*」を参照してください。

ビジネス・ポリシーのサイトへの追加

サイトでビジネス・ポリシー・フレームワークをインプリメントする方法の詳細については、205 ページの『第 18 章 契約資産』を参照してください。

第 6 章 インスタンス・アーキテクチャー

この章では、WebSphere Commerce Server のインスタンス・アーキテクチャーについて概要を説明します。

WebSphere Commerce Server

WebSphere Commerce Server は、e-commerce ソリューションのうち、ストアおよびコマースに関連したさまざまな機能を処理する WebSphere Application Server アプリケーションです。ストアフロント資産およびビジネス・ロジックは、 WebSphere Commerce Server 内の Web アプリケーションの中に存在しています。 WebSphere Commerce には、デフォルトで使用できる Web アプリケーション (Stores.war) が用意されていますが、独自に作成することもできます。

1 つの Web アプリケーションには、1 つのストアのための資産が含まれる場合と、複数のストアのための資産が含まれる場合とがあります。 1 つの Web アプリケーションに複数のストアフロントおよびビジネス・ロジックが含まれる場合、各ストアの資産はストア・ディレクトリー (storedir) によって分離されます。

WebSphere Commerce Server インスタンス

WebSphere Commerce Server インスタンスは、関連するデータベースを伴う展開された WebSphere Application Server アプリケーションです。 1 つのインスタンスで、複数のストアをサポートできます。 1 つのインスタンス中のすべてのストアは同じデータベースを共用します。さらにはカタログ、フルフィルメント、または領収証などのある種のデータも共用できます。また、1 つのインスタンスに含まれるすべてのストアは、同じ EJB コンテナーを共用します。

1 つのインスタンスに含まれる単一のストアを作成したり、 1 つのインスタンスの中に複数のストアを作成したりすることができます。インスタンスでの複数のストアの詳細については、 72 ページの『単一インスタンス中の複数のストア』 を参照してください。

第 7 章 ストア・アーキテクチャー

オンライン・ストアの作成をサポートするために、WebSphere Commerce ではストア・アーキテクチャーを提供します。このアーキテクチャー、およびこれを使用してインプリメントできるストアのいくつかの例をこの章で説明します。

WebSphere Commerce ストア・アーキテクチャーについて

サイトでストアをサポートするために、WebSphere Commerce ではストア・アーキテクチャーを提供することによって、オンライン・ストアの作成を可能にします。ストア・アーキテクチャーは、以下のコンポーネントで構成されます。

- ストア資産
- 単一インスタンスでの複数ストアのサポート
- ストア間の関係

ストア資産

WebSphere Commerce では、オンライン・ストアは、オンライン・ビジネスのすべてのトランザクションが発生する場所です。 WebSphere Commerce で作成されるすべてのオンライン・ストアには、次のタイプの資産のいずれかが最低 1 つ入っています。

- ストアフロント: ストアの外部を構成する部分、つまり顧客に対して表示される部分は、ストアフロントと呼ばれます。ストアフロント資産は、HTML ページ、JSP ファイル、スタイルシート、イメージ、グラフィックス、およびその他のマルチメディア・ファイル・タイプなどの Web 資産で構成されています。このマニュアルでは、ストア・ページを構築するために必要な JSP ファイルの作成に関係したさまざまな概念や作業について説明します。詳細については、83 ページの『第 4 部 ストアフロントの開発』を参照してください。
- ビジネス・ロジック: コマンド、カスタマイズされたコードを含む顧客要求を処理するストアの部分は、ビジネス・ロジックと呼ばれます。ビジネス・ロジックとカスタマイズされたコードの作成の詳細については、「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」を参照してください。
- ストア・データ: ストアを構成するデータ資産。ストアが正しく動作するためには、顧客のすべてのアクティビティーをサポートするデータがストアに配置されていなければなりません。たとえば、顧客が商品を購入するためには、販売商品のカタログ、オーダー処理のためのプロセス、要求を実行する在庫機能、および配送プロセスが必要です。さらに、支払を処理し、集金するための手段も必要です。ストア・データの作成に関係するさまざまな概念や作業については、123 ページの『第 6 部 ストア・データの開発』で説明します。

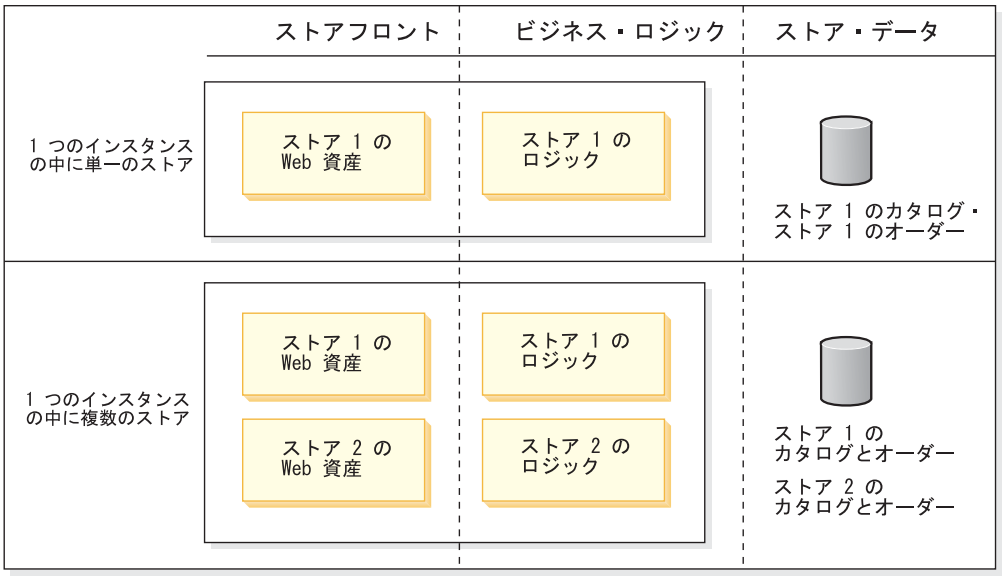
ストアにストアフロント資産、ビジネス・ロジック、およびストア・データの 3 つのタイプがすべて含まれる場合、そのストアは完全に操作可能なストアです。ストアに含まれるのが資産のサブセットだけである場合、つまりストアフロント資産とビジネス・ロジック、またはストア・データとビジネス・ロジック、またはストア・データだけである場合は、WebSphere Commerce で*資産ストア* と呼ばれます。

資産ストア

資産ストアは、他のストアで利用できる共用可能な資源 (ビジネス成果物、ビジネス・プロセス、およびストアフロント資産) のコレクションです。たとえば、ハブ・ストアの一部としてカタログを作成するのではなく、ハブ・ストアが、ハブのチャネルまたはパートナーも共用できるカタログ資産ストアを利用できるかもしれません。資産ストアは通常、複数のストアが使用できる資産で構成されています。詳細については、74 ページの『ストア間の関係』を参照してください。

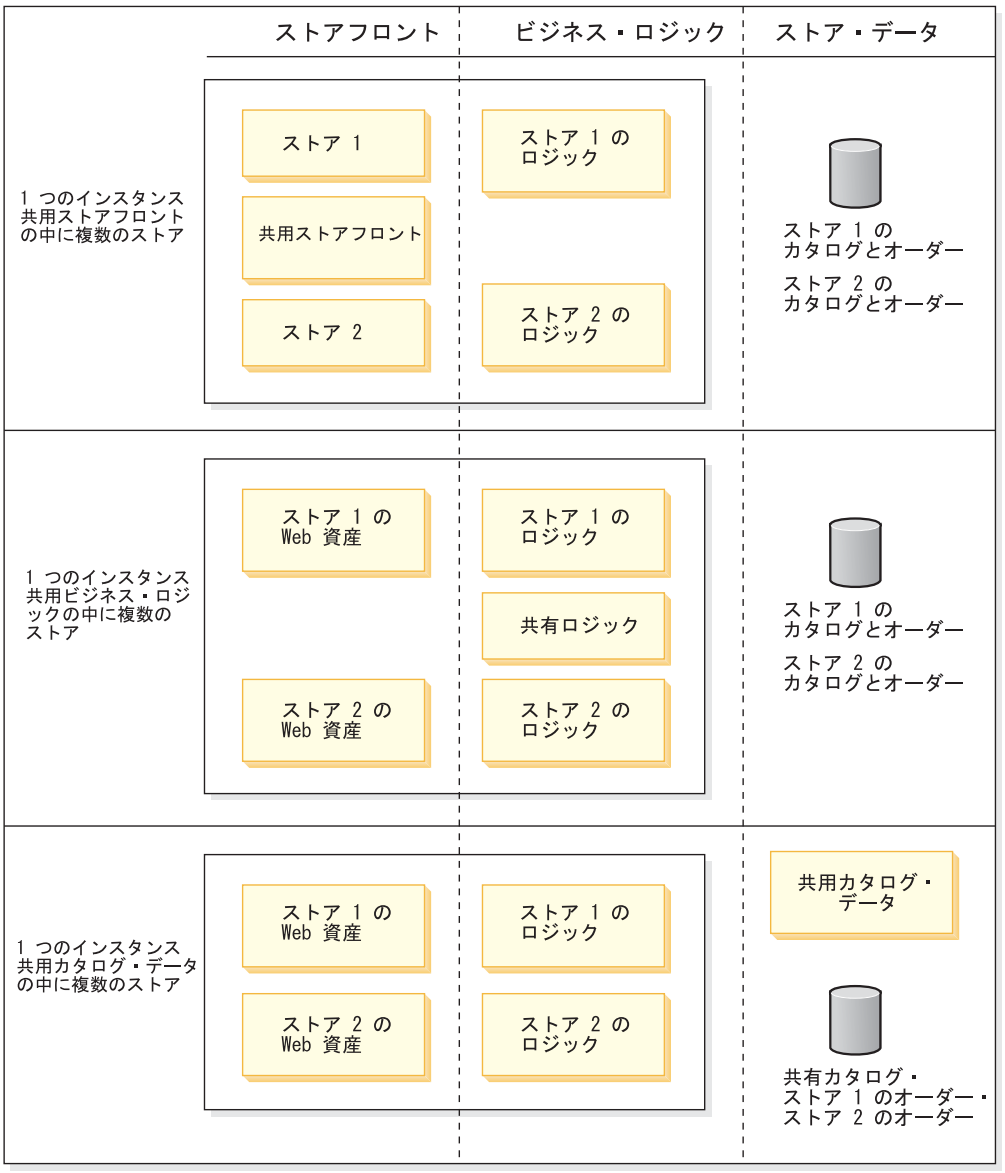
単一インスタンス中の複数のストア

WebSphere Commerce により、WebSphere Commerce Server インスタンス内で、複数のオンライン・ストアをサポートできます。次の図は、可能性のあるいくつかのストア構成を示しています。



前述の図で詳しく説明されているストアは、スタンドアロン・ストアです。つまり、同じインスタンスの中にあっても、データを共用したり、互いに関係したりすることはありません。個別のストアフロント、ビジネス・ロジック、およびストア・データを持ちます。

また、同じストアフロント、同じビジネス・ロジック、同じストア・データ (カタログを含む)、またはこれら 3 つのうちの任意の組み合わせを共用するインスタンスの中で、複数のストアを作成することもできます。次の図は、ストアが資産を共用するための可能な構成をいくつか示します。



注: 前述の図は、1 つのインスタンス中の複数のストア間で考えられる構成を数例リストしたものです。ストアは、複数の資産タイプを共用することができます。たとえば、サイト中の複数のストアは、ストアフロント、ビジネス・ロジック、およびデータ、またはこの 3 つからの任意の組み合わせを共用できます。

1 つのインスタンス中の複数のストアが共通のストア資産を共用する方法の詳細については、 74 ページの『ストア間の関係』を参照してください。

複数のストアが 1 つのストア Web モジュール内に存在することができます。その場合、ストア資産は以下の方法で分割されます。

- ストアフロント資産 : ストア Web モジュール内の各ストアのストアフロント資産は、別のストア・ディレクトリー (*storedir*) に保管されます。たとえば、MyStore 用のすべてのストアフロント資産は MyStore ディレクトリーに入れられます。
- ビジネス・ロジック : ストア ID を使用して、コマンド・レジストリーで指定された各ストアのコマンド・インプリメンテーションが選択されます。
- ストア・データ : データ資産は、ストアごとに固有索引によって識別されます。

ストア間の関係

Business 同じストアフロント、ビジネス・ロジック、ストア・データ、または共用資産の任意の組み合わせを持つサイトで複数のストアをサポートするため、また 1 つのホストが別のホストをホスティングしたり、ショッピング・カートをあるストアから別のストアに転送したりする、サイト内のストア間の他のタイプの関係をサポートするため、WebSphere Commerce はストア間のさまざまな関係のアーキテクチャーを提供します。

ストア間の関係により、あるストアが別のストアにサービスを提供できます。たとえば、ストア A はストア B のホストになることができ、ストア C はストア D からのカタログ・データを使用できます。

これらのストア関係をインプリメントするには、各ストア関係をサポートするコードが必要です。 WebSphere Commerce には、多くのストア関係と、サポートするコードが入っています。これらのストア関係は、大まかに次のカテゴリーにグループ化されています。

- あるストアが別のストアに資産を提供する関係。これらのタイプのストア関係には、あるストアが、URL、コマンド、ビジネス・ポリシー、プロパティー・ファイル、および通貨を別のストアに提供することが含まれます。
- あるストアが別のストアとの間に「ビジネス関係」を持つ関係。これらのタイプのストア関係は、あるストアが別のストアをホスティングする関係、またはあるストアが別のストアに対してオーダーを参照する関係が含まれます。

注: WebSphere Commerce が提供するデフォルトのストア関係の詳細なリストについては、 151 ページの『第 14 章 ストア間の関係』を参照してください。

ストア・アーキテクチャーがビジネス・モデルをサポートする方法について

ビジネス・モデルに必要なストアをサポートするために、WebSphere Commerce はストア・アーキテクチャーを使って、以下のタイプのストアを作成します。

- 顧客対面ストア
- プロキシー・ストア
- 資産ストア

注: これらの特定のストアは、WebSphere Commerce がサポートするビジネス・モデルをインプリメントするために推奨されています。ストア・アーキテクチャーを使用して、独自のタイプのストアを作成することもできます。

顧客対面ストア

顧客対面ストアは、顧客が直接アクセスできるストアです。これらのストアは、サイトのメイン・コンポーネントです。 WebSphere Commerce は次のタイプの顧客対面ストアをサポートします。

- ダイレクト・セールス・ストア: ビジネスと消費者、または 2 つのビジネスまたはパーティー間で、商品、サービス、または情報が関係する商取引を直接サポートするストア。 WebSphere Commerce は、以下の 2 つのタイプのダイレクト・セールス・ストアをサポートしています。
  - 消費者向け
  - Business B2B 向け
- Business ハブ・ストア: サイト上の他のストアを使用して、ハブ所有者の 1 人以上のパートナーまたはクライアントから購入可能な商品またはサービスに、顧客またはパートナーがアクセスするためのストア。
- Business ホストされるストア: ストアの所有者向けに、サイト・オペレーターによってホストされるストア。ストア所有者はストアを管理するオプションを持っている場合があります。

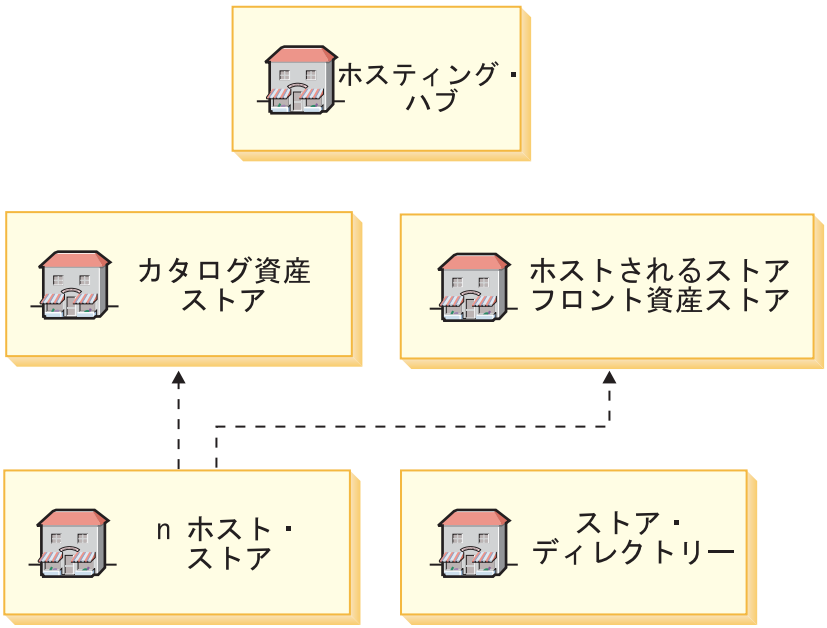
ダイレクト・セールス・ストアおよびハブ・ストアの作成

ダイレクト・セールス・ストアと Business ハブ・ストアは、ストア作成の点では、WebSphere Commerce で最も典型的なストアです。つまり、各ストアごとにストアフロント資産、ビジネス・ロジック、およびストア・データを作成する必要があります。そのストア専用の資産を作成することによって、これらの資産を従来どおりに作成するというオプションがあります。しかし、資産ストアまたはストア全体で使用できるデータのどちらかでストアフロントおよびビジネス・ロジックを作成することによって、他のストアが使用する資産を作成するというオプションもあります。また、他のストアからの資産を使用して、ダイレクト・セールス・ストアまたはハブ・ストアの部分を作成することもできます。

ストアフロント資産の作成の詳細については、83 ページの『第 4 部 ストアフロントの開発』を参照してください。ビジネス・ロジックとカスタマイズされたコードの作成の詳細については、「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」を参照してください。ストア・データの作成の詳細については、123 ページの『第 6 部 ストア・データの開発』を参照してください。ストア間の資産の共用の詳細については、151 ページの『第 14 章 ストア間の関係』を参照してください。

ホストされるストアの作成

Business WebSphere Commerce が提供するサンプルでは、ホストされるストアの大多数は、既存の資産ストアからの資産を共用することによって作成されます。たとえば、ホスティングしている各ストアごとにストアフロントまたはカタログ資産を作成するのではなく、別のストアからストアフロント、およびビジネスによってはカタログを使用します。ホスティング・ストアの作成を容易にするために、WebSphere Commerce は資産ストアを使用します。次の図は、ホストされるストアがホストされるストアフロント資産ストアおよびカタログ資産ストアから資産を使用する方法を示します。



ホストされるビジネス管理者には、ストアをカスタマイズするための外見上の変更 (新しいルック・アンド・フィール、独自の新規ロゴ、および独自のテキストの一部など)、さらに特定のデータの変更 (カタログのフィルター操作、価格の変更など) のオプションがあります。

また、ホストされるストアごとにストアフロント資産、ビジネス・ロジック、およびストア・データを別々に作成することによって、ホストされるストアを従来どおり作成することもできます。ストアフロント資産の作成の詳細については、83 ページの『第 4 部 ストアフロントの開発』を参照してください。ビジネス・ロジックとカスタマイズされたコードの作成の詳細については、「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」を参照してください。ストア・データの作成の詳細については、123 ページの『第 6 部 ストア・データの開発』を参照してください。

「ストア作成 (Store Creation)」ウィザード: WebSphere Commerce が提供する「ストア作成 (Store Creation)」ウィザードにより、ホストされるストアをすばやく、容易に作成することができます。ウィザードでは、顧客がストアについてのいくつかの基本的なデータ (名前、説明など) を提供するように促し、顧客が使用したいストアフロントまたはカタログを選択できるようにしてから、顧客に合わせてストアを作成します。その結果、作成されるストアには固有のデータ (固有のストアにするための基本データ) がありますが、使用するストアフロントおよびカタログ・データは既存の資産ストアからのものです。

「ストア作成 (Store Creation)」ウィザードの動作はテンプレートによって制御されます。このテンプレートは、ストア関係、配送モード、メッセージ、および共用される配送センターを含む、ホストされるストアを作成するのに使用できるオプションを判別します。 WebSphere Commerce では「ストア作成 (Store Creation) ウィザードのいくつかのテンプレートを、サポートされるビジネス・モデルごとに 1 つずつ提供します。これらのテンプレートは、以下に示すディレクトリーにあります。

WC_installdir/xml/trading/xml

テンプレートは、ウィザードで選択されるストアフロント資産のタイプに基づいて、「ストア作成 (Store Creation)」ウィザードと関連付けられます。たとえば、販売店のストアフロント資産ストア (STORE テーブルの STORETYPE フィールドで RPS として識別される) から資産を使用することを選択した場合、「ストア作成 (Store Creation)」ウィザードは TemplateHostingContractRPS.xml を使用します。

「ストア作成 (Store Creation)」ウィザードを使用して、ホストされるストアを作成する方法の詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

注: 「ストア作成 (Store Creation)」ウィザードを使用しないで、ホストされるストアを作成したい場合は、テンプレートの 1 つに基づいてサービス契約を作成し、それを WebSphere Commerce にインポートすることができます。詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

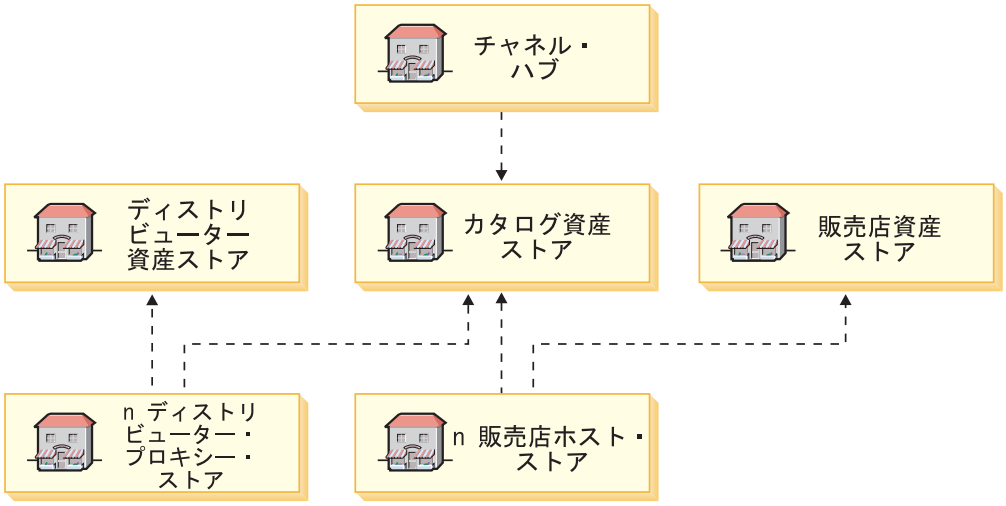
ホストされるストアが共用する資産を変更するには、資産ストアを変更する必要があります。詳細については、78 ページの『資産ストアの作成』を参照してください。

プロキシー・ストア

WebSphere Commerce は、プロキシー・ストアと呼ばれるエンティティーもサポートします。プロキシー・ストアはビジネス・パートナーの運用可能資産を現すストアで、 WebSphere Commerce サイトが外部ビジネス・パートナーと対話するためのビジネス・ロジックを提供します。たとえば、プロキシー・ストアはリモート・オーダー資金化システムに転送されるオーダーだけでなく、サプライヤーの在庫情報、またはサプライヤーの配送センターに送信される情報を取り込むことがあります。顧客対面ストアとは異なり、プロキシー・ストアにはストアフロントが含まれておらず、ユーザーによるアクセスはできません。

プロキシー・ストアの作成

プロキシー・ストアの作成は、プロキシーのストア資産の大部分が既存のストア (資産ストアを含む) から提供されるという点で、ホストされるストアの作成に非常に似ています。 WebSphere Commerce で提供されるサンプルでインプリメントされているとおり、プロキシー・ストアにはストアフロントが入っていません。その結果、別のストアのカタログからの資産のみが共用されます。次の図は、ディストリビューター・プロキシー・ストアが、ディストリビューター資産ストアおよびカタログ資産ストアから資産を使用する方法を示します。



WebSphere Commerce は、ユーザー・インターフェースを提供してプロキシー・ストアを作成するのではなく、サービス契約を介してプロキシー・ストアをインプリメントし、それは WebSphere Commerce にインポートされて、プロキシー・ストアが作成されます。サービス契約はテンプレートによって制御されます。テンプレートは作成する必要がある情報を判別します。プロキシー・ストアを作成するためのテンプレート (TemplateReferralContract.xml) は、次のディレクトリーにあります。

WC_installdir/xml/trading/xml

プロキシー・ストアを作成するには、テンプレートに従って新しいサービス契約を作成してから、それを WebSphere Commerce にインポートします。詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

資産ストア

顧客対面ストアとプロキシー・ストアの作成を容易にするために、WebSphere Commerce は資産ストアをインプリメントします。資産ストアは、他のストアで利用できる共用可能な資源 (ビジネス成果物、ビジネス・プロセス、およびストアフロント資産) のコレクションです。たとえば、ハブ・ストアの一部としてカタログを作成するのではなく、ハブ・ストアが、ハブのチャネルまたはパートナーも共用できるカタログ資産ストアを利用できるかもしれません。資産ストアは通常、複数のストアが使用できる資産で構成されています。詳細については、74 ページの『ストア間の関係』を参照してください。

WebSphere Commerce は、カタログ資産ストアとストアフロント資産ストアの例を提供しています。

資産ストアの作成

資産ストアは、別のストアに資産を提供するストアです。 WebSphere Commerce で提供されるサンプルにインプリメントされているように、資産ストアは資産のコレクションで構成されていますが、完全に機能するストアではありません。資産ストアを作成するには、ダイレクト・セールス・ストアまたはハブ・ストアで資産を作成する場合と同じ方法に従います。つまり、資産ストアにカタログ資産を入れる場合、123 ページの『第 6 部 ストア・データの開発』の指示に従って、カタログ・

データを作成します。資産ストアにストアフロント資産を入れる場合は、83 ページの『第 4 部 ストアフロントの開発』を参照してください。資産ストアにビジネス・ロジックを入れる場合は、「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」を参照してください。

サポートされるビジネス・モデルのストア

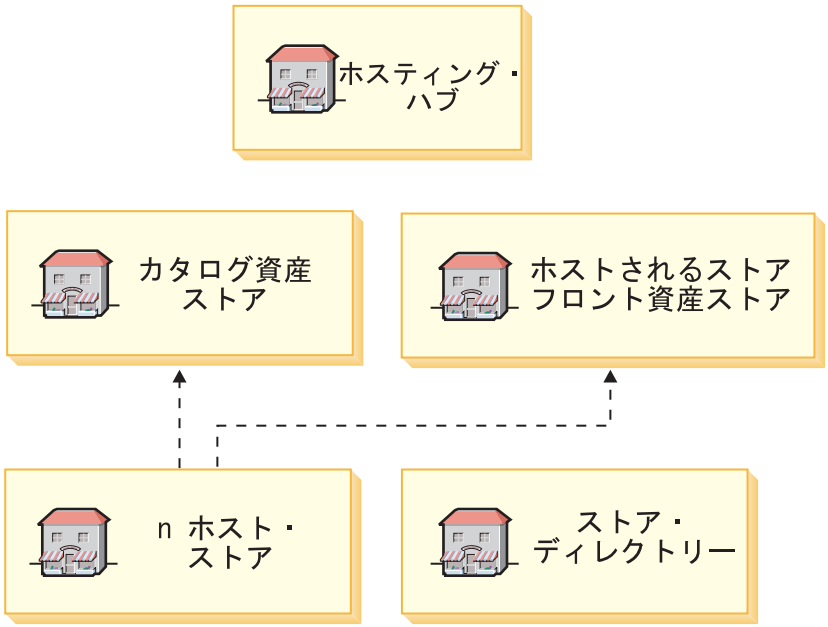
次のセクションでは、ストアがサンプル・ビジネスでインプリメントされる方法を示します。

注: 消費者向けおよび Business B2B 向けのサンプルには、それぞれ 1 つのダイレクト・セールス・ストアがあるので、ここでは扱いません。

ホスティング

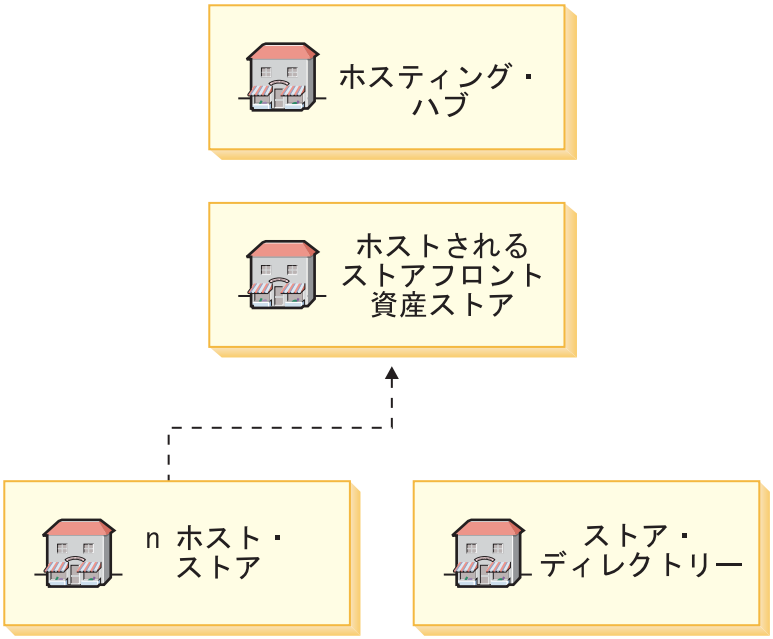
Business

次の図は、ホスティングのサンプルを構成するストアのタイプを示します。



ホスティング・サイトのサンプルには、1 つのハブ・ストア (ホスティング・ハブ)、 2 つの資産ストア (カタログ資産ストアとホストされるストアフロント資産ストア)、およびストア・ディレクトリーが入っています。ストア・ディレクトリーは、サイト内のすべてのホストされるストアのリストで、それらのストアに対してゲートウェイとしての役割を果たします。ホスティング・ストアは、2 つの資産ストアからの資産を使用して作成されます。

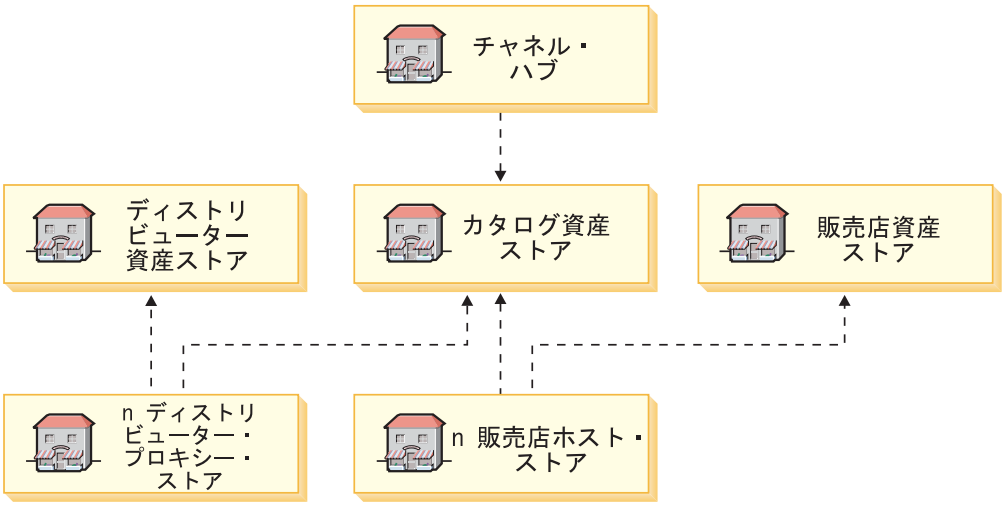
顧客が、カタログ資産ストアで定義されるカタログを使用するのではなく、独自のカタログ・データを作成することを選択できることに注意してください。このさまざまな変化によって、ホスティング・サイトの 2 番目のインプリメンテーションが作成されます。次の図でそれを示します。



デマンド・チェーン

Business

次の図は、デマンド・チェーンのサンプルを構成するストアのタイプを示します。

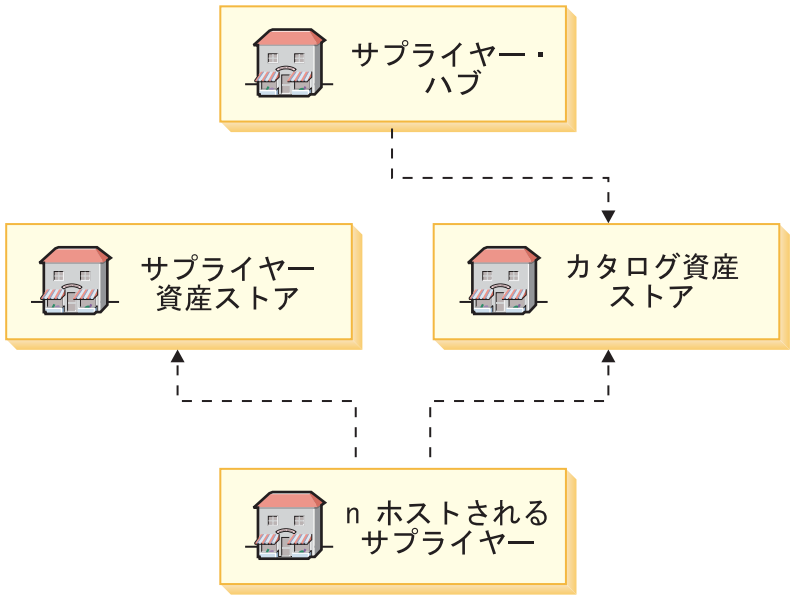


デマンド・チェーンのサンプル・サイトには、1 つのハブ・ストア (チャネル・ハブ)、および 3 つの資産ストア (ディストリビューター資産ストア、カタログ資産ストア、および販売店ストアフロント資産ストア) が入っています。チャネル・ハブが、カタログ資産ストアで定義されるカタログ資産を使用することに注意してください。ディストリビューター・プロキシー・ストアはディストリビューター資産ストアからの資産を使用して作成され、販売店のホストされるストアはカタログ資産および販売店のストアフロント資産ストアからの資産を使用して作成されます。

サプライ・チェーン

Business

次の図は、サプライ・チェーンのサンプルを構成するストアのタイプを示します。



サプライ・チェーンのサンプル・サイトには、1 つのハブ・ストア (サプライヤー・ハブ)、および 2 つの資産ストア (カタログ資産ストアおよびサプライヤー資産ストア) が入っています。サプライヤー・ハブが、カタログ資産ストアで定義される資産を使用することに注意してください。ホストされるサプライヤーは、カタログ資産ストアおよびサプライヤー資産ストアからの資産を使用して作成されます。

注: サプライヤー・ハブの所有者は、ホストされるサプライヤーがカタログ資産ストアで使用するカタログ分類法 (たとえば、カタログ構造や共用可能な商品およびアイテムなど) を定義します。

第 4 部 ストアフロントの開発

第 8 章 ストアフロントの開発

この章では、WebSphere Commerce ストアフロント・アーキテクチャーの概要を示します。これには、ストアの外側の部分となる Web 資産 (HTML ページ、JSP ファイル、スタイルシート、イメージ、グラフィックス、およびその他のマルチメディア・ファイル・タイプなど) が、顧客に対してどのように表示されるかが含まれます。

ストアフロント・アーキテクチャー

WebSphere Commerce では、*コマンド* と*ビュー* によって、ストアフロント内の Web 資産を顧客に表示します。

- *コマンド* は、特定のビジネス処理 (ショッピング・カートへの商品の追加、オーダーの処理、顧客の住所録の更新、特定の商品ページの表示など) を実行します。アクションが完了すると、コマンドはビューを戻します。
- *ビュー* はコマンドとユーザー・アクションの結果を表示します。つまり、ビューはストア・ページ (JSP ファイル) を顧客に表示します。ビューが JSP ファイルを呼び出すためには、JSP ファイル名がビュー・レジストリー (VIEWREG) テーブルのビューに登録されていなければなりません。対応する JSP ファイルは、WebSphere Commerce ストア Web アプリケーションの下にあるストアのサブディレクトリー (storedir) に、JSP ファイル名で保管されます。

コマンドとビューはどちらも URL を使用して呼び出されます。たとえば、顧客がサンプル・ストア内で**「ショッピング・カート」**をクリックすると、URL https://hostname/path/OrderItemDisplay? が呼び出されます。この URL は、WebSphere Commerce Server に渡されます。 WebSphere Commerce Server は OrderItemDisplay コマンドを呼び出し、ショッピング・カートのページが顧客に表示されます。

顧客がサンプル・ストア内で**「ヘルプ」**をクリックすると、URL https://hostname/path/HelpView? が呼び出されます。この URL は、WebSphere Commerce Server に渡されます。 WebSphere Commerce Server は HelpView を呼び出します。これは「ヘルプ」ページを戻します。

また、WebSphere Commerce Server では複数のコマンドを 1 つの URL にマップすることができます。これにより、オプションで、コマンドのインプリメンテーションをストアごとに独自のものにすることができます。

同様に、WebSphere Commerce Server は、複数の JSP ファイルを単一のビューにマップすることもできます。これにより、オプションで、各ストアはデバイス・タイプごとに異なる JSP ファイル名を登録することができます。

デフォルト・コマンドおよびデフォルト・ビュー

WebSphere Commerce には、ストアで使用できるデフォルトのコマンドと、デフォルトのビューが備わっています。これらのデフォルト・コマンドとデフォルト・ビューは、 `wcs.bootstrap.xml` ファイルにリストされています。ブートストラップ・ファイルは以下のディレクトリーに置かれています。

- *WC_installdir*/schema/xml

必要なコマンドまたはビューが提供されていない場合は、独自に作成することができます。コマンドとビューの作成の詳細については、「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」を参照してください。

ストア・ページの作成

ストアフロントを作成する上で最大のタスクは、実際のストア・ページを作成することです。ストア・ページの開発作業を開始する前に、以下の計画アクティビティーを完了する必要があります。

- 必要なストア・ページのリストの開発
- コマンドとビューの URL のリストの開発
- JSP ファイル名とビューの関連付け
- アクセス制御ポリシーのリストの開発。詳細については、327 ページの『第 33 章 ストアでのアクセス制御』を参照してください。

注: ストア・ページの計画中に、キャッシング戦略を作成することも必要です。キャッシングの詳細については、95 ページの『第 9 章 ストア・ページのキャッシング』を参照してください。

ストア・ページのリストの開発

ストアを作成するために必要なページのリストを開発するには、ストアのビジネス要件や機能要件、および定義されているビジネス・プロセスのすべてを知っていなければなりません。

ユース・ケースからの作業

多くの人々は、要件をユース・ケースの形式で収集します。ユース・ケースは、ストアのビジネス・プロセスを、顧客と提案システムの間の相互作用の形で定義します。オンライン・ストアの場合、顧客がストアに登録したり、カタログをブラウズしたり、アイテムをオーダーしたりする方法を、ユース・ケースで定義できるかもしれません。

オンライン・ヘルプに、サンプル・ストアのビジネス・プロセスを詳述している一連のユース・ケースが収められています。これらのユース・ケースは、サンプル・ストアのフローをより良く理解するのに役立ちますし、独自のストアのユース・ケースを作成するためのガイドとしても使用できます。

以下に、登録ユース・ケースの例を挙げます。

登録ユース・ケース: 登録処理により顧客はデータベースに個人情報を入力することができます。

実行者:

- 顧客

メイン・フロー: 顧客はサイドバーから**「登録」**を選択します。次に、システムは以下のフィールドのあるページを表示します。

- E メール

- パスワード
- 確認パスワード
- 名
- 姓
- 年齢 (オプション)
- 性別 (オプション)

顧客は上記フィールドに該当する情報を入力し、**「送信」**を選択します。システムはシステムに新規顧客を作成し、顧客の情報を保存します (E1、E2、E3)。システムは顧客に、個人アカウント管理のユース・ケースの手順に従い、アカウントを管理するよう促すプロンプトを出します。

代替フロー: なし

例外フロー: E1: E メール・アドレスがすでに存在する場合

- E メール・アドレスがすでにシステムに存在する場合、システムは、ユーザーに別の E メール・アドレスの入力を求めるエラー・メッセージを表示します。そしてそのユース・ケースを最初から再開します。

E2: 必須フィールドが抜けている場合

- 以下のフィールド (E メール、パスワード、確認パスワード、名、姓) の 1 つでも指定されていない場合、システムはエラー・メッセージを表示します。そしてそのユース・ケースを最初から再開します。

E3: パスワードが無効な場合

- パスワードが無効であるか確認パスワードと一致しない場合、システムは警告を表示します。

ストア・ショッピング・フローの決定: ユース・ケースを開発してストアのビジネス・プロセスを示すか、別の方法を使用するかにかかわりなく、いったんビジネス・プロセスが使用可能になったら、ストアのショッピング・フローを作成することができます。

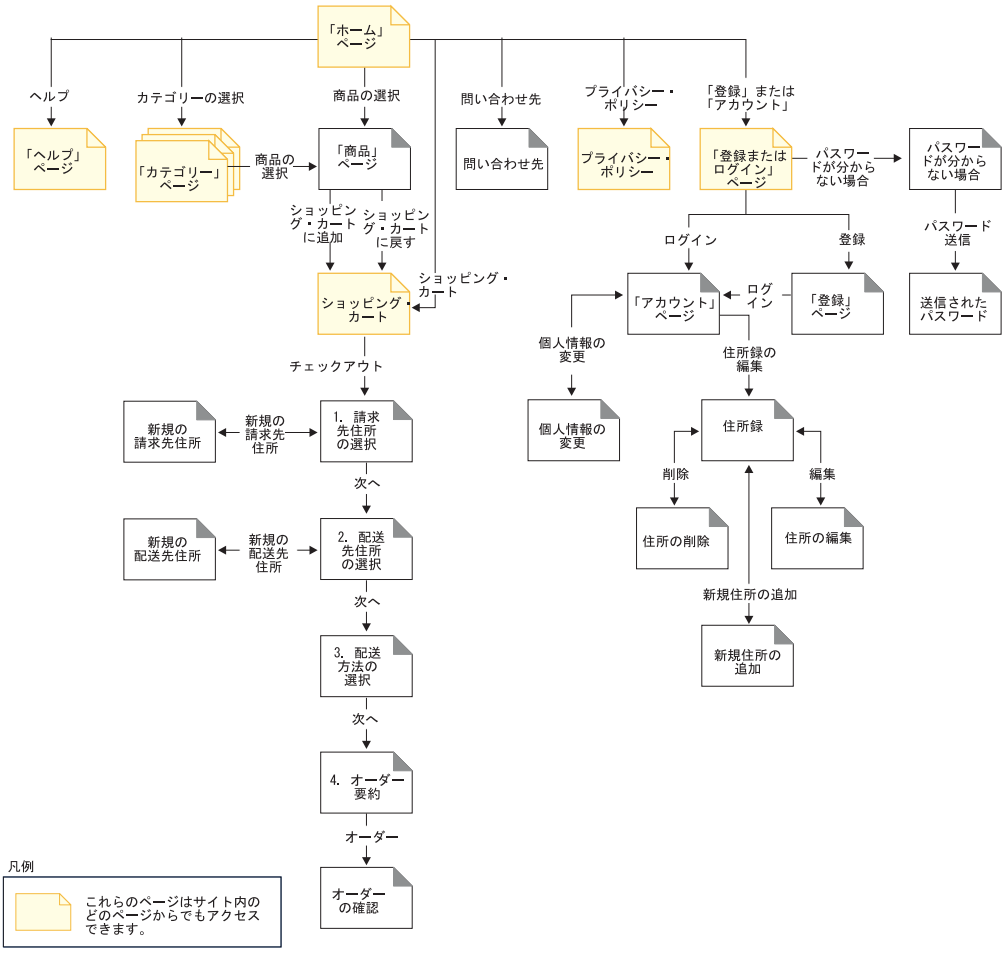
注: ユース・ケースにはしばしば、「顧客が**「送信」**を選択すると、「オーダー」ページが表示されます」などといったフロー情報が含まれているため、ショッピング・フロー・ダイアグラムを作成する上で役立つ情報となることがあります。

ショッピング・フローは、顧客がストアの中をどのように移動するかを示すにあたり、ストアに定義されている要件やビジネス・プロセスを反映します。たとえば、ホーム・ページからサイトに入ってきた顧客に対し、カタログをブラウズする前に登録を済ませるよう促すこともできますし、登録しなくてもゲストとしてカタログを見られるようにすることもできます。また、顧客が「クイック・チェックアウト」できるショッピング・フローもあれば、購入のたびにすべてのチェックアウト・ステップを完了することの必要なショッピング・フローもあります。あるいは、顧客が両方のチェックアウトから選択できるようなショッピング・フローもありえます。

ストアのフロー・ダイアグラムが完成したかどうかを確かめるため、ストアのユース・ケースの全ステップをストアのフロー・ダイアグラムに表示するようにしてください。

FashionFlow サンプル・ストアのショッピング・フローに関する以下のダイアグラムのように、ショッピング・フローを視覚的にマップすると、顧客がストアをどのように移動するかを見ることができます。

注: この図に含まれているのは、FashionFlow ストア・フローの一部だけです。フローの全体については、「*WebSphere Commerce サンプル・ストア・ガイド*」を参照してください。



FashionFlow のショッピング・フローのダイアグラムは非常に簡潔です。ストアの中の顧客の移動を示すメイン・フローはありますが、エラー・シナリオは何もありません。たとえば、間違ったパスワードを使用して顧客がログインした場合、あるいは間違ったクレジット・カード番号を入力した場合は、どうなるのでしょうか。しかし、このような簡潔なダイアグラムでも、ストアに必要なページのリストを開発することが可能です。まず、ショッピング・フローのダイアグラムにリストされているすべてのページのために、ビューを作成する必要があります。

たとえば、FashionFlow の図と同じショッピング・フローを使用してストアを作成する場合、以下のページを作成する必要があるでしょう。

注: 以下の表には、FashionFlow ストアで使用されるビュー名がリストされています。

FashionFlow ショッピング・フロー・ダイアグラムのページ (顧客の側から見たもの)	対応するビュー
「ホーム」ページ	StoreCatalogDisplayView
「ヘルプ」ページ	HelpView
「問い合わせ先」	ContactView
「プライバシー・ポリシー」	PrivacyView
「登録またはログイン」ページ	LogonForm
「パスワードが分からない場合」	LogoffView
「送信されたパスワード」	ResetPasswordForm
「アカウント」ページ	LogonForm
「個人情報の変更」	UserRegistrationForm
「住所録」	AddressBookForm
「新規住所の追加」	AddressForm
「住所の削除」	AddressBookForm
「住所の編集」	AddressForm
「登録」ページ	UserRegistrationForm
「ショッピング・カート」	OrderItemDisplayViewShiptoAssoc
「請求先住所の選択」	BillingAddressView
「新規の請求先住所」	AddressForm
「配送先住所の選択」	MultipleShippingAddressView
「新規の配送先住所」	AddressForm
「配送方法の選択」	MultipleShippingMethodView
「オーダー要約」	AllocationCheck
「オーダーの確認」	OrderOKView

注: FashionFlow で使用されるビューの多くは、特に FashionFlow 用に作成されたものです。これらのビューは、FashionFlow ストア・アーカイブにある `command.xml` ファイルにリストされています。詳細については、159 ページの『WebSphere Commerce でのコマンド、ビュー、および URL の登録』を参照してください。

上記の表は、作成する必要のある一連の基本的なページを暗に示しているにすぎません。他に作成する必要のあるページを決めるためには、ユース・ケース、またはビジネス・プロセスを定義するために使用する別の方法を詳しく調べてみてください。

エラー・ページ: ユース・ケースの中の例外フローも、ストアで作成する必要のあるエラー・ページを決めるのに役立ちます。 FashionFlow の登録ユース・ケースでは、以下の例外フローが指定されています。

- E メール・アドレスがすでに存在する: E メール・アドレスがすでにシステムに存在する場合、システムは、ユーザーに別の E メール・アドレスの入力を求めるエラー・メッセージを表示します。そしてそのユース・ケースを最初から再開します。
- 必須フィールドが欠落している: 以下のフィールド (E メール、パスワード、確認パスワード、名、姓) の 1 つでも指定されていない場合、システムはエラー・メッセージを表示します。そしてそのユース・ケースを最初から再開します。
- パスワードが無効である: パスワードが確認パスワードと一致しない場合、システムは警告を表示します。

結果として、それぞれの例外フローごとに、エラー・ページまたはエラー・メッセージを作成する必要が生じます。

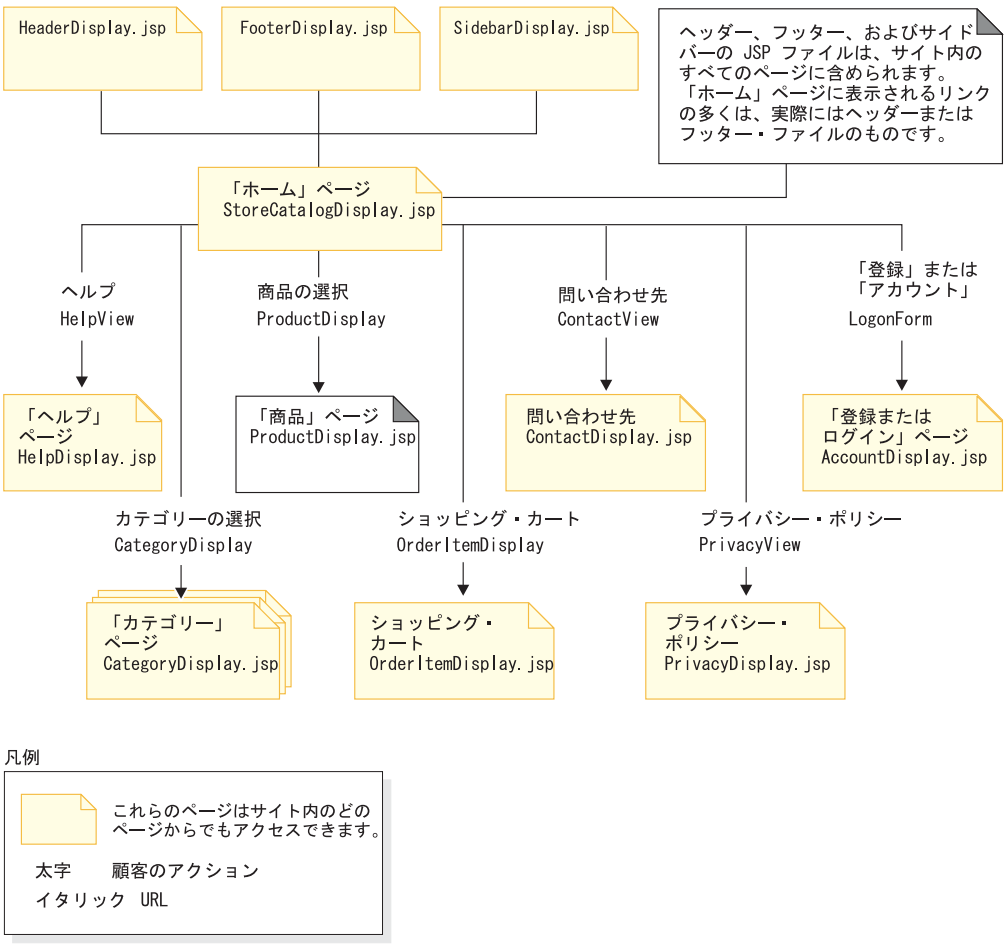
コマンドとビューの URL のリストの開発

FashionFlow のショッピング・フロー・ダイアグラムで示したように、チェックアウトや登録などのビジネス・プロセスは、いくつかのページを必要とする場合があります。これらのページを単にページの集合にとどめず、組み合わせて作業ビジネス・プロセスまたはフローとするためには、ページにコマンドまたはビューを組み込む必要があります。

必要な URL のリストの開発

ストアを作成するために必要なページのリストを開発したのとまったく同様、ストアのビジネス・プロセスをインプリメントするために必要なコマンドとビューの URL のリストを開発することも必要です。ストアのショッピング・フロー・ダイアグラムとデフォルトのコマンドとビューのリストを使用して、各アクションを完了するために必要な URL を識別します。

サンプル・ストアで使用されているコマンドとビューの URL を理解しておくことも、ストアで必要な URL を判別する助けになります。以下の図は、FashionFlow のショッピング・フロー・ダイアグラムにあるいくつかのアクションの URL を示しています。詳細については、「*WebSphere Commerce サンプル・ストア・ガイド*」にある『サンプル・ストアの情報』を参照してください。



JSP ファイル名とビューの関連付け

WebSphere Commerce Server は、要求に対する応答としてビューを構成するために、ビュー・コマンドを使用します。 WebSphere Commerce Server には、以下のビュー・コマンドが用意されています。

- `HttpForwardViewCommandImpl`: このビュー・コマンドは、ビュー要求を JSP ファイルに転送します。
- `HttpRedirectViewCommandImpl`: このビュー・コマンドは、ビュー要求を別の URL にリダイレクトします。
- `HttpDirectViewCommandImpl`: このタイプのビュー・コマンドは、応答ビューを直接クライアントに送信します。 このコマンドは JSP ファイルを呼び出しません。直接ビューでは、ビュー・コマンドによってではなくコントローラー・コマンドによって、出力応答が作成されるようになります。

JSP ファイルを直接レンダリングするには、 `HttpForwardViewCommandImpl` ビュー・コマンドを使用します。たとえば、FashionFlow で使用される URL を示すダイアグラムの場合、「ヘルプ」ページ (`Help.jsp`) を表示するには、HelpView をビューに登録し、 `Help.jsp` および `HttpForwardViewCommandImpl` コマンドと関連付けます。これを次の例に示します。

```
<viewreg
viewname="HelpView"
devicefmt_id="-1"
storeent_id="@storeent_id_1"
interfacename="com.ibm.commerce.command.ForwardViewCommand"
classname="com.ibm.commerce.command.HttpForwardViewCommandImpl"
properties="docname=Help.jsp"
internal="0"
https="0"
/>
```

インターフェースおよびインプリメンテーション・クラスに完全修飾クラス名が使用されていることに注意してください。

表示コマンドから戻されるビューをレンダリングするには、HttpForwardViewCommandImpl ビュー・コマンドを使用します。表示コマンドはデータベースからデータを読み取りますが、そのデータに対して変更を加えることはありません。たとえば、FashionFlow で使用される URL を示すダイアグラムの場合、OrderItemDisplay コマンドは OrderItemDisplayViewShiptoAssoc ビューを戻します。このビューがビュー・レジストリーに登録されたときに、 OrderItemDisplay.jsp と HttpForwardViewCommandImpl がそのビューに関連付けられました。これを次の例に示します。

```
<viewreg
viewname="OrderItemDisplayViewShiptoAssoc"
devicefmt_id="-1"
storeent_id="@storeent_id_1"
interfacename="com.ibm.commerce.command.ForwardViewCommand"
classname="com.ibm.commerce.command.HttpForwardViewCommandImpl"
properties="docname=OrderItemDisplay.jsp"
internal="0"
https="0"
/>
```

使用するあらゆる表示コマンド (たとえば OrderItemDisplay) に関連したあらゆるビューについて、 JSP ファイル名を関連付ける必要があります。ビューに JSP ファイル名を関連付けることについては、 159 ページの『WebSphere Commerce でのコマンド、ビュー、および URL の登録』を参照してください。

注: 商品の表示コマンドおよびカテゴリーの表示コマンドは、ビューとともに JSP ファイル名を戻します。商品およびカテゴリーを表示するこれらの JSP ファイル名は、カタログ・データに保管されます。詳細については、 187 ページの『ストア・カタログ資産の表示』を参照してください。オプションで、商品およびカテゴリーを表示するために割り当てる JSP ファイル名を、ストアでサポートされるメンバー・グループまたは言語ごとに変えることができます。

HttpRedirectViewCommandImpl ビュー・コマンドを使用して、非表示コマンド (データベースを変更するコマンド) の出力を出すことができます。非表示コマンドは表示に関連付けて、顧客が不作為にページを再ロードしたり「戻る (Back)」ボタンを押したときにコマンドが再実行されることを防止する必要があります。

表示コマンドにリダイレクトするためには、非表示コマンドの URL に &URL= パラメーターを使用して表示コマンドを指定します。たとえば、FashionFlow サンプル・ストアの Address フォームに住所情報を追加し、**「送信」**をクリックすると、AddressAdd コマンドが呼び出されます。 AddressAdd コマンドを呼び出すために使用する URL は、 &URL= パラメーターに AddressBookForm コマンドを指定しま

す。その結果、AddressBookForm ディスプレイ・コマンドへリダイレクトされ、AddressBookForm ビューへ戻ります。 AddressBookForm ビューがビュー・レジストリーに登録されたときに、 `AddressBookForm.jsp` と `HttpForwardViewCommandImpl` がそのビューに関連付けられました。

すべての非表示コマンドについては、 `URL=parameter` テクニックを使用してください。非表示コマンドとは、データベース内のデータに変更を加えるコマンドです。

第 9 章 ストア・ページのキャッシング

ストアフロントの開発時には、ストア・ページをキャッシュする方法について決めることも必要です。この章では、ストアに合わせたキャッシング・ストラテジーの作成とインプリメントを説明します。

キャッシング・ストラテジーの計画

キャッシング・ストラテジーの決定時には、まず以下のことを考慮する必要があります。

- キャッシュするページ
- ページ全体をキャッシュするかまたはページのフラグメントをキャッシュするか

キャッシュするページ

高水準のキャッシング・ストラテジーの作成時には、まずストアのどのページをキャッシュするかを決める必要があります。キャッシングの適切な候補となるページは、頻繁にアクセスされ、かつ一定期間安定しており、さまざまなユーザーが再利用できるコンテンツを含むページです。たとえば多くの場合、カタログ表示ページはキャッシングを使用可能にするページの候補となります。

ページ全体をキャッシュするかまたはページのフラグメントをキャッシュするか

バージョン 5.5 では、WebSphere Commerce は WebSphere Application Server 動的キャッシュ・サービスを使用します。このサービスにより、WebSphere Commerce は Web ページ全体のキャッシング、およびページのフラグメントのキャッシングの両方をサポートできます。 Web ページ全体のキャッシングでは、そのページがいくつかの小さいフラグメントから構成されていても、単純にページ全体を 1 つのエンティティーとしてキャッシュします。ページのフラグメントには、独立したヘッダー、サイドバー、またはフッターなどがあります。ページの本体をいくつかのフラグメントに分割することもできます。たとえば、本体ページ上の 1 つのフラグメントが商品を示し、2 番目のフラグメントが価格を示す場合があります。ページのフラグメント化により、コンテンツを個々のユーザーに個別設定して表示できます。 WebSphere Commerce が提供するストア・ページのサンプルは、いくつかのフラグメント (ヘッダー、サイドバー、フッター、本文) から構成されます。

ストア・ページがフラグメントから構成されている場合、フラグメント単位でページをキャッシングすることもできます。個々のフラグメントをキャッシングすることにより、幅広い利用者に再使用可能なページの部分をキャッシュできます。あるページに利用者のごく一部だけを対象とした個別設定情報が含まれている場合、ページ全体をキャッシュしても、キャッシュされたそのページを再使用できるのは利用者のごく一部だけなので、そのページが頻繁に再使用されることはありません。たとえば、ヘッダーに含まれる各顧客にウェルカム・メッセージを表示するページをユーザー ID に基づいてキャッシュした場合、キャッシュされたそのページを再使用できるのはその特定のユーザーだけになります。しかし、そのページをフラグ

メントに分解すると、ほとんどの利用者に再利用できるフラグメントをキャッシュできます。たとえば、フッター、サイドバー、および商品表示フラグメントがすべてのユーザーに該当するものにして、価格およびヘッダーのフラグメントは個別設定にできるでしょう。

ページが要求されると、個々のフラグメントは再アセンブルされ、ページが生成されます。

ストア・ページは、ページ全体のキャッシングまたはページのフラグメントのキャッシング、もしくはそれら 2 つの方法の組み合わせを使用してキャッシュすることができます。

詳細なキャッシング・ストラテジーの開発

どのページ、およびページ・フラグメントをキャッシュするかを決定した後、より詳細なキャッシング・ストラテジーを決定することが必要です。キャッシュする予定の各ページまたはフラグメントごとに、以下のことを決定する必要があります。

- ページまたはフラグメントを要求する方法
- ページまたはフラグメントがストア関係に依存するかどうか
- キャッシュされるデータを無効にする方法

ページまたはフラグメントを要求する方法

JSP ファイルを (単一ページであるか、ページ・フラグメントであるかに関係なく) 要求する方法によって、 WebSphere Application Server がそのファイルをキャッシュする方法が決まります。たとえば、WebSphere Application Server は、JSP ファイルがサーブレット、オブジェクト、EJB、またはコマンドとして表示されるかどうかを把握する必要があります。このため、キャッシュする予定の各ページまたはフラグメントを要求する方法のリストをコンパイルする必要があります。

ページまたはフラグメントがストア関係に依存するかどうか

Business 151 ページの『第 14 章 ストア間の関係』および 71 ページの『第 7 章 ストア・アーキテクチャー』で説明したとおり、ストアには、別のストアからのデータを使用するために他のストアとの関係がなければなりません。たとえば、ストア A がストア B で定義されたカタログ・データを使用する場合があります。またストアには、複数の異なるソースからのデータを使用できるようにするために、複数のストアとの関係を持つ場合もあります。詳細なキャッシング計画の一部として、各ページまたはフラグメントで表示されるデータが別のストアとの関係に依存するかどうかを決定することが必要です。ページが別のストアからの情報を表示する場合、他のストアからのデータが更新されるたびに、キャッシュされるページも更新される必要があります。ストア関係のキャッシングの詳細については、 103 ページの『ストア関係を使用するストア・ページのキャッシングのインプリメント』を参照してください。

キャッシュされるデータを無効にする方法

キャッシュする予定の各ページまたはページ・フラグメントごとに、キャッシュされるページまたはフラグメントがいつ無効になり対応するキャッシュ・エントリーをキャッシュから除去するのかを決定する必要もあります。このプロセスは、無効

化として知られています。キャッシュされるページが変更されて有効ではなくなるときを決定するには、キャッシュされるページの期限が切れる理由を決定することが必要です。たとえば、キャッシュされるショッピング・カート・ページは、顧客が新しいアイテムをカートに追加すると無効になります。また、キャッシュされるページは、管理者が WebSphere Commerce アクセラレーターを使用してストアを更新するときや、新規カタログ・データがローダー・パッケージまたは WebSphere Commerce アクセラレーターのツールで追加されるときにも無効にされることがあります。

キャッシュされるページまたはフラグメントを無効にするすべての可能な方法のリストをコンパイルした後に、無効化を実行するために使用するイベントを決める必要があります。無効化を生じさせるイベントには、サーブレット要求、コントローラー・コマンド、タスク・コマンド、などがあります。たとえば、WebSphere Commerce アクセラレーターで商品管理ツールを使用して商品説明を更新すると、WebSphere Commerce は内部的にコマンド AddCatalogEntryDescCmd または UpdateCatalogEntryDescCmd を呼び出して、変更を加えます。これらのコマンドによって変更されたキャッシュ済みページを無効にしたい場合、コマンドの実行を代行受信して無効化を起動する無効化ポリシーを cachespec.xml ファイルに追加する必要があります。無効化のインプリメントの詳細については、以下を参照してください。

- 「*WebSphere Commerce 管理ガイド*」の『動的キャッシング』の章には、新規の無効化ポリシーの設定方法およびキャッシュ無効化の例が示されています。
- 102 ページの『cachespec.xml ファイルでのキャッシュされるデータの無効化』には、WebSphere Commerce に備わっているサンプルの無効化ポリシーをストアの cachespec.xml ファイルにマージする方法が示されています。

キャッシング・ストラテジーのインプリメント

キャッシング・ストラテジーに必要なすべての詳細情報を収集した後に、キャッシュ・ポリシー・ファイルを作成することによって、キャッシング・ストラテジーをインプリメントします。このファイルでは、キャッシュの対象とキャッシュの方法、およびキャッシュされるページを無効化する方法を含め、収集した情報を定義します。 WebSphere Application Server 動的キャッシュ・サービスは、cachespec.xml と呼ばれるこのキャッシュ・ポリシー・ファイルを使用して、ストアにキャッシングをインプリメントします。

WebSphere Commerce が提供する各ストアのサンプルには、そのストアのキャッシング・ストラテジーを定義する cachespec.xml ファイルが含まれています。これらのファイルは以下のディレクトリーにあります。

WC_installdir/samples/dynacache/*BusinessModel*

ストアがサンプルに基づく場合は、これらのファイルを変更するオプション、またはストアに合わせて cachespec.xml ファイルのベースとしてこれらのファイルの 1 つを使用するオプションがあります。

cachespec.xml ファイルについて

WebSphere Commerce のストア・ページをキャッシュするには、キャッシュ可能オブジェクトを cachespec.xml ファイルに定義する必要があります。 WebSphere Commerce は、cachespec.xml ファイルに定義されたエレメントのサブセットだけを使用します。エレメントのサブセットについては、このセクションで説明されています。 cachespec.xml ファイルの詳細については、 WebSphere Application Server Information Center (http://www.ibm.com/software/webservers/appserv/infocenter.html) のトピック『Cachespec.xml file』を参照してください。詳細については、「*WebSphere Commerce 管理ガイド*」の『動的キャッシング』の章を参照してください。

WebSphere Commerce が使用するエレメントについて

WebSphere Commerce では、cachespec.xml の次のエレメントを使用しました。

- クラス
- 名前
- プロパティー

これらの 4 つのエレメントの使用を、以下の例で示します。

```
<cache-entry>
   <class>servlet</class>
       <name>/FashionFlow/ShoppingArea/CatalogSection/CategorySubsection
/StoreCatalogDisplay.jsp</name>
        <property name="save-attributes">false</property>
```

クラス: クラス・エレメントは必須エレメントです。これは、WebSphere Application Server がその他のキャッシュ・ポリシー定義を解釈する方法を決定します。 WebSphere Commerce は以下のクラス値を使用します。

- コマンド
- サーブレット

値コマンドは、WebSphere Commerce プログラミング・モデルを使用するクラスを参照します。

値サーブレットは、WebSphere Application Server サーブレット・エンジンで展開されたサーブレットまたは JSP ファイルを参照します。

注: WebSphere Commerce バージョン 5.5 では、コマンドの無効化だけがサポートされています。

名前: 名前は、サーブレットまたはコマンドの完全修飾クラス名です。名前は必須エレメントです。

コマンドの名前値は、パッケージ名を含む必要があります。たとえば、com.ibm.commerce.dynacache.commands.MemberGroupsCacheCmdImpl となります。

サーブレットおよび JSP ファイルの名前値には、キャッシュされる JSP ファイルまたはサーブレットの完全な URI を組み込むことが必要です。たとえば、com.ibm.commerce.server.RequestServlet.class/ToolTech/ShoppingArea/CatalogSection/CategorySubsection/StoreCatalogDisplay.jsp となります。

プロパティー: プロパティー・エレメントは次のような形式になります。`<property name=`*key*`>`*value*`</property>`。ここで、*key* は定義されているプロパティーの名前、*value* は対応する値です。キャッシュ可能オブジェクトにオプションのプロパティーを設定することもできます。たとえば、`<property name="consume-subfragments">true</property>` などとなります。

WebSphere Commerce ストア・ページをキャッシュするときには、以下のプロパティーが使用されます。

プロパティー	値	有効クラス	説明
EdgeCacheable	真または偽。デフォルトは偽です。	サーブレット	プロパティーが真である場合、指定されたサーブレットまたは JSP ファイルは、Edge Server から外部的に要求されます。サーブレットまたは JSP ファイルがキャッシュ可能かどうかは、キャッシュ指定の残りに依存します。

消費サブフラグメント	真または偽。デフォルトは偽です。	サーブレット	サーブレットのキャッシュ時に、そのサーブレットのコンテンツだけが保管されます。そのコンテンツを入れるまたは転送する先の、他のフラグメントのプレースホルダーが作成されます。消費サブフラグメント (CSF) は、組み込みによって子サーブレットを検出したときに、キャッシュがコンテンツの保管し続けるように指示します。親エントリー (CSF とマークされたエントリー) は、そのキャッシュ・エントリーにすべてのフラグメントからのすべてのコンテンツを入れ、その結果として組み込みや転送はないが、エントリーのツリー全体からのコンテンツを持つ 1 つの大きなキャッシュ・エントリーが作成されます。このメソッドによって、アプリケーション・サーバー処理のかなりの部分を省くことができますが、これが役立つのは、組み込まれるフラグメントのツリー全体を判別するために必要なすべての情報が、外部 HTTP 要求に入っている場合だけです。
保管属性	真または偽。デフォルトは真です。	サーブレット	保管属性が偽に設定されると、依頼の属性はキャッシュ・エントリーでは保管されません。

ストア cookie	真または偽。デフォルトは真です。	サーブレット	ストア cookie が偽に設定されると、要求 cookie はキャッシュ・エントリーで保管されません。

デフォルトでは、DynaCache は cookie (サーブレット・クラスによってキャッシングするとき) とすべての要求属性 (サーブレットと JSP) を、キャッシュ・エントリーと共にキャッシュします。しかし、WebSphere Commerce の cookie と要求属性には、キャッシュすべきではないユーザー特定の情報が含まれています。その結果、ページ全体をキャッシュするときには、以下のプロパティー名および値が必須となります。

```
<property name="save-attributes">false</property>
<property name="store-cookies">false</property>
```

以下のプロパティー名および値は、JSP ファイルに定義されたすべてのキャッシュ・エントリーに必須です。

```
<property name="save-attributes">false</property>
```

キャッシュ ID 規則について

キャッシュ ID は、キャッシュ・エントリーを一意的に識別します。 WebSphere Application Server がオブジェクトをキャッシュするためには、そのオブジェクトのさまざまな呼び出しの固有の ID を生成する方法を把握している必要があります。これらの ID は、ユーザー作成のカスタム Java コード、またはキャッシュ・エントリーのキャッシュ・ポリシーで定義される規則から作成されます。

`cachespec.xml` ファイルでは、キャッシュ ID エレメントは、ID を生成するための規則を定義します。各キャッシュ・エントリーには、規則が空でないキャッシュ ID を戻すか実行する規則がなくなるまで、定義された順序で実行される、複数のキャッシュ ID 規則を持つ場合があります。キャッシュ ID 生成の規則が、どれも有効なキャッシュ ID を作成しない場合、オブジェクトはキャッシュされません。

これらの ID は次のいずれかの方法で開発されます。

- キャッシュ・エントリーのキャッシュ・ポリシーで定義されたコンポーネント・エレメントを使用する
- カスタム Java コードを作成し、入力変数およびシステム状態から ID を作成する

従属 ID 規則について

従属 ID エレメントは、複数のキャッシュ・エントリーを同じグループ ID に関連付ける、追加のキャッシュ・グループ ID を指定します。従属 ID は、従属 ID の基本ストリングを、そのコンポーネント・エレメントが戻す値と連結することによって生成されます。必要なコンポーネントがヌル値を戻す場合、従属 ID 全体が生成されず、それは使用されません。

従属 ID を明示的に検証するには、WebSphere Dynamic Cache API を使用するか、または別のキャッシュ・エントリー無効化エレメントを使います。複数の従属 ID 規則は、キャッシュ・エントリーごとに存在させることができます。すべての従属

ID 規則は個別に実行します。従属 ID 規則を定義する方法の詳細については、「*WebSphere Commerce 管理ガイド*」の『動的キャッシング』の章を参照してください。

無効化規則について

無効化規則は、従属 ID とまったく同じ方法で定義できます。ただし、無効化規則により生成される ID は、同じ従属 ID を持つキャッシュ・エントリーを無効化するのに使用されます。無効化 ID は、無効化 ID の基本ストリングを、そのコンポーネント・エレメントが戻す値と連結することによって生成されます。必要なコンポーネントがヌル値を戻す場合、無効化 ID 全体が生成されず、無効化は実行されません。複数の無効化規則は、キャッシュ・エントリーごとに存在させることができます。すべての無効化規則は個別に実行します。無効化規則を定義する方法の詳細については、「*WebSphere Commerce 管理ガイド*」を参照してください。

cachespec.xml ファイルでのキャッシュされるデータの無効化

デフォルトでは、ストア・アーカイブのサンプルと共に配送される `cachespec.xml` ファイルには、無効化ポリシーが含まれていません。サンプル・ストア内またはサンプルに基づくストア内で DynaCache を使用してキャッシュの無効化を自動化したい場合、無効化ポリシーをストアの `cachespec.xml` ファイルに追加する必要があります。以下のディレクトリーにあるいくつかの `cachespec.xml` ファイルでは、サンプルの無効化ポリシーが提供されています。

WC_installdir/samples/dynacache/invalidation

このディレクトリーには、カタログ、ショッピング・カート、ストアなど、機能エリア用の個別の `cachespec.xml` ファイルが入っています。各ファイルには、その特定のエリアの無効化ポリシーが含まれます。

ストアでカタログ・ページをキャッシュする予定の場合は、以下のファイルからストアに無効化ポリシーを追加することが必要です。

- *WC_installdir*/samples/dynacache/invalidation/catalog/cachespec.xml
- *WC_installdir*/samples/dynacache/invalidation/membergroup/cachespec.xml

  注: これらのメンバー・グループ無効化規則では、追加の従属 ID をキャッシュ・エントリーに追加する必要があります。詳細については、この `cachespec.xml` ファイルの内容を参照してください。
- *WC_installdir*/samples/dynacache/invalidation/store/cachespec.xml

ストアの cachespec.xml ファイルへの無効化ポリシーのサンプルの追加

サンプルの無効化ファイルで提供される無効化ポリシーをストアに追加するには、以下のようにします。

1. ストアの `cachespec.xml` ファイルをオープンします。
   - *WAS_installdir*/installedApps/*cell_name*/*WC_instanceName.ear*/Stores.war/WEB-INF directory

   ストアにキャッシング・ポリシーが定義されていない場合で、そのストアが WebSphere Commerce に備わっているサンプルに基づいている場合には、以下のディレクトリーにあるサンプルの `cachespec.xml` ファイルを使用できます。

- *WC_installdir*/samples/dynacache/*BusinessModel*

2. 無効化 cachespec.xml ファイルのサンプルをオープンします。無効化 cachespec.xml ファイルのサンプルは、以下のディレクトリーにあります。
   - *WC_installdir*/samples/dynacache/invalidation
3. 無効化ファイルのサンプルからストアの cachespec.xml ファイルに、無効化ポリシーをコピーします。その無効化ポリシーを、ストアの cachespec.xml ファイルの末尾にある最後のエレメントの後に置きます。
4. 無効化 ID が、キャッシング・ポリシーの対応する従属 ID に一致することを確認します。一致する従属 ID が存在しない場合、無効化ポリシーは実行されないので、無効化規則の ID または従属 ID 規則の ID を変更してそれらが一致するようにする必要があります。

   注: ストアに追加のまたは異なるビジネス要件があれば、追加の無効化ポリシーおよび従属 ID を追加する必要があります。
5. 必要なら、サンプルの無効化ファイルからコピーしたセクション中の JSP ファイルの名前とディレクトリーを、ストアの cachespec.xml ファイルの残りの情報に一致するように変更します。
6. ファイルを保管します。

ストア関係を使用するストア・ページのキャッシングのインプリメント

Business ストアが、ストア関係を介して別のストアで定義されるデータを使用している場合、キャッシュ・フィルターによって指定された依頼の属性を使用してその関係を定義することが必要です。キャッシュ・フィルターとは、サーブレット・フィルターで、セッションおよびストア関係情報から WebSphere Application Server DynaCache で使用可能な依頼の属性を定義します。その後 Dynacache は、この情報を使用してキャッシュ無効化に使用するキャッシュ ID および従属 ID を構成します。セッション情報のためにセットアップされる依頼の属性のリストについては、「*WebSphere Commerce 管理ガイド*」の『動的キャッシング』の章を参照してください。

キャッシュ・フィルターは、StoreAccessBean から getStorePath() および getStoresForRelatedStore() メソッドを呼び出すことによってストア関係を作成します。対応する情報は、以下の表にリストされています。

表 3.

ストア関係タイプ	ストア関係 ID	getStorePath() の依頼の属性名	getStoresForRelatedStore() の依頼の属性名
IBM commerce businessPolicy	-1	DC_busN	DC_bus_RS_N
IBM commerce ビジネス・キャンペーン	-3	DC_campN	DC_camp_RS_N
IBM commerce ビジネス・カタログ	-4	DC_catN	DC_cat_RS_N

表 3. (続き)

IBM commerce ビジネス・コマンド	-5	DC_cmdN	DC_cmd_RS_N
IBM commerce のホストされるストア	-6	DC_hostN	DC_host_RS_N
IBM Commerce 価格	-7	DC_prcN	DC_prc_RS_N
IBM Commerce 参照	-8	DC_refN	DC_ref_RS_N
IBM Commerce セグメンテーション	-9	DC_segN	DC_seg_RS_N
IBM Commerce URL	-10	DC_urlN	DC_url_RS_N
IBM Commerce ビュー	-11	DC_viewN	DC_view_RS_N
IBM Commerce 在庫	-13	DC_invN	DC_inv_RS_N
IBM commerce 基本アイテム	-14	DC_baseItemN	DC_baseItem_RS_N
IBM commerce チャネル・ストア	-15	DC_chsN	DC_chs_RS_N
IBM commerce 通貨変換	-17	DC_currConvN	DC_currConv_RS_N
IBM commerce 通貨形式	-18	DC_currFmtN	DC_currFmt_RS_N
IBM commerce のサポートされる通貨	-19	DC_supCurrN	DC_supCurr_RS_N
IBM commerce カウンター値通貨	-20	DC_cterCurrN	DC_cterCurr_RS_N
IBM commerce 計算形式	-21	DC_meaFmtN	DC_meaFmt_RS_X

注: DynaCache は依頼の属性の配列をサポートしないので、キャッシュ・フィルターは、複数のストア ID が戻されると、複数の依頼の属性をセットアップします。たとえば、getStorePath() が配列 [10051,10002] をリソース ID -4 (IBM commerce ビジネス・カタログ) に対して戻す場合、依頼の属性のセットアップは以下のようになります。

- DC_cat0 は 10051
- DC_cat1 は 10002

ストア関係キャッシングの例

Business ストア関係を使用するキャッシング・ページがどのように動作するかを理解するため、以下の例を考慮してください。

サイトでサンプルの複合ストア・アーカイブ Business DemandChain.sar をパブリッシュしてから、ホストされるストア (たとえば ResellerOne) を作成すると、以下のストアが作成されます。

表 4.

ストア ID	ディレクトリー	ストア・タイプ
10001	CommercePlaza	チャネル・ハブ
10002	CommercePlazaCatalog	カタログ資産ストア
10003	CommercePlaza	ディストリビューター・プロキシー
10004	ConsumerDirectResellerProfile	ホストされるストアフロント資産ストア
10051	ResellerOne	販売店のホストされるストア

ResellerOne (10051)、つまり販売店のホストされるストアは、ホストされるストアフロント資産ストア (10004) およびカタログ資産ストア (10002) で定義される資産を使用します。

キャッシング関係をセットアップするため、キャッシュ・フィルターは次の情報を入手します。

表 5.

ストア ID	関係タイプ	getStorePath()	getStoresFor RelatedStore()
10001	-1 (ビジネス・ポリシー) -4 (カタログ) -7 (価格) -17 (通貨形式) -19 (サポートされる通貨)	10002	該当しない
10001	-6 (ホストされるストア)	10051	該当しない
10051	-1 (ビジネス・ポリシー) -14 (基本アイテム)	10051, 10002, 10004	10051
10051	-3 (キャンペーン) -5 (コマンド) -10 (URL) -11 (ビュー)	10051, 10004	10051
10051	-4 (カタログ) -7 (価格) -17 (通貨変換) -18 (通貨形式) -19 (サポートされる通貨) -20 (カウンター値通貨) -21 (計算形式)	10051, 10002	10051

その後、キャッシュ・フィルターは次の依頼の属性をセットアップします。

表 6.

ストア関係	ストア ID 10051	ストア ID 10051	ストア ID 10001
-1 (ビジネス・ポリシー)	DC_bus0=10051 DC_bus1=10002 DC_bus2=10004	DC_bus_RS_0=10051	DC_bus0=10002
-2 (税)	DC_tax0=10051 DC_tax1=10004	DC_tax_RS_0=10051	
-4 (カタログ)	DC_cat0=10051 DC_cat1=10002	DC_cat_RS_0=10051	DC_cat0=10002
-6 (ホストされるストア)	DC_host0=10051	DC_host_RS_0=10001	DC_host0=10051

カタログ資産ストア (10002) のカタログが変更されるときはいつでも、 ResellerOne ストア (10051) のカタログ・ページも無効化されてからでなければ、それがカタログ資産ストア (10002) からの情報を使用することはできません。 10051 のページを無効化するには、追加の従属 ID をこのストア関係に対して設定することが必要です。 StoreCatalogDisplay に追加の従属 ID を設定する方法は、以下の例に示されています。

```
<!-- Start Store Relationship Dependency Ids -->
<!-- DC_cat1 is the catalog Profile Store ID -->
<dependency-id>storeId
<component id="DC_cat1" type="attribute">
<required>true</required>
</component>
</dependency-id>

<dependency-id>storeId:catalogId
<component id="DC_cat1" type="attribute">
<required>true</required>
</component>
<component id="catalogId" type="attribute">
<required>true</required>
</component>
</dependency-id>

<dependency-id>StoreCatalogDisplay:storeId
<component id="DC_cat1" type="attribute">
<required>true</required>
</component>
</dependency-id>
<!-- Ends Store Relationship Dependency Ids -->
```

作成される追加の従属 ID は以下のとおりです。

- storeId:10002
- storeId:catalogId:10002:10051
- StoreCatalogDisplay:storeId:10002

これらの追加の従属 ID が定義された後は、カタログ資産ストア 10002 に対してそのカタログ資産ストアを無効にする変更が加えられるときはいつでも、ホストされるストア (10051) のページも無効化されます。

キャッシュ・コマンド機能を動的キャッシングに置き換える

WebSphere Commerce の以前のバージョンは CacheCommand (com.ibm.commerce.cache.commands.CacheCommandImpl) を使用して、顧客プロファイルから判別されるユーザーの状態およびタイプによってページをキャッシュするなど、より高度なキャッシング構成をインプリメントしました。

バージョン 5.5 では、キャッシュ・コマンド論理を JSP ファイルに追加することにより、キャッシュ・サーブレットを使用する場合と同様の方法で動的キャッシングを使用してサーブレットまたは JSP ファイル結果をキャッシュできます。

次の例を考えてください。

StoreCatalogDisplay コマンドは、ユーザーの状態およびタイプ属性に基づいて、異なるヘッダーを表示することができます。ヘッダー JSP ファイルをキャッシュするには、ユーザーの状態およびタイプ属性を含む新規の JSP ファイル、CacheParametersSetup.jsp を作成します。たとえば、

```
<%@ page import="com.ibm.commerce.command.CommandContext" %>
<%
    String userState = null;
    String userType = null;

    CommandContext cmdcontext = (CommandContext) request.getAttribute
(ECConstants.EC_COMMANDCONTEXT);
    if (cmdContext != null) {
        userState = cmdcontext.getUser().getState();
        userType = cmdcontext.getUser().getRegisterType();
    }
%>
```

その後、StoreCatalogDisplay.jsp は userState および userType を入力パラメーターとして使用して、 CacheParametersSetup.jsp を静的に組み込み、CachedHeaderDisplay.jsp を動的に組み込みます。

```
<%@ include file="CacheParametersSetup.jsp"%>

<jsp:include page="CachedHeaderDisplay.jsp" flush="true">
  <jsp:param name="storeId" value="<%= storeId %>" />
  <jsp:param name="catalogId" value="<%= catalogId %>" />
  <jsp:param name="langId" value="<%= languageId %>" />
  <jsp:param name="userState" value="<%= userState %>" />
  <jsp:param name="userType" value="<%= userType %>" />
 </jsp:include>
```

CachedHeaderDisplay.jsp ファイルには、入力パラメーターに基づいて異なる情報を表示するためのロジックが含まれています。

```
<%
    if (userType.equals("G")) {
%>
        <table cellpadding="0" cellspacing="0" border="0" width="100%" height="28">
        . . .
        </table>
<%
    }
else {
%>
        <table cellpadding="0" cellspacing="0" border="0" width="100%" height="28">
        . . .
```

```
        </table>
<%
    }
%>
```

キャッシングを完了するためには、入力パラメーターがキャッシュ ID 規則によって識別される必要があります。

```
<cache-entry>
        <class>servlet</class>
        <name>.../CachedHeaderDisplay.jsp</name>
        <property name="save-attributes">false</property>

        <cache-id>
            <component       id="storeId" type="parameter">
                <required>true/required>
            </component>
            <component       id="catalogId" type="parameter">
                <required>true</required>
            </component>
            <component       id="userState" type="parameter">
                <required>true</required>
            </component>
            <component       id="userType" type="parameter">
                <required>true</required>
            </component>
       < /cache-id>
        . . .
    </cache-entry>
```

第 5 部 ストア・データの概要

第 10 章 ストア・データ

この章は、ストアを構成する WebSphere Commerce Server のストア・データ・アーキテクチャーおよびデータ資産の概要を示します。 WebSphere Commerce Server の情報モデルについてもこの章で説明します。

ストア・データとは

ストア・データとは WebSphere Commerce Server のデータベースにロードされる情報で、これによってストアは機能するようになります。ストアが正しく動作するためには、顧客のすべてのアクティビティーをサポートするデータがストアに配置されていなければなりません。たとえば、顧客が買い物できるようにするには、販売商品のカタログ (カタログ・データ)、オーダーの処理に関係するデータ (税および配送データ)、および要求を実行するための在庫 (在庫およびフルフィルメント・データ) をストアに含めなければなりません。

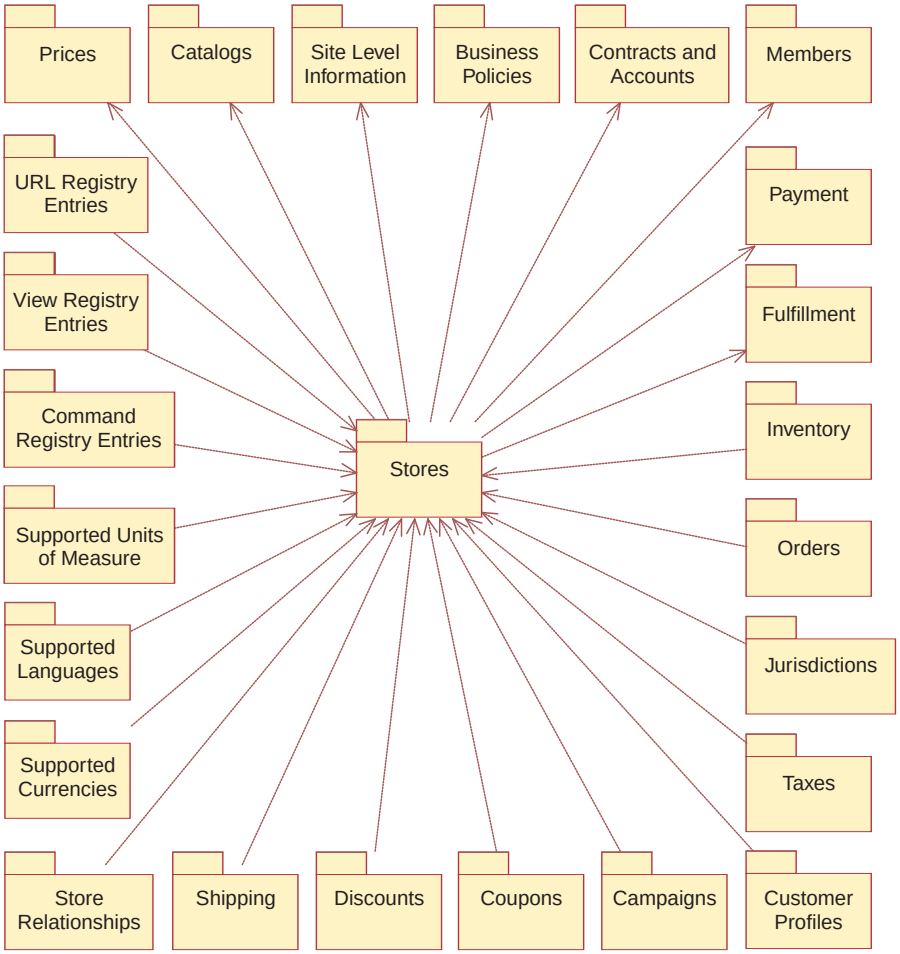
ストア・データの情報モデル

本書では、WebSphere Commerce Server のストア・データの構造を示すために情報モデルが使用されます。 WebSphere Commerce Server の情報モデルは、WebSphere Commerce Server のデータ・モデルに含まれる情報を高水準で抽象化したものです。情報モデルは、データ・モデルの最も重要な機能を強調していますが、スキーマおよびオブジェクトのインプリメンテーションに固有の、より低いレベルの詳細は含まれていません。

たとえば、データ・モデル内のテーブルやオブジェクトにエンティティー・リレーションシップのデータ (外部キーの対など) が含まれている場合、それらのテーブルやオブジェクトがエンティティーとして情報モデルで表現されることはありません。その代わりに、情報モデルの中でそのようなエンティティー・リレーションシップは、エンティティー同士の関連を表す線で表現されます。データ・モデルでは個々のエンティティーが 1 つのテーブルを表しますが、情報モデルでは示されているどのオブジェクトも同じデータベース・テーブルにマップできたり、 1 つのオブジェクトを複数のデータベース・テーブルにマップできたりするという点でも、情報モデルとデータ・モデルは異なります。また、情報モデルでは*詳細拡張* が示されません (これはインプリメンテーションの結果として別個のテーブルに保管される、エンティティーの追加のデータ属性です。たとえば商品の説明は、商品のエンティティーとは別個に保管される拡張です)。最後の点として、データ・モデルとは異なり、情報モデルは継承の概念も示すことができます。エンティティー・リレーションシップ・データおよび詳細拡張の詳細については、 WebSphere Commerce オンライン・ヘルプのデータ・モデルを参照してください。

WebSphere Commerce オブジェクトおよびデータ・モデルの詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

以下の図は、WebSphere Commerce ストアのデータ資産を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce 情報モデルの一部です。情報モデルの詳細については、111 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、487 ページの『付録 A. UML の凡例』を参照してください。

上記の図で示されているそれぞれのデータ資産については、123 ページの『第 6 部 ストア・データの開発』の各章で詳しく説明します。

注:

UML 表記では、あるオブジェクトから伸びて別のオブジェクトを指している矢印付きの点線は、 1 つ目のオブジェクトには 2 つ目のオブジェクトに対する従属関係があることを示します。この図に示されているオブジェクトはパッケージと見なされます。サポートされる通貨のリストなど、一部のパッケージ中のデータは特定のストアに固有なので、このようなパッケージはストア・パッケージに対する従属パッケージとして表示されます。カタログなど、その他のパッケージは特定のストアに固有でなく、逆に個々のストアがカタログを使用できるので、ストア・オブジェクトがカタログ・パッケージに対する従属オブ

ジェクトとして示されます。その結果、サポートされる通貨のリストはストアの*一部* になり、一方ストアはカタログを*使用* します。

Business ストアの*一部* のうち、他のストアとのストア関係に特に注目します。個々のストア関係は、あるストアが別のストアに従属し、特定のサービスか情報を備えることを示します。関係を定義すると、あるストアのデータ (サポートされる通貨のリストなど) を別のストアで使いやすくすることができます。このシナリオでは、1 つ目のストアが、 2 つ目の (クライアント) ストアで使用されるデータのプロバイダーつまりコンテナーの働きをします。追加のクライアント・ストアを作成するたびに、それらのストアでも他の特定のストアから特定のデータを入手することを示す関係を定義できます。こうすると、関係によってデータを共用しやすくなります。つまり、プロバイダー・ストアでデータを一度作成して保守すると、複数のクライアント・ストアでそのデータを使用できます。ストア関係の詳細については、 151 ページの『第 14 章 ストア間の関係』を参照してください。

情報モデル中のデータは、以下の方法で分類できます。

- サブシステム別
- データ・タイプ別

サブシステム別に表示されるストア・データ情報モデル

ストア・データ情報モデル中のそれぞれのデータ資産は、以下の機能領域にグループ化できます。

表 7.

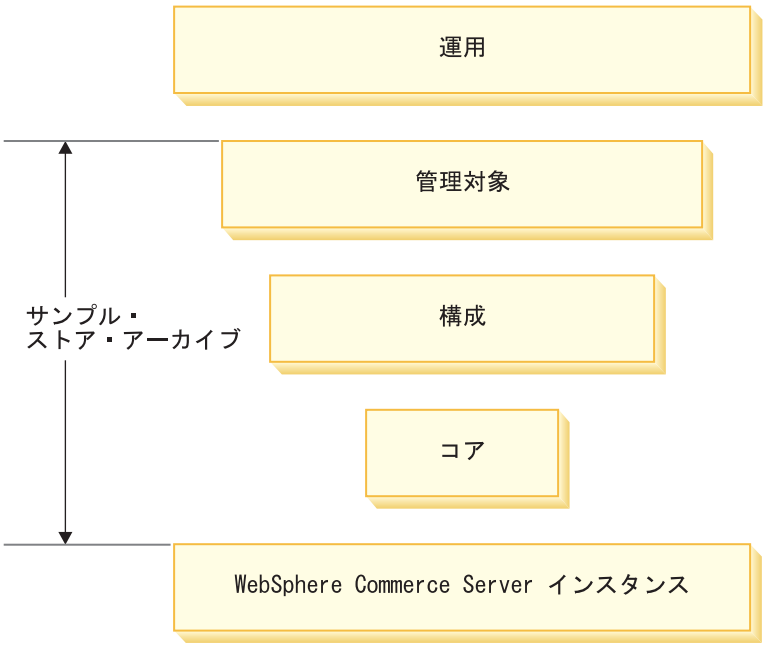
取引管理	市場	取引	オーダー管理	カタログ	メンバー	ランタイム
割引	キャンペーン	契約	配送	カタログ	組織	組織
取引先	顧客プロファイル	アカウント	税	価格	グループ	URL、コマンド、およびビュー・レジストリー
オークション	E メール・アクティビティー	Business RFQ	管轄区域		ユーザー	サポートされている言語
	クーポン		オーダー			サポートされている計測単位
			在庫			サポートされている通貨
			フルフィルメント			サイト
			支払			ストア
						ストア関係

表 7. (続き)

						ビジネス・ポリシー

データ・タイプ別に表示されるストア・データ情報モデル

WebSphere Commerce ストア内のデータは、以下の図に示されているタイプに従います。 113 ページの『サブシステム別に表示されるストア・データ情報モデル』にある図で示されているストア・データ資産はそれぞれ、以下に示すストア・データのタイプの 1 つ以上に属するものとして分類することができます。



WebSphere Commerce Server インスタンス

基本レベルのデータは、WebSphere Commerce Server インスタンスに含まれています。インスタンスが作成されると、 XML 形式でロードされるブートストラップ・ファイルによりデータベースに情報が取り込まれます。ブートストラップ・ファイルは、以下のデータ・タイプを作成します。

- 計算方法タイプ、デバイス・タイプ (ブラウザー、E メール、I モードなど)、メッセージ・タイプ、役割、およびアドレス
- デフォルト管理 ID (WCSADMIN)
- デフォルトのコマンド、ビュー、および URL
- デフォルトのビジネス・ポリシー
- デフォルトのアクセス・グループおよびアクセス制御ポリシー
- インスタンスによってサポートされる言語および通貨
- デフォルトの数量単位および数量単位変換
- デフォルトのスケジュール・ジョブおよび都道府県コード
- デフォルトの条件
- デフォルトの組織 (ストアの所有者として使用可能)

- デフォルトのサイト組織
- デフォルトのストア・グループ
- デフォルトのステージング用情報

この情報は、そのインスタンス内に存在するすべてのストアで適用可能です。また、この情報は 113 ページの『サブシステム別に表示されるストア・データ情報モデル』の図で、サイト・レベル情報として示されています。

ブートストラップ・ファイルおよびこのファイルがデータを読み込むデータベース・テーブルの詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

コア・データ

ストア・データの次のレベルは、コア・データです。コア・データは、2 つのレベルに分割されています。

- 組織
- ストア

組織コア・データは、ビジネス・モデルの特定の環境における最小データを作成します。これには次のものが含まれます。

- 組織構造。
- 事前定義されたユーザー役割。
- 必要なアクセス制御ポリシー。

組織コア・データは、サンプル複合ストア・アーカイブ、およびサンプル組織構造コンポーネント・ストア・アーカイブの両方から入手できます。

コア・データは、その環境内のストアの最小データを構成します。これには次のものが含まれます。

- STOREENT テーブル内のストア ID。これはデータベースにストアを作成します。
- デフォルトの契約
- 契約データベース・テーブル内のストア ID
- 契約データベース・テーブル内のストアを所有する組織のメンバー ID
- STORE テーブル内のストア・ディレクトリー。ストア・ディレクトリーは、ストアの Web 資産があるディレクトリーです。
- STADDRESS テーブル中のストアのアドレスのためのニックネームまたは識別名。ニックネームはストアごとに固有です。

ストア・コア・データは、サンプル複合ストア・アーカイブ、およびサンプル・コンポーネント・ストア・アーカイブの両方から入手できます。

管理コンソールの発行ユーティリティーを使用して、上に示されているサンプル・ストア・アーカイブのいずれかを発行した場合、この情報は自動的に作成されました。発行ユーティリティーを使用すると、ストア所有者の役割を果たすデフォルトの組織を選択することができます。あるいは、組織管理コンソールを使用して、所有者の役割を果たす別の組織を作成することもできます。サンプル複合ストア・アーカイブを発行してストアの基盤として使用しなかった場合、ローダー・パッケー

ジを使用してその情報をデータベースにロードするか、またはデータベースを直接編集する必要があります。ローダー・パッケージの使用法について詳しくは、 379 ページの『第 37 章 ストア・データのロードの概要』を参照してください。

113 ページの『サブシステム別に表示されるストア・データ情報モデル』の図にあるストア・データは、コア・データです。

構成データ

構成データは、コマース・サーバーのランタイムを制御します。コマース・サーバー・ランタイムは、コマース・アプリケーションがデプロイ、実行されるフレームワークを提供します。このフレームワークは、コマンドの実行、例外処理、トランザクション制御、データ・アクセス、およびパーシスタンスで構成されます。コマース・サーバー・ランタイムは WebSphere Application Server によって提供されるランタイム・サービスを使用して、 WebSphere Commerce Server アプリケーションをサポートします。構成データは、ストア・ページを表示するためにストアが使用するコマンド、ビュー、および JSP ファイルを決定します。

113 ページの『サブシステム別に表示されるストア・データ情報モデル』の図に示されている以下のデータ資産は、構成データとして分類されます。

- コマンド・レジストリー項目
- ビュー・レジストリー項目
- URL レジストリー項目

管理対象データ

管理対象データは、セラーが作成するデータです。セラーのサイトの顧客にとって、これは読み取り専用データです。セラーはこのデータの状態を完全に制御できるため、管理対象データはコンテンツ・マネージメント・システムを介して管理することもできます。

113 ページの『サブシステム別に表示されるストア・データ情報モデル』の図に示されている以下のデータ資産は、管理対象データとして分類されます。

- ビジネス・ポリシー
- キャンペーン
- カタログ
- 契約
- クーポン
- 通貨
- 顧客プロファイル
- 割引
- E メール・アクティビティー
- 配送センター
- 在庫 (カタログ・アイテムの構成情報)
- 管轄区域
- 言語
- メンバー

- 支払
- 価格
- セラー
- 配送
- 税
- 計測単位
- 取引先

運用データ

運用データとは、サイトの顧客がサイトとの対話の結果として (直接的または間接的に) 作成または変更するデータのことです。たとえば、顧客のオーダーは運用データと見なされます。また、在庫レベルも運用データと見なされます。ストアの運用に応じて上下するからです。顧客も運用データと見なされます。また、セラーにより作成されたデータも運用データと見なすことができます。

セラーは運用データに対して加える変更を完全には制御できないため、このデータはコンテンツ・マネージメント・システムを使用しても管理されません。

113 ページの『サブシステム別に表示されるストア・データ情報モデル』の図に示されている以下のデータ資産は、運用データとして分類されます。

- オークション
- 契約
- 顧客
- E メール・アクティビティー
- フルフィルメント
- 在庫 (受け取り、予測受け取り、在庫の割り振り)
- オーダー
- Business 見積依頼 (RFQ)

注: 中には、運用データと管理対象データのどちらに属するかを区別しにくいインスタンスもあります。たとえば、あるストアでは顧客および契約データが管理対象データと見なされ、別のストアでは同じタイプのデータが運用データと見なされる場合があります。最初のストアでは、顧客の集団が限定されているため、顧客データとそれに関連する契約を管理しているのかもしれません (つまり、顧客はオンラインで登録できません)。一方、2 番目のストアでは、顧客はオンラインで登録して、オンラインで契約情報を作成できます。

2 番目の例は、カタログ・データです。単一セラー・サイトでは、カタログは管理対象データと見なされます。バリュー・チェーン・サイトでは、カタログ・データは運用データと見なされます。

サイトによっては、同じタイプのレコードでも、管理対象データと見なされる場合と、運用データと見なされる場合とがあります。たとえば、デフォルト契約は管理対象データと見なされる一方、オンラインで結ばれる特定の契約は運用データと見なされることがあります。別の例は E メール・アクティビティーです。 E メール・アクティビティー情報およびテンプレートは管理対象データ

と見なされますが、テンプレートで生成されて顧客に送信される実際の E メール・アクティビティーは、運用データと見なされます。また、たとえば顧客が E メールを開封したり、E メールのクリック可能なコンテンツをクリックしたりといった、メール送信の結果として発生するあらゆるイベントも同様に運用データと見なされます。

ストア・データ・タイプおよびサンプル・ビジネス

WebSphere Commerce に付属のサンプル・ビジネスには、ストア・データ・アーキテクチャーのストア・データ・タイプがほとんど含まれています。たとえば、WebSphere Commerce Server インスタンスがあらかじめ存在していなければ、サンプル・ストアを使用してストアを作成したり、あるいはサンプル・ストアを発行したりすることはできません。その後、管理コンソールにある発行ユーティリティーのツールを使用して、サンプル・ストアに基づくストアを作成するときに、コア・データが作成されます。サンプル・ストアには、必要となるすべての構成データすべてや、機能するストアに必要な管理対象データのほとんどが含まれています。特定のサンプル・ストアに基づいてストアを作成する場合、 WebSphere Commerce アクセラレーターのツールを使用してデータのセットアップのいくつかを完了するように指示される場合があります。

データ作成用のツール

WebSphere Commerce には、ストア・データを作成したり操作したりするためのツールがいくつか用意されています。それらのツールは、以下のとおりです。

WebSphere Commerce ローダー・パッケージ

ローダー・パッケージは、基本的に言って、データを準備して WebSphere Commerce データベースにロードするユーティリティーで構成されます。詳細については、 363 ページの『第 10 部 ストアの発行』を参照してください。

管理コンソール

管理コンソールは、管理操作および構成タスクを実行して、サイトまたはストアを制御する手段になります。また、管理コンソールを使用して、新しい組織や新しいユーザーを作成したり、ユーザーに役割を割り当てたりすることもできます。また、管理コンソールを使用して、ストアで使用可能な通知タイプおよびメッセージ・タイプを識別することもできます。管理コンソールには、サンプル・ビジネスおよびサンプル・ストアを発行できるようにする発行ユーティリティーが含まれています。

WebSphere Commerce アクセラレーター

WebSphere Commerce アクセラレーターは、オンライン・ツールのワークベンチで、これを使用してさまざまなストア資産を作成および保守することができます。大部分のストア・データは、 WebSphere Commerce アクセラレーターにあるツールを使用して作成および管理できます。詳細については、 119 ページの『ツールとストア・データの要約表』を参照してください。

組織管理コンソール

組織管理コンソールを使用すると、サイトまたはストアにアクセスする組織を作成および管理することができます。組織管理コンソールを使用すると、バイヤーの管理者が組織内のバイヤーを管理することも可能です。

ツールとストア・データの要約表

以下の図は、各データ・タイプを作成するために使用できるツールを示しています。

データ作成用のツール	コア・データ	構成データ	管理対象データ	運用データ
WebSphere Commerce ローダー・パッケージ	コア・データを XML ファイルの形式でロードするには、ローダー・パッケージを使用します。詳細については、145 ページの『XML ファイルによるストア・データ資産の作成』を参照してください。	構成データを XML ファイルの形式でロードするには、ローダー・パッケージを使用します。詳細については、160 ページの『コマンド、ビュー、および URL の登録用 XML ファイルの作成』を参照してください。	管理対象データを XML ファイルの形式でロードするには、ローダー・パッケージを使用します。詳しくは、管理対象データ資産に関する対応する章を参照してください。	一般に、ローダー・パッケージで運用データをロードすることはできません。しかし、ローダー・パッケージを使用して、選択した顧客データをローダーことはできます。
管理コンソール	管理コンソールを使用してストア・アーカイブを発行すると、コア・データが自動的に作成されます。発行の使い方の詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。	該当しない。	該当しない。	該当しない。

データ作成用のツール	コア・データ	構成データ	管理対象データ	運用データ
WebSphere Commerce アクセラレーター	該当しない。	該当しない。	以下のデータを作成または編集するには、WebSphere Commerce アクセラレーターを使用します。 • キャンペーン • 契約 (データベースにあらかじめデフォルト契約が存在していないと、WebSphere Commerce アクセラレーターのビジネス関係管理ツールを使用して追加の契約を作成したり、既存の契約を変更したりすることができません。データベースにデフォルト契約を作成するには、ローダー・パッケージを使用するか、ストア作成ウィザードを使用するか、ストア・アーカイブを発行してください。) • 管轄区域 • 税 • 配送 • 通貨 • 言語	顧客はストアに登録するとき、またはストアで買い物をするときに、運用データを作成します。ただし、場合によっては、WebSphere Commerce アクセラレーターを使用して顧客のオーダーを発行したり、返品を作成したりすることもできます。 また、WebSphere Commerce アクセラレーターを使用して在庫 (受け取りおよび予測受け取り) を管理することもできます。

データ作成用のツール	コア・データ	構成データ	管理対象データ	運用データ
WebSphere Commerce アクセラレーター (続き)	該当しない。	該当しない。	• フルフィルメント • 割引 • カタログ (データベースにあらかじめマスター・カタログが存在していないと、WebSphere Commerce アクセラレーターの商品管理ツールを使用して、商品情報を追加したり変更したりすることができません。商品取引とサービスを別個に表示するために、 XML ソースに変更を加えてセールス・カタログを作成します。データベースにマスター・カタログを作成するには、ローダー・パッケージを使用するか、ストア・アーカイブを発行してください。) • 価格	該当しない。

データ作成用のツール	コア・データ	構成データ	管理対象データ	運用データ
組織管理コンソール	組織を作成して管理するには、組織管理コンソールを使用します。	該当しない。	該当しない。	顧客およびバイヤーは、ストアに入るときに作成されます。ただし、組織管理コンソールを使用して、ユーザーの管理とバイヤーの承認を行ったり、あるいは新規バイヤーを作成したりすることもできます。

第 6 部 ストア・データの開発

ここに含まれる章では、各ストア・データ資産についてさらに詳しく説明します。ここで説明されているストア・データ資産は、 WebSphere Commerce ストア・データ・アーキテクチャー構造に従って編成されています。

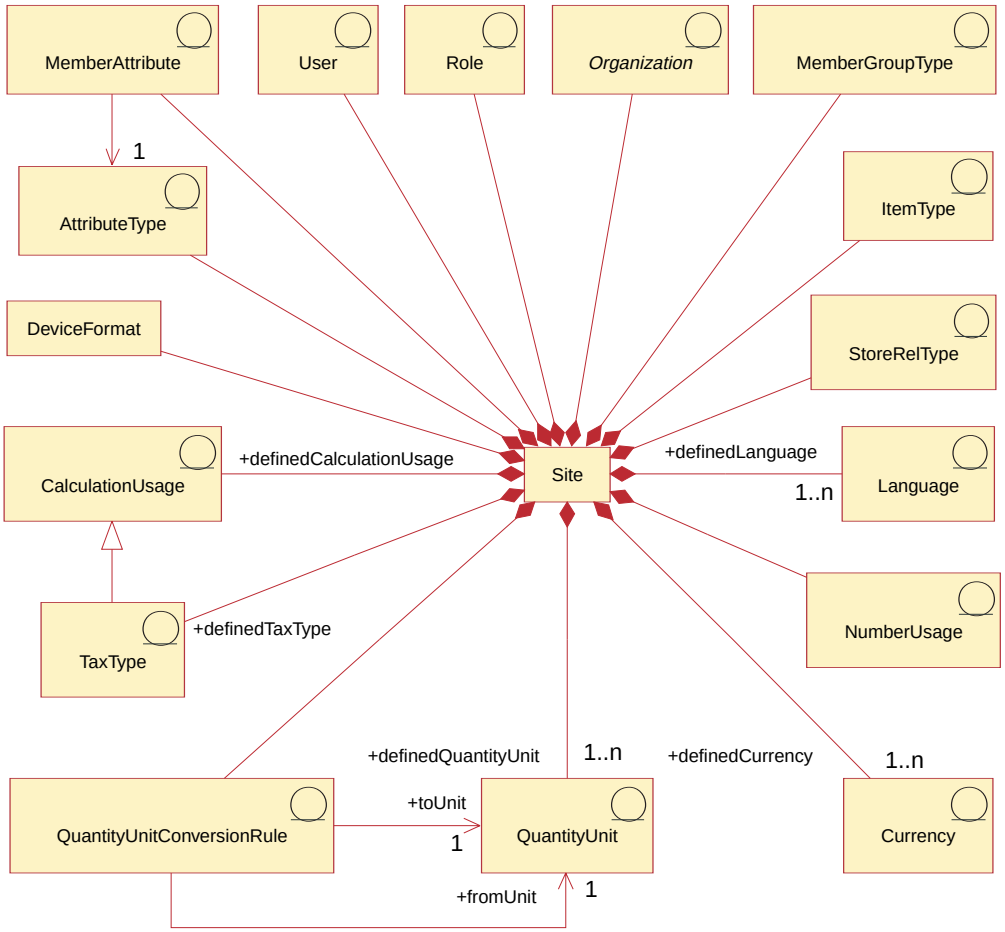
- WebSphere Commerce Server インスタンス
  - サイト
- コア・データ
  - 組織
  - ストア
  - ストア間の関係
- 構成データ
  - コマンド・レジストリー
  - ビュー・レジストリー
  - URL レジストリー
- 管理対象データ
  - カタログ
  - 価格
  - 契約 (ビジネス・ポリシーを含む)
  - フルフィルメント
  - キャンペーン
  - 支払
  - サポートされている言語
  - サポートされている通貨
  - サポートされている計測単位
  - 管轄区域
  - 配送
  - 課税
  - 割引
- 運用データ
  - 在庫
  - オーダー
  - 顧客
  - オークション
  - Business RFQ

第 11 章 サイト資産

WebSphere Commerce Server インスタンスにはそれぞれ、関連情報の独自のデータベースがあります。インスタンスはブートストラップ・ファイルによって作成されます。ブートストラップ・ファイルは、スキーマが作成された後、データベース・テーブルに情報を取り込みます。データがロードされたら、該当するデータベース・テーブルで事前にロードされた情報を見ることができます。多くのデータベース・テーブルには、ストア・レベルの情報またはストア・グループ・レベルの情報が入っています。これは、ストアまたはストアのグループに固有の情報です。いくつかのテーブルには、インスタンス内のすべてのストアで使用可能な WebSphere Commerce サイト・レベルの機能を表す情報が入っています。このすべての情報は WebSphere Commerce サイト管理者により管理されます。この章では、それらの機能について説明します。ブートストラップ・ファイルの詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。ストア固有の資産情報に関する詳細については、143 ページの『第 13 章 ストア資産』を参照してください。

WebSphere Commerce のサイト資産について

以下の図は、サイトに含まれるデータのタイプ、およびそれらのデータのタイプとサイトとの関係を示しています。



この図で使用されている規則の詳細については、 487 ページの『付録 A. UML の凡例』を参照してください。この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce 情報モデルの一部です。情報モデルの詳細については、 111 ページの『ストア・データの情報モデル』を参照してください。

言語

サイトでは、 LANGUAGE テーブルの中に多くの*言語* を定義し、 LANGUAGEDS テーブルの中でそれらを記述することができます。各ストアは一般に、 STORELANG テーブルに行を追加することによって、これらの言語のサブセットをサポートします。事前に定義されている言語は、ドイツ語、中国語 (繁体字)、中国語 (簡体字)、日本語、韓国語、イタリア語、フランス語、スペイン語、ブラジル・ポルトガル語、および英語です。

メンバー属性

メンバー属性 は MBRATTR テーブルに保管されます。メンバー属性は、組織またはユーザーに応じた値を保管できる、定義済みの属性のセットを表します。その種の属性名の例として、JobFunction、ProcurementCard、SpendingLimit、ReferredBy、お

よび CountryOfOperation などがあります。特定の組織またはユーザーの属性値は MBRATTRVAL テーブルに保管されます。これらの値は各ストアまたはストア・グループごとに異なる場合があります。

属性タイプ

属性タイプ は ATTRTYPE テーブルに保管され、属性値を表すために使用できる定義済みデータ・タイプを表します。データ・タイプの例として、INTEGER、STRING、および FLOAT があります。

メンバー・グループ・タイプ

メンバー・グループ・タイプ は MBRGRPTYPE テーブルに保管され、定義済みのメンバー・グループの使用法を表します。 MBRGRPUSG テーブルに行を追加することによって、メンバー・グループに使用法が割り当てられます。メンバー・グループの使用法の例として、AccessGroup (アクセス制御ポリシーに関して使用)、および UserGroup (顧客グループなど一般用) などがあります。

ユーザー

ユーザー は、認証されたユーザー ID を表します。ユーザーは一般に、購買組織に代わってオーダーを発行または承認する顧客、販売組織のオーダーを処理したり、あるいはストア・レベルの資産を保守したりする販売エージェント、または WebSphere Commerce Server インスタンスを保守するサイト管理者を表します。各ユーザーは、1 つのサイトと関連付けられ、USERS テーブルの中で定義されます。

組織

組織 という資産は、組織、および組織内にある組織単位を表します。組織は一般に、購買または販売に責任を持つビジネス・エンティティーを表します。 B2B 向け購買組織内の顧客が発行するオーダーは、その購買組織の代わりに発行されたものとして記録されます。ストア、カタログ、および配送センターは、販売の特定の局面に責任を持つ組織によって所有されます。組織は ORGENTITY テーブルに定義されています。

役割

役割 は、組織内でユーザーに割り当てることができる、定義済みの役割のセットを表します。たとえば、ユーザーに販売組織内の顧客サービス担当者の役割を割り当てたり、あるいは購買組織内のバイヤー承認者の役割を割り当てたりすることもできます。デフォルトの役割の名前と説明は、ROLE テーブルに読み込まれます。特定の役割の詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

数量単位変換

各サイトには、*数量の変換規則* があります。それらの変換規則は、異なる計測単位を変換するために使用される乗算または除算の演算を表します。これは、QTYCONVERT テーブルに読み込まれます。

数量単位

数量単位 は、サイトの計測単位のセットを表します。それらは、QTYUNIT テーブルの中で定義され、QTYUNITDSC テーブルの中に記述します。各ストアは、計測単位の値を丸めて表示用に形式設定する方法を、使用目的に応じて指定できます。これは、QTYFORMAT テーブルに行を追加することによって行えます。

税タイプ

税タイプ は、税を計算する計算方法を表します。消費税と配送税は、税を計算する 2 つの異なる計算方法です。税タイプは TAXTYPE テーブルに定義されています。

計算の使用法

計算の使用法 は、 OrderPrepare コマンドで実行できる、さまざまな種類の計算を表します。計算の使用法は、割引、配送、消費税、配送税、および e クーポンに関して定義されます。計算の使用法は、CALUSAGE テーブルの中で定義されます。

通貨

各サイトでは、 SETCURR テーブルの中でいくつかの*通貨* を定義し、SETCURRDSC テーブルの中でそれらを記述します。各ストアは、CURLIST テーブルに行を追加することによって (サポートされる通貨に対してそれぞれ 1 行)、それらの通貨のサブセットをサポートします。

注: 言語、通貨、数量単位、および数量単位変換規則など、サイト資産の一部については、サイト管理者は該当するテーブルに行を追加することによって、サイト・レベルの機能を拡張することができます。 その他のサイト資産については、それらが表すサイト・レベルの機能を拡張するために、関係するカスタマイズも必要になる可能性があります。たとえば、サイト管理者が、カスタマイズされた通貨記号と一緒に小計を表示するよう、数値の新しい使用法を追加するとします。その場合、小計を表示するプログラムにカスタマイズを加えて、小計の金額を表示用に形式設定する際、新しい小計の数値使用法を指定する必要があるかもしれません。

数値の使用法

数値の使用法 は、数値を使用する方法を表します。ストアは、表示する数値の使用方法に応じて、それぞれの数値に異なる丸めと形式設定のルールを指定することができます。たとえば、あるストアは単価については単価の使用法を指定して小数第 4 位で丸め、他の通貨の額についてはデフォルトの使用法を指定して小数第 2 位で丸めるかもしれません。数値の使用法は、NUMBRUSG テーブルの中で定義され、NUMBRUSGDS テーブルの中で記述されます。

アイテム・タイプ

アイテム・タイプ は、さまざまな基本アイテムを表します。 WebSphere Commerce には、ダイナミック・パッケージと通常アイテムの 2 種類の基本アイテムがあります。アイテム・タイプは、ITEMTYPE テーブルで事前定義されています。基本アイテムの詳細については、 305 ページの『第 29 章 在庫資産』を参照してください。

デバイス形式

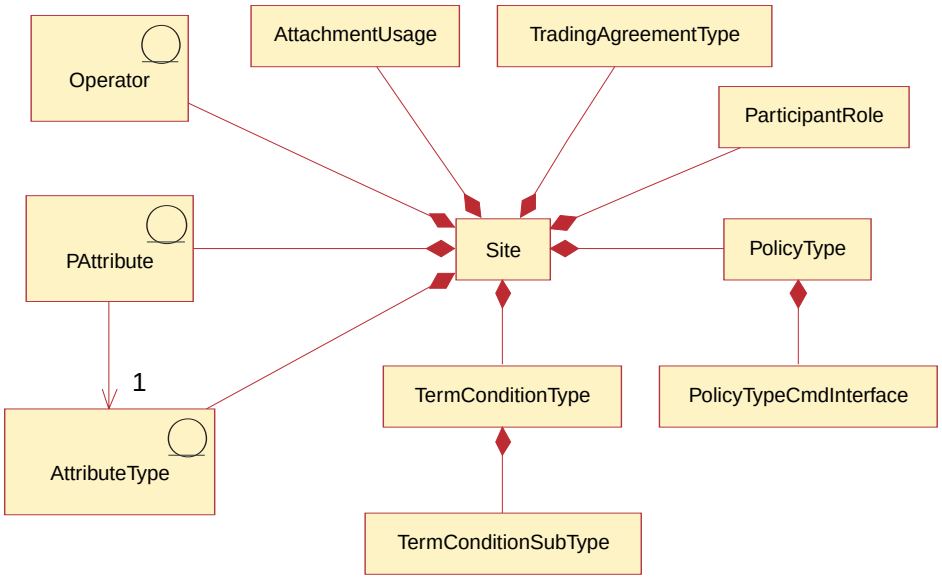
デバイス形式 は DEVICEFMT テーブルに保管され、サイトが使用する多くのデバイス形式 (ブラウザー、I モード、E メール、XMLMQ、および XMLHTTP など) を表します。これらすべてのデバイス・タイプによって、ユーザーはさまざまなメディアを介してサイトと対話することができます。

ストア関係タイプ

Business *ストア関係タイプ* (StoreRelType) は、2 つのストア間の関係のタイプを定義します。ストア関係の各タイプは、それ自体の関係を定義しています。すなわち、関係がある各パートナーの果たす役割、および 2 つパートナー間にどのような関係があるかを定義しています。ストア関係は、STRELTYP テーブルの中で定義され、STRELTYPDS テーブルの中で記述されます。

サイト・レベルの取引条件データ

以下の図は、サイトに含まれる取引条件データのタイプ、およびそれらのタイプとサイトとの関係を示しています。



この図で使用されている規則の詳細については、 487 ページの『付録 A. UML の凡例』を参照してください。この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce 情報モデルの一部です。情報モデルの詳細については、 111 ページの『ストア・データの情報モデル』を参照してください。

取引条件タイプ

WebSphere Commerce には、バイヤーとセラーの間の相互作用を制御する取引メカニズムが多数用意されています。取引条件は、取引メカニズムのインスタンスを表し、取引メカニズムのそのインスタンスのプロパティーを記録します。 WebSphere Commerce の契約、ビジネス・アカウント、および RFQ は、それぞれ取引条件に示されます。 WebSphere Commerce では、すべてのオークションを制御する単一の

取引条件があります。 WebSphere Commerce は、アカウント、契約、RFQ、取引、およびオークションを含む、いくつかの*取引条件タイプ* をサポートしています。取引条件タイプは TRDTYPE テーブルに定義されています。取引条件タイプの詳細については、 205 ページの『第 18 章 契約資産』を参照してください。

参加者の役割

取引条件の参加者は、各取引条件内で特定の役割を持ちます。 WebSphere Commerce は、作成者、セラー、バイヤー、サプライヤー、承認者、管理者、ディストリビューター、サービス・プロバイダー、販売店、ホスト、および宛先を含む、いくつかの*参加者の役割* をサポートします。参加者の役割は PARTROLE テーブルに定義されています。

ポリシー・タイプ

WebSphere Commerce は、価格、商品セット、配送モード、配送料、支払、その他を含む、ビジネス・ポリシーのいくつかのタイプをサポートします。ポリシー・タイプは POLICYTYPE テーブルに定義されています。ビジネス・ポリシーの詳細については、 205 ページの『第 18 章 契約資産』を参照してください。

ポリシー・タイプ・コマンド・インターフェース

ポリシー・タイプ・コマンド・インターフェース は、ビジネス・ポリシー・オブジェクト用の Java コマンド・インターフェースです。各ポリシー・インスタンス用のコマンドが、このインターフェースをインプリメントしている必要があります。ビジネス・ポリシー・オブジェクトごとに、ゼロあるいはそれ以上のコマンドがあります。

契約条件タイプ

契約条件は、取引条件の振る舞いとプロパティーを定義します。 WebSphere Commerce は、価格設定、支払、および配送を含む、いくつかの*契約条件タイプ* をサポートします。契約条件タイプは TCTYPE テーブルに定義されています。契約条件の詳細については、 205 ページの『第 18 章 契約資産』を参照してください。

契約条件のサブタイプ

各契約条件タイプには、いくつかの*契約条件のサブタイプ* を含めることができます。契約条件のサブタイプは TCSUBTYPE テーブルに定義されています。

個人情報設定属性

個人情報設定属性 は、商品の属性を作成できるようにします。個人情報設定属性は PATTRIBUTE テーブルに定義されています。各個人情報設定属性は、1 つだけ属性タイプを持ちます。

属性タイプ

属性タイプ は、属性のタイプを定義します。属性タイプは ATTRTYPE テーブルに定義されています。

演算子

サイトで使用される*演算子* には、単項演算子 (単一値を指定できる)、複合演算子 (範囲 - 連続)、および複合演算子 (設定)が含まれます。演算子は OPERATOR テーブルに定義されています。

添付ファイルの使用法

添付ファイルは、取引文書のサポート文書です。たとえば、商品の仕様、または価格表スプレッドシートなどです。*添付ファイル使用法* では、添付ファイルが使用される方法と場所を説明しています。添付ファイル使用法は ATTACHUSG テーブルに定義されています。

WebSphere Commerce のサイト資産の作成

サイト資産は、WebSphere Commerce Server のインスタンスを作成するときに作成されます。 WebSphere Commerce Server のインスタンスの作成の詳細については、「*WebSphere Commerce インストール・ガイド*」の『WebSphere Commerce インスタンスの作成』を参照してください。

第 12 章 メンバー資産

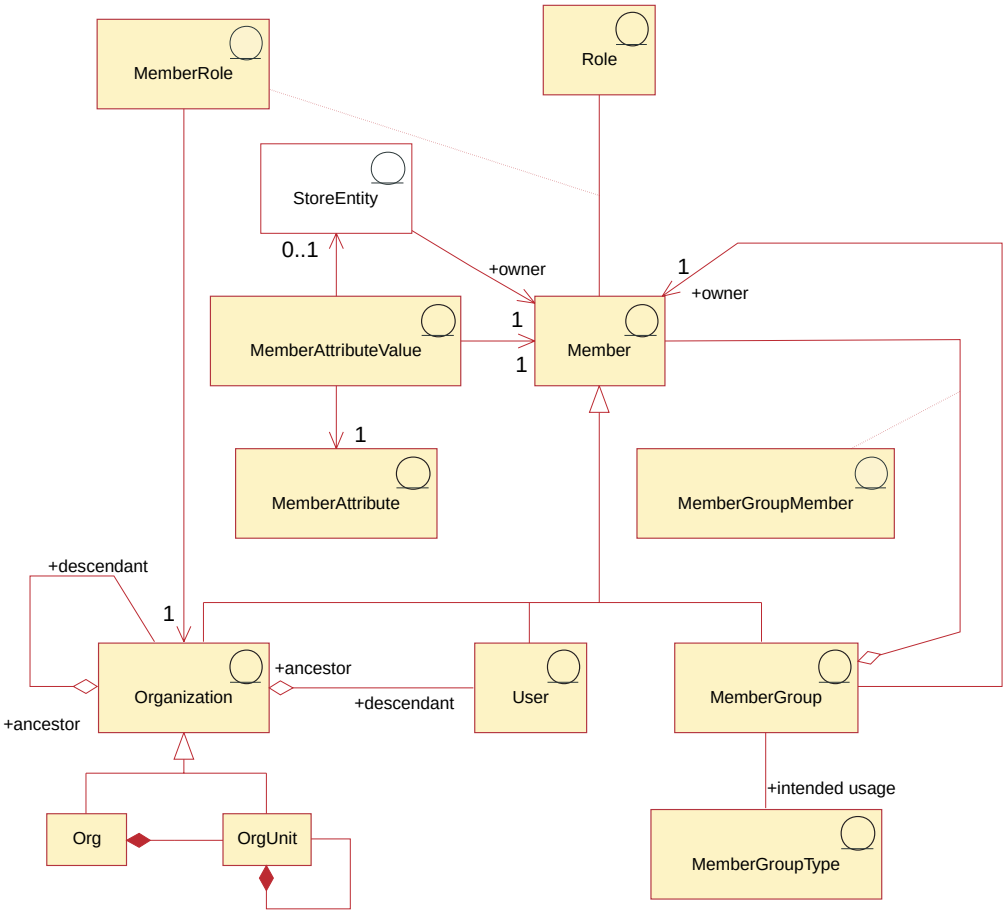
この章では、WebSphere Commerce メンバー・サブシステムについて説明してから、ストア・デベロッパーに関連した 3 つのタイプのメンバーである顧客、セラー、および管理者について説明します。 WebSphere Commerce には、メンバーまたはユーザー、および組織を含むメンバー・サブシステムが備わっていることに注目してください。

WebSphere Commerce のメンバー資産について

WebSphere Commerce メンバー資産には、WebSphere Commerce システムの参加者に関するデータが組み込まれます。メンバーになれるのは、ユーザー、ユーザーのグループ、または組織エンティティーです。管理者 (サイト管理者など) は、ユーザーや組織エンティティーのメンバーに役割を割り当てます。メンバーに役割が割り当てられると、アクセス制御コンポーネントは、メンバーがアクティビティーに参加するのを許可します。たとえば、組織にはバイヤーまたはセラーのいずれか、あるいは両方の役割を割り当てることが可能です。ユーザーへは複数の役割を割り当てることができます。管理者は、さまざまなビジネス目的に応じて分類されるユーザーのグループである、メンバー・グループを作成することができます。
WebSphere Commerce の管理コンソールを使用して、組織、ユーザー、役割、およびメンバーのグループを作成して処理してください。

メンバー資産に関するビジネス・ロジックは、メンバー登録およびプロファイル管理サービスを提供します。メンバー資産と密接に関係する別のサービスには、アクセス制御、認証、およびセッション管理が含まれます。これらのトピックに関する詳細情報は、 WebSphere Commerce オンライン・ヘルプを参照してください。

以下の図は、WebSphere Commerce のメンバー資産を示しています。図の後に、個々の資産の説明があります。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、111 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、487 ページの『付録 A. UML の凡例』を参照してください。

メンバー

WebSphere Commerce のメンバー は、以下のいずれかになります。

- *組織エンティティー*。組織エンティティーとは、「IBM」などの組織や「電子商取引部門」などの組織内の組織単位のことです。
- ユーザー (登録済みであることも、登録されていないこともあります)。登録済みユーザーには、固有 ID とパスワードがあり、登録上の目的により、プロファイル・データの入力が必要です。登録済みユーザーは、そのプロファイル・タイプに従って分類されます。タイプ「B」はビジネス・ユーザー (または Business B2B 向け顧客) を、 タイプ「C」は小売ユーザー (または消費者向け顧客) を指します。登録済みユーザーと登録されていないユーザーの詳細については、WebSphere Commerce オンライン・ヘルプの『メンバー』を参照してください。

- メンバー・グループ。これは、さまざまなビジネス上の理由でカテゴリー化されたユーザーのグループです。グループ分けはアクセス制御の目的、承認の目的、およびマーケティングの目的 (割引や価格の計算や商品の表示など) で使用することができます。

メンバーは、個々のストア・エンティティー (つまり、ストアまたはストア・グループ) を所有します。

メンバー属性

WebSphere Commerce のメンバーには*属性* の集合があり、個々の属性には関連した*値* があります。メンバーの基本ユーザー・プロファイルは、登録情報、個人情報、住所情報、購入ヒストリー、およびその他の各種属性をまとめたものです。

ビジネス・ユーザー・プロファイルには、基本ユーザー・プロファイルに加え、従業員番号、仕事の肩書き、仕事の内容などのような雇用情報が含まれます。登録の際に、ビジネス・ユーザーは、所属する企業組織を識別する必要があります。組織エンティティーのプロファイルは、組織名や業種などのような追加情報を含みます。

アクセス制御ルールにより、プロファイル管理を実行するためのユーザー権限が決まります。メンバー・プロファイルには、さまざまな個別属性およびビジネス関連属性 (例えば、役割、支払い情報、住所、優先言語および優先通貨、パーベイシブ・コンピューティング・デバイス) を入れることができます。属性をストアに依存させることができます。これらの属性はユーザーおよび組織エンティティーに関してサポートされていますが、メンバー・グループに関してはサポートされていません。

役割

各ユーザーは、組織で 1 つ以上の役割を果たすことができます。サイト管理者は、各メンバーに役割を割り当てます。たとえば、IBM 社のメンバーであるジョン・スミス氏の役割は顧客サービス担当者で、ジョン氏は IBM 社の顧客にかかわる仕事をし、顧客の登録情報、オーダー、または返品に関する問い合わせや関心事に答えます。またジョン氏には顧客サービス・スーパーバイザーの役割もあり、前述の仕事をすべて担当することに加えて、他の顧客サービス担当者全員の承認と監視の許可権限もあります。

WebSphere Commerce システムには、以下のデフォルトの役割タイプの設定があります。

- ビジネス関係役割
- 顧客サービス役割
- マーケティング役割
- 運用役割
- 組織管理役割
- 商品管理および取引管理役割
- 技術操作役割

上記の個々の役割に関する詳細情報は、WebSphere Commerce オンライン・ヘルプのトピック『役割』を参照してください。サイト管理者は、組織エンティティーで

サイト管理者が作成する新しい役割のほかに、上記の役割も割り当てることができます。つまり、組織エンティティーに属しているユーザーが、その組織エンティティーに割り当てられている役割を果たすことができます。

ユーザーに役割が割り当てられると、組織エンティティーに役割の有効範囲が設定されます。これは任意の組織エンティティーであることができます。ユーザーの上位組織である必要はありません。しかし、役割は継承されるので、ユーザーは役割が割り当てられた組織の下位組織に対してその割り当てられた役割を実行できます。たとえば、ユーザーがルート組織で役割を付与された場合、ユーザーはその役割をすべての組織エンティティーに対して実行できます。

WebSphere Commerce の役割は、組織管理コンソールを使用して手作業で割り当てること、および登録とセッション管理コマンドを使用して自動的に割り当てることができます。自動化された役割の割り当ては、MemberRegistrationAttributes.xml ファイルに指定された構成に基づいています。 WebSphere Commerce 5.5 には MemberRegistrationAttributes.xml ファイルが備わっていて、これは特定の登録要件に適合するように変更することができます。自動化された役割の割り当ておよび MemberRegistrationAttributes.xml ファイルの詳細については、 WebSphere Commerce オンライン・ヘルプのトピック『MemberRegistrationAttributes XML および DTD ファイル』を参照してください。

WebSphere Commerce のメンバー資産の構造に関する詳細は、 WebSphere Commerce オンライン・ヘルプで『メンバー・オブジェクト・モデル』と『データ・モデル』を参照してください。

WebSphere Commerce の顧客資産について

*顧客*は、WebSphere Commerce 内でのユーザーです。顧客はストアのオンライン・カタログのブラウズ、オーダーの発行、買い物候補リストの作成、(一般連絡先、請求先、配送先などの) 住所の設定、およびストアやセラーからの購入を行うことができます。顧客はユーザーでもあります。以下の図は、顧客がストアからオーダーを発行するのに必要な資産を示しています。

上の図で示されているように、WebSphere Commerce システムにはメンバーが含まれています。個々のユーザーや組織エンティティー・メンバーに役割を割り当てることができます。

注: WebSphere Commerce では、メンバー は組織エンティティー、ユーザー、またはメンバー・グループのいずれかです。詳細については、 134 ページの『メンバー』を参照してください。

この場合、ユーザーは顧客です。顧客は、住所情報を提示しなければならず、買い物候補アイテムのリストを作成できます。上記の図では、メンバー (顧客) と、そのメンバーに関連付けられた顧客資産との間の相互の関係が示されています。顧客は住所を所有して提示しなければならず、買い物候補リストを作成してストアで買い物できます。住所と買い物候補リストは顧客に従属しています。

住所情報

顧客は、ストアから購入する際に、連絡先住所、請求先住所、および配送先住所の 3 種類の住所情報を提示しなければなりません。これらの住所タイプについて以下に説明します。個々の住所は固有にすることもできますし、同じにすることもできます。

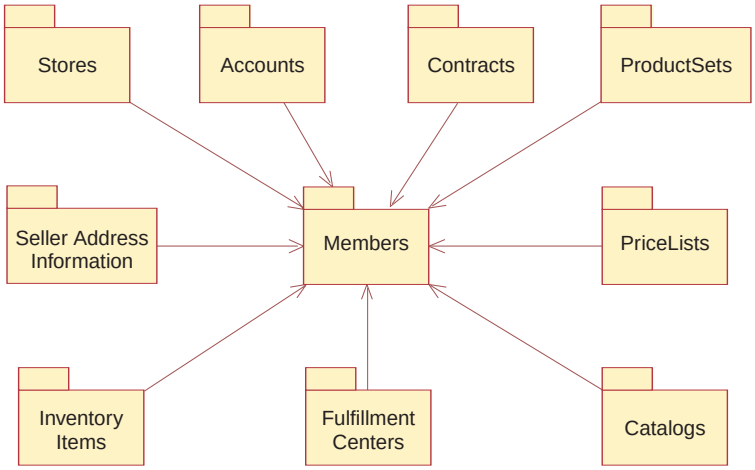
- 連絡先住所は、さまざまな目的で顧客に通知するのに使用されます。たとえば、オーダーの状況や変更内容、および近日中に行われるストアの行事 (販売促進活動やストアのメインテナンスなど) が含まれます。顧客の連絡先住所には、番地の名前と番号、市区町村、都道府県、郵便番号、国、 E メール・アドレス、電話番号、および FAX 番号が含まれます。通常、連絡先住所は、勤務先住所などの、最も容易に顧客と連絡が取れる場所にします。
- 請求先住所は、購入に対する勘定書または送り状を送るために使用されます。請求先住所には、番地の名前と番号、市区町村、都道府県、郵便番号、国、電話番号、および E メール・アドレスが含まれます。請求先住所は、連絡先住所や配送先住所と同じにすることも、違うものにすることも可能です。
- 配送先住所は、購入商品の配送に使用されます。配送先住所には、番地の名前と番号、市区町村、都道府県、郵便番号、国、電話番号、および E メール・アドレスが含まれます。配送先住所は、連絡先住所や請求先住所と同じでも違っていてもかまいません。

買い物候補リスト

ストアで*買い物候補リスト* をサポートできます。つまり、顧客が、買い物候補リストに、将来オーダーしたいと思う商品を追加します。買い物候補リストはショッピング・カートではありません。買い物候補リストには、複数のストアからのアイテムを含めることができますが、価格、配送先住所、配送モード、在庫納期情報、あるいは割引、配送料、税などの計算額は含まれません。

WebSphere Commerce のセラー資産について

セラーは、WebSphere Commerce 内でのユーザーです。セラーは、ストア・セールスの追跡に加え、全体的なストア目標と管理を監視します。セラーは、商品とサービスを顧客に販売します。セラーの役割はマーチャントと同じで、すべての WebSphere Commerce アクセラレーター機能に対するアクセス権があります。以下の図は、セラーがストアを保守し、顧客に対する販売を行うのに必要な資産を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、111 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、487 ページの『付録 A. UML の凡例』を参照してください。

上の図で示されているように、WebSphere Commerce システムにはメンバーが含まれています。各メンバーには、ストアの顧客サービス担当者やウェアハウスの受取人などの役割が割り当てられます。セラーの役割が顧客に対する販売を行うには、以下の資産を保守できます。

- ストア
- Business アカウント (オプション)
- 契約 (最低限 WebSphere Commerce のデフォルト契約)
- 商品セット
- 価格表
- カタログ
- 配送センター
- 在庫アイテム

上記の図では、メンバー (セラー) と、セラー資産との間の関係が示されています。つまり、セラーは上にリストされている資産を持ってストアを保守することができ、一方で資産はセラーによってデプロイされる必要があります。

ストア

WebSphere Commerce のオンライン・ストア は、 HTML および JavaServer Pages のファイルに加え、課税、配送、支払い、カタログ、およびその他のデータベース資産を持つ集合で構成されます。これらの集合はストア・アーカイブに入れられます。またストアには保管データもあります。このデータは WebSphere Commerce データベースに読み込まれる情報で、ストアはこのデータを使用して機能します。

WebSphere Commerce ストアの詳細については、 143 ページの『第 13 章 ストア資産』および 123 ページの『第 6 部 ストア・データの開発』を参照してください。

アカウント

Business

ストアでビジネス・アカウント をセットアップし、顧客がこれらのアカウントを使用してストアから購入できるようにすることができます。アカウントには、以下の情報があります。

- アカウント名。多くの場合、これは顧客に関連する組織の名前です。この組織はストアと契約を交わしており、そのストアで買い物をする顧客との取引条件を明記しています。たとえば IBM という組織が、ABC オフィス・サプライ社と契約を結ぶ場合などです。
- 担当者名。これは、アカウントを担当するセラーの組織内の担当組織の名前です。
- アカウントに属する契約の数。

WebSphere Commerce アカウントの詳細については、 206 ページの『アカウント (ビジネス・アカウント)』、および WebSphere Commerce オンライン・ヘルプを参照してください。

契約

通常 WebSphere Commerce では、すべての顧客は契約 の下で買い物しなければなりません。顧客とセラーとの間のアカウントごとに 1 つ以上の契約 (未登録の顧客か顧客がストアで買い物する場合、または顧客が他の契約の範囲外の商品を購入できるようにしたい場合は、少なくともデフォルト契約) と関連付けなければなりません。契約により、顧客は、指定された価格で、指定された期間に渡り、契約に定められている契約条件とビジネス・ポリシーに基づいて、ストアから商品を購入できます。セラーが契約をデプロイして、顧客がストアから購買できるようにします。

契約におけるバイヤーは、ユーザー、組織、またはメンバー・グループであることができます。ユーザーの場合、バイヤーは顧客として扱われます。契約でバイヤーとして定義された組織の場合、その組織の下位組織もその契約に関してバイヤーとして行動できます。メンバー・グループの場合、そのメンバー・グループ内のユーザーは契約に関してバイヤーとして行動できます。

セラーが使用できる WebSphere Commerce の契約とデフォルト契約の詳細については、 207 ページの『契約』を参照してください。

商品セット

商品セット には、セラーがオンライン・カタログを論理サブセットにカテゴリー化して、さまざまな顧客がさまざまなカタログ・ビューの利点を利用できるようにするための機構が備えられています。さらに、セラーは顧客用の契約を作成し、顧客が事前定義済みの商品セットの下の商品しか購入できないことを決定できます。

WebSphere Commerce の商品セットの詳細については、 167 ページの『商品セット』を参照してください。

価格表

価格表 は、セラーが顧客にオファーしたり提示したりする価格と関連付けられます。セラーは、同じ商品について、顧客によって異なる価格をリストできます。WebSphere Commerce では、価格のオファーは*取引位置* とも呼ばれ、カタログ・エントリーの価格と、その価格での購入資格を得るために顧客が満たす必要のある基準を表します。

WebSphere Commerce では、Offer オブジェクトは、メンバーが所有する、TradingPositionContainer の一部です。 TradingPositionContainer には、TradingPositionが含まれており、すべての顧客に利用可能にするか、または取引条件や契約によって特定のグループの顧客のみに利用可能にすることができます。TradingPositionContainer が価格表と呼ばれることもあります。価格表には 2 種類あります。 1 つは標準価格表で、ストア・カタログ内の商品の基本価格が含まれます。もう 1 つはカスタム価格表で、商品とそれらのカスタマイズ済みの価格のリストを指定します。

WebSphere Commerce の価格表の詳細については、 197 ページの『第 17 章 価格設定資産』を参照してください。

カタログ

WebSphere Commerce のストアでは、 1 つ以上のオンライン・*カタログ* を使用して、セラーが販売をオファーする商品やサービスを陳列します。通常、販売するアイテムのオンライン・カタログには価格、イメージ、および説明が含まれます。オンライン・カタログではまた、容易にナビゲーションできるよう、商品が明確なカテゴリーに分けて表示されている場合があります。

WebSphere Commerce システムのストアごとに、マスター・*カタログ* がなければなりません。マスター・カタログは、カタログの管理に使用されます。マスター・カタログを中心として、セラーの商品取引が管理されます。これは、すべての商品、アイテム、関係、およびストアで販売されるものすべての標準価格を含む 1 つのカタログです。セラーに複数のストアがある場合は、それらのストア間でマスター・カタログを共用できます。

WebSphere Commerce の商品セットの詳細については、 163 ページの『第 16 章 カタログ資産』を参照してください。

配送センター

配送センター は、ストアにより、在庫保管庫、および配送受取センターの両方として使用されます。 1 人のセラーには、1 つ以上の配送センターがあります。

WebSphere Commerce Server の側から見ると、 FulfillmentCenter オブジェクトは、Store オブジェクトから独立したものです。それは商品の在庫と配送を管理します。オーダーの発送において、配送センターは顧客が指定する ShippingMode オブジェクトに従います。 ShippingMode オブジェクトは、オーダーを実行するために、運送会社と配送方法を指示します。配送センター内で、ShippingArrangement オブジェ

クトは、ある特定の ShippingMode を使用して商品を配送するように、 Store オブジェクトが FulfillmentCenter オブジェクトと手配済みであることを示します。

WebSphere Commerce 配送センターの詳細については、 225 ページの『第 19 章 フルフィルメント資産』および 263 ページの『第 26 章 配送資産』を参照してください。

在庫アイテム

在庫アイテム には、セラーの配送センター内にある、物理的に報告できるすべてのものが含まれます。 WebSphere Commerce システムでは満たすべき特定の在庫タイプ (例えば、アイテム、商品、SKU、バンドル、パッケージなど) が定義されていますが、これらはすべて在庫と見なされます。 WebSphere Commerce アクセラレーターの商品管理ツールを使用して、商品が配送されるよう構成されます。

WebSphere Commerce 在庫アイテムの詳細については、 WebSphere Commerce オンライン・ヘルプおよび 305 ページの『第 29 章 在庫資産』を参照してください。

WebSphere Commerce の管理者資産について

管理者は、単に特定の管理活動を実行するための役割が割り当てられたユーザーまたはメンバーです。管理者に関連付けられる資産の詳細については、 133 ページの『WebSphere Commerce のメンバー資産について』を参照してください。

WebSphere Commerce でのメンバー資産の作成

セラー (ストア所有者として活動する組織) を作成して、そのセラーに関する情報を保守するには、WebSphere Commerce 管理コンソールを使用してください。詳細については、WebSphere Commerce オンライン・ヘルプのトピック『組織の作成』を参照してください。

管理者を作成するには、WebSphere Commerce 管理コンソールを使用してユーザーを作成してから、必要な役割をそのユーザーに割り当てます。詳細については、WebSphere Commerce オンライン・ヘルプのトピック『ユーザーの作成』および『ユーザー識別名による役割の割り当て』を参照してください。

ストア開発者は顧客を作成しません。顧客がストアに登録する際に、WebSphere Commerce システムに登録情報が徴収されて保守されます。

WebSphere Commerce に備わっているサンプル・ストアのそれぞれには、MemberRegistrationAttributes.xml ファイルの独自のバージョンが含まれています。それは登録とセッション管理コマンドに関して自動化された役割の割り当てを構成するために使用します。組織構造の変更したい場合、または役割の割り当てに関して特定の要件がある場合は、このファイルを変更する必要があります。このファイルおよびそれを特定の必要に合わせて構成する方法の詳細については、 WebSphere Commerce オンライン・ヘルプのトピック『MemberRegistrationAttributes XML および DTD ファイル』を参照してください。

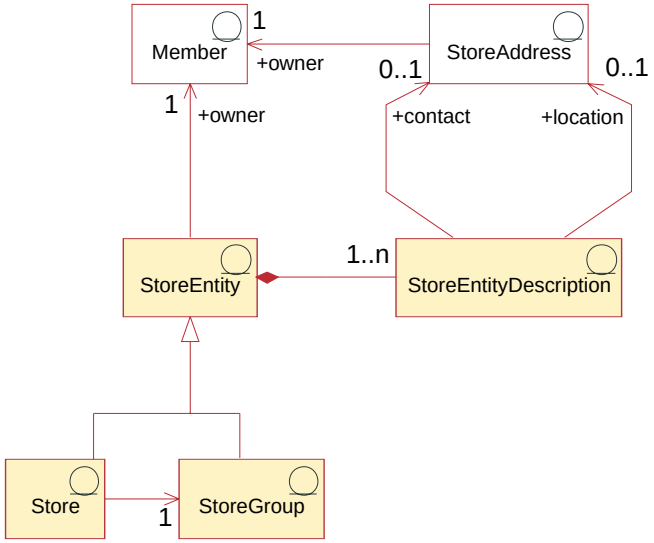
第 13 章 ストア資産

WebSphere Commerce でストアを作成するには、まず以下のものをデータベースに作成する必要があります。

- ストア
- ストアが属するグループ
- ストアまたはストア・グループの二者を表す、抽象ストア・エンティティー・オブジェクト。

WebSphere Commerce のストア資産について

以下の図は、WebSphere Commerce Server 内のストア資産を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、111 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、487 ページの『付録 A. UML の凡例』を参照してください。

ストア・エンティティー

ストア・エンティティー は、ストアまたはストア・グループのいずれかを表す抽象スーパークラスです。

ストア・エンティティーには 1 人の所有者 (メンバー) がいます。メンバーの詳細については、133 ページの『WebSphere Commerce のメンバー資産について』を参照してください。

ストア・エンティティーの説明
ストア・エンティティーの説明 は、ストア・エンティティーについて記述します。ストア・エンティティーには、説明が含まれることがあります。ストアが複数の言語をサポートしている場合、ストア・エンティティーの説明が複数の言語のことがあります。説明には、ストア・エンティティーの連絡先住所が 1 つと、ストア・エンティティーのロケーションの住所が 1 つ含まれる場合があります。

ストア
ストア はストア・エンティティーです。ストアは、1 つのストア・グループに属していなければなりません。

ストア・グループ
ストア・グループ はストアの集合です。ストア・グループはストア・エンティティーです。ストア・グループは、共通情報 (ストア・グループ・レベルで保管でき、ストア・グループ内のすべてのストアで共有できるもの) のコンテナーとして機能します。たとえば、同じストア・グループ内のストアは、課税カテゴリー、サポートされる言語、サポートされる通貨、計算コード、および配送管轄区域などの情報を共有できます。

現在のところ、WebSphere Commerce Server に存在し、サイト管理レベルで保守できるストア・グループは 1 つだけです。

WebSphere Commerce Server のストア資産の構造の詳細については、WebSphere Commerce オンライン・ヘルプの『ストアのオブジェクト・モデル』と『データ・モデル』を参照してください。

WebSphere Commerce でのストア資産の作成
WebSphere Commerce アクセラレーターのストア・ツールを使用すると、以下のストア資産を作成または編集することができます。

- 連絡先資産内のストア ID とメンバー ID
- STOREENT テーブル内のストア ID
- STORE テーブル内のストア・ディレクトリー
- STADDRESS 内の住所のニックネーム
- ストアの説明
- ストアのアドレス

結果として、以下の 2 つの方法でストア資産を作成できます。

- WebSphere Commerce に付属のサンプル・ストアの 1 つにある、既存のストア資産を編集する。
- XML ファイル形式でストア資産を作成する。XML ファイルは、ストア・アーカイブの一部として発行することもできますし、ローダー・パッケージを使用してロードすることもできます。

XML ファイル形式でストア資産を作成することに関する情報は、 145 ページの『XML ファイルによるストア・データ資産の作成』を参照してください。

WebSphere Commerce アクセラレーターを使用したストアの編集の詳細については、 WebSphere Commerce オンライン・ヘルプを参照してください。

XML ファイルによるストア・データ資産の作成

ストア資産を XML ファイルの形式で作成します。XML ファイルは、ローダー・パッケージを使用して、データベースにロード可能です。グローバル・ストアを作成する場合は、ストアでサポートされているロケールごとに別々の XML ファイルを作成することもできます。説明情報はすべてロケール固有のファイルに指定されるので、これを翻訳するのは容易です。ローダー・パッケージの詳細については、363 ページの『第 10 部 ストアの発行』を参照してください。

サンプル・ストア (以下のタスクの多くの例がサンプル・ストアから取られています) では、翻訳の不要な情報はすべて 1 つの store.xml ファイルに指定されており、翻訳の必要な情報は、そのストアがサポートするロケールごとに別の store.xml ファイルに指定されています。ロケール固有のファイルには、すべての説明情報が含まれています。

ストア資産を作成するには、以下のようにします。

1. 429 ページの『第 38 章 WebSphere Commerce データベース資産グループのロード』に記載されている情報を確認します。
2. サンプル・ストアのストア資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。

   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

   - *WC_installdir*/samplestores

     注: 「*WebSphere Commerce サンプル・ストア・ガイド*」には、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

   各サンプル・ストアには、store.xml ファイルがいくつか組み込まれています。これには言語別のストア情報が含まれています。サンプル・ストアは複数の言語に翻訳されているため、各ストアには複数の store.xml ファイルが存在します。ストア・アーカイブの store.xml ファイルを表示するには、ZIP プログラムを使用してストア・アーカイブを解凍します。 store.xml ファイルは、データ・ディレクトリーにあります。言語ごとの store.xml は、データ・ディレクトリーのロケールごとのサブディレクトリーにあります。
3. 489 ページの『付録 B. データの作成』に記載されている情報を確認します。
4. サンプル・ストア・アーカイブの store.xml ファイルの 1 つをコピーするか、新しいファイルを作成することにより、store.xml ファイルを作成します。詳細については、wcs.dtd ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。

   - *WC_installdir*/schema/xml
5. ストア・エンティティーを作成します。

   a. 次の例を参考にして、XML ファイルの STOREENT テーブルにストア・エンティティーを定義します。

```
<storeent
 storeent_id="@storeent_id_1"
 member_id="&MEMBER_ID"
```

```
 type="S"
 identifier="ToolTech"
 setccurr="USD"
/>
```

ここで

- storeent_id は、生成される固有キーです。
- member_id は、ストア・エンティティーの所有者です。
- type は、ストア・エンティティーの種類で、 G = StoreGroup、S = Store です。
- identifier は、所有者と共にストア・エンティティーを固有に識別するストリングです。
- setccurr は、ストア・エンティティーのデフォルト通貨です。言い換えると、希望する通貨を持たない顧客によって使用される通貨です。ストアに関してこれが NULL の場合、デフォルト通貨はストア・グループから取得されます。

6. ストアの住所を作成します。
   a. 次の例を参考にして、 XML ファイルの STADDRESS テーブルにストアの連絡先 (複数の場合もあり) を作成します。グローバル・ストアを作成する場合は、ロケール固有の XML ファイルにこの情報を組み込む必要があります。

```
<staddress
 staddress_id="@staddress_id_en_US_1"
 member_id="&MEMBER_ID"
 nickname="storeaddress_English"
 address1="12xx Martindale Avenue"
 address2="Suite 9xx"
 businesstitle="ToolTech"
 city="Toolsville"
 state="Ontario"
 zipcode="Lxx 1xx"
 country="Canada"
 phone1="1-800-555-1234"
 fax1="1-800-555-4321"
 email1="info@tooltech.xxx"
 />
```

ここで

- staddress_id は、生成される固有キーです。
- member_id は、ストア・エンティティーの所有者です。

7. ストア・エンティティーの説明を作成します。
   a. 次の例を参考にして、 XML ファイルの STOREENTDS テーブルにストア・エンティティーの説明を作成します。グローバル・ストアを作成する場合は、ロケール固有の XML ファイルにこの情報を組み込む必要があります。

```
<storeentds
 description="Commerce Models Store entity"
 language_id="&en_US"
 displayname="ToolTech"
 storeent_id="@storeent_id_1"
 staddress_id_cont="@staddress_id_en_US_1"
 staddress_id_loc="@staddress_id_en_US_1"
```

ここで

- description は、顧客に対して表示するのに適した、ストア・エンティティーの詳細説明です。

- language_id は、ストアでショッピングする顧客に対して表示される情報のためのデフォルト言語です。
- displayname は、顧客に対して表示するのに適した、ストア・エンティティーの簡略説明です。
- storeent_id は、ストア・エンティティーです。
- staddress_id_cont は、StoreEntity の連絡先住所です。
- staddress_id_loc は、StoreEntity の物理的場所です。

8. データベースにストアを作成します。
   a. 次の例を参考にして、 XML ファイルの STORE テーブルにストアを定義します。

```
<store
store_id="@storeent_id_1"
directory="ToolTech"
ffmcenter_id="@ffmcenter_id_1"
language_id="&en_US"
storegrp_id="-1"
allocationgoodfor="43200"
bopmpadfactor="0"
defaultbooffset="2592000"
ffmcselectionflags="0"
maxbooffset="7776000"
rejectedordexpiry="259200"
rtnffmctr_id="@ffmcenter_id_1"
pricerefflags="0"
storetype="B2B"
/>
```

   ここで
   - store_id は、生成される固有キーです。
   - directory は、ストア固有の Web 資産があるディレクトリーです。そのディレクトリーは、Store.war Web モジュールの文書ルートの下にあります。
   - ffmcenter_id は、ストアのデフォルトの配送センターです。
   - language_id は、ストアでショッピングする顧客に対して表示される情報のためのデフォルト言語です。
   - storegrp_id は、そのストアが関連付けられているストア・グループです。この数値は、STOREGRP テーブルに生成されます。
   - allocationgoodfor は、割り振りが行われてから長時間経過した場合に、ATP 在庫割り振りを取り消すために ReleaseExpiredAllocations スケジューラー・ジョブを使用できることを意味します。
   - bopmpadfactor は、このストアが異なる配送センターごとに (税金または配送料のような) オーダー金額を計算する場合には、配送センターがバックオーダー済みのアイテムに最終的に割り振られる際に、以前に送信済みのオーダーのオーダー金額を変更できることを意味します。必要であるなら、 Payment Manager に提供されるオーダー金額によるパーセントを表すこの埋め込み係数を増加させることができます。たとえば、5 を指定すると 5 % 増加できます。
   - defaultbooffset は、バックオーダーされた OrderItem の販売予定時期が判別できない場合、指定した秒数後に設定されることを意味します。

- maxbooffset は、バックオーダーされた OrderItem の販売予定時期が通常では将来さらに遅くなってしまう場合に、指定した秒数後に設定されることを意味します。
- rejectedordexpiry は、支払状況が拒否状況にある時間がこの秒数を超えると、オーダーがキャンセルされることを示します。
- rtnffmctr_id は、商品をストアに返品するためのデフォルトの配送センターです。
- pricerefflags には、GetContractUnitPrices タスク・コマンドのデフォルト・インプリメンテーションによって価格が更新される際に、どの TradingAgreements およびオファーが検索されるかを制御するビット・フラグが入ります。
  - 1 = usePreviousOnly。OrderItems によって参照されるものを使用します。それらが使用できなくなると、障害が発生します。
  - 2 = usePreviousOrSearchAgain。usePreviousOnly と同じですが、それらが使用できなくなった場合、障害が発生するのではなく、ORDIOFFER および ORDITRD テーブルに保管されているものを検索します。
  - 4 = alwaysSearchAgain。常に ORDIOFFER および ORDITRD テーブルに保管されているものを検索します。
- storetype は、ユーザー・インターフェースが使用する以下のいずれかのストア・タイプを示します。 StoreType に応じて、ユーザー・インターフェースは適切な機能を提供することになります。
  - B2B = 企業向け
  - B2C = 企業消費者間取引 (消費者向け)
  - CHS = 販売店ハブ (Commerce プラザ)
  - CPS = マスター・カタログ・プロファイル・ストア (カタログ資産ストア)
  - RHS = 販売店ホスト・ストア
  - RPS = 販売店プロファイル・ストア (販売店ストアフロント資産ストア)
  - DPS = ディストリビューター・プロファイル・ストア (ディストリビューター資産ストア)
  - DPX = ディストリビューター・プロキシー・ストア
  - HCP = コマース・ホスティング・ハブ (ホスティング・ハブ)
  - PBS = ストア・ディレクトリー
  - MPS = マーチャント・プロファイル・ストア (ホスティング・ストアフロント資産ストア)
  - MHS = マーチャント・ホスト・ストア
  - SCP = サプライヤー・ハブ
  - SPS = サプライヤー・プロファイル・ストア (サプライヤー資産ストア)
  - SHS = サプライヤー・ホスト・ストア

  注: 括弧で示されている名前は、WebSphere Commerce に付属の、対応するサンプルの名前です。

9. ストアのサポートされる言語を定義します。

a. 次の例を参考にして、ストアのサポートされる言語を XML ファイルに定義して、情報を STORELANG テーブルに追加します。ストアがマルチリンガル・サポートされている場合には、この情報をロケール固有の XML ファイルに含める必要があります (ストアがサポートする言語ごとに 1 つずつ)。

```
<storelang
 language_id="&en_US"
 storeent_id="@storeent_id_1"
 />
```

ここで

- language_id は、ストア・エンティティーによってサポートされる言語です。
- storeent_id は、ストア・エンティティーです。

b. 次の例を参考にして、言語に関する情報を STORELANGDS テーブルに追加します。ストアがマルチリンガル・サポートされている場合には、この情報をロケール固有の XML ファイルに含める必要があります (ストアがサポートする言語ごとに 1 つずつ)。

```
<storlangds
  description="United States"
  language_id="&en_US"
  storeent_id="@storeent_id_1"
  language_id_desc="&en_US"
 />
```

ここで

- description は、選択リストの中で顧客に対して表示するのに適した、言語の簡略説明です。
- language_id は、説明の言語です。
- storeent_id は、言語をサポートするストア・エンティティーです。
- language_id_desc は、説明される言語です。

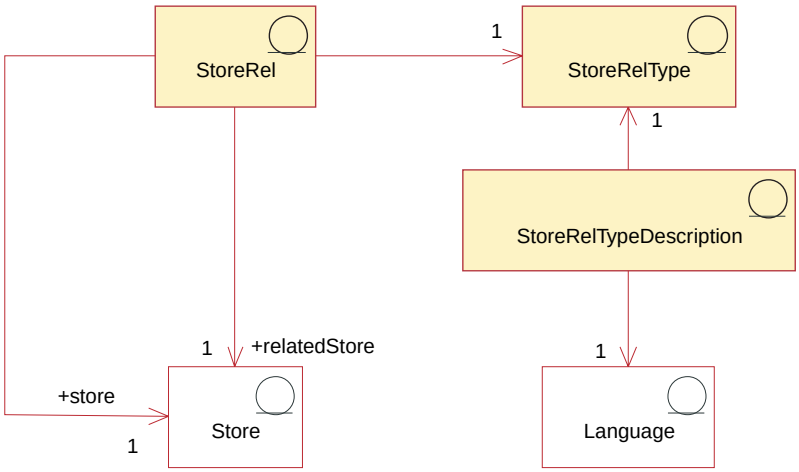
@ と **&** の使用法の詳細については、 489 ページの『付録 B. データの作成』を参照してください。

第 14 章 ストア間の関係

Business WebSphere Commerce は、サイト内のストア間でのいくつかのタイプの関係をサポートしています。たとえば、あるストアが別のストアにホスティング・サービスを提供したり、あるストアが別のストアにより提供されたカタログまたは通貨資産を使用することができます。

WebSphere Commerce でのストア間の関係について

以下の図は、WebSphere Commerce Server でのストア関係を示しています。



ストア関係

ストア関係 (StoreRel テーブルでキャプチャーされた) は、2 つのストア間の関係です。すべてストア関係には方向性があり、それぞれのストア関係では、 1 つのストアがサービスを提供し、その関係内の 2 番目のストアはそれらのサービスを使用します。たとえば、ストア A は、ストア B により提供されたカタログを使用します。

それぞれのストア関係には、1 つのストア関係タイプ (StoreRelType) があります。

ストア関係タイプ

ストア関係タイプ (StoreRelType) は、2 つのストア間の関係のタイプを定義します。ストア関係の各タイプは、それ自体の関係を定義しています。すなわち、関係がある各パートナーの果たす役割、および 2 つパートナー間にどのような関係があるかを定義しています。

WebSphere Commerce によりサポートされるストア関係タイプ

WebSphere Commerce はストア間のいくつかの関係タイプをサポートします。WebSphere Commerce で提供されるデフォルトの関係タイプは、大まかに以下の 2 つのカテゴリーにグループ分けできます。

- あるストアが別のストアにデータ資産を提供する関係。たとえば、ストア A がストア B で使用されるカタログ・データを提供します。
- あるストアが別のストアとの間に「ビジネス関係」を持つ関係。すなわち、あるストアが、別のストアにホスティングしたり、ショッピング・カートを別のストアに転送したりできます。

あるストアが別のストアにデータ資産を提供する関係:

表 8.

関係タイプ	説明	詳細の参照先
com.ibm.commerce. businessPolicy	あるストアが別のストアで定義されたビジネス・ポリシーを使用します。	205 ページの『第 18 章 契約資産』
com.ibm.commerce. campaigns	あるストアが別のストアで定義されたキャンペーンを使用します。	233 ページの『第 20 章 キャンペーン資産』
com.ibm.commerce.catalog	あるストアが別のストアで定義されたカタログ・データを使用します。	163 ページの『第 16 章 カタログ資産』
com.ibm.commerce.command	あるストアが別のストアで定義されたコマンドを使用します。	157 ページの『第 15 章 コマンド、表示、および URL レジストリー・データ』
com.ibm.commerce.price	あるストアが別のストアで定義された価格データを使用します。	197 ページの『第 17 章 価格設定資産』
com.ibm.commerce. segmentation	あるストアが別のストアで定義された顧客プロファイル・データを使用します。	323 ページの『第 32 章 顧客プロファイル』
com.ibm.commerce.URL	あるストアが別のストアで定義された URL を使用します。	157 ページの『第 15 章 コマンド、表示、および URL レジストリー・データ』
com.ibm.commerce.view	あるストアが別のストアで定義されたビューを使用します。	157 ページの『第 15 章 コマンド、表示、および URL レジストリー・データ』
com.ibm.commerce.storeitem	あるストアが別のストアで定義されたアイテムを使用します。	305 ページの『第 29 章 在庫資産』
com.ibm.commerce. propertyFiles	あるストアが別のストアで定義されたプロパティー・ファイルを使用します。	
com.ibm.commerce.currency. conversion	あるストアが別のストアで定義された通貨換算率を使用します。	249 ページの『第 23 章 通貨資産』

表 8. (続き)

com.ibm.commerce.currency.supported	あるストアが別のストアでサポートされる通貨を使用します。	249 ページの『第 23 章 通貨資産』
com.ibm.commerce.currency.format	あるストアが別のストアで定義された通貨形式を使用します。	249 ページの『第 23 章 通貨資産』
com.ibm.commerce.currency.countervalue	あるストアが別のストアで定義された通貨カウンター値を使用します。	249 ページの『第 23 章 通貨資産』
com.ibm.commerce.measurement.format	あるストアが別のストアで定義された単位を使用します。	257 ページの『第 24 章 計測単位資産』

1 つのストアは、複数のストアと関係を持てます。すなわち、ストア A は、ストア B、C、および D のカタログ・リソースを使用する必要があります。複数のストアとの間のそうした関係を促進するため、ストアの順序を指定して、他のストアはその順序で資産を使用する必要があります。ストア間での関係の順序付けは、以下のように機能します。

- オーバーライド: ストア関係が順序付けのオーバーライド・メソッドの後に続く場合、最も小さいシーケンス番号のストア関係が使用されるストア関係です。以下のストア関係がオーバーライド・メソッドを使用します。
  - コマンド
  - 通貨
  - 測定
  - 価格
  - プロパティー・ファイル
  - storeitem
  - URL
  - ビュー
- マージ: ストア関係の後に順序付けのマージ・メソッドが続く場合、 WebSphere Commerce は、そのストアと関連付けられたすべてのストア関係を検索し、関連したすべてのストアからのデータをマージします。以下のストア関係がマージ・メソッドを使用します。
  - ビジネス・ポリシー
  - キャンペーン
  - カタログ
  - セグメンテーション

すべてのデフォルトのストア関係タイプは、順序付けでオーバーライド・メソッドかマージ・メソッドを使用するように指定されています。

注: 1 つのストア関係タイプが複数の契約に対して存在することはありませんが、1 つの契約を複数のストアに展開することはできます。詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

あるストアが別のストアとの間に「ビジネス関係」を持つ関係:

表 9.

関係タイプ	説明
com.ibm.commerce. hostedStore	ハブ・ストアは、販売店、サプライヤー、またはホストされるストアをホスティングします。
com.ibm.commerce. referral	ハブ・ストアは、複数のディストリビューターと参照関係にあります。ハブ・ストアは、ショッピング・カートをディストリビューターのストアに転送することができます。通常、ショッピング・カートを受け取るストアは、外部システム用のプロキシー・ストアです。
com.ibm.commerce. channelStore	あるストアが別のストア用のハブ・ストアとして動作します。この関係は、ストア・ディレクトリーとホスティング・ハブの関係を定義します。

ストア関係タイプの説明

ストア関係タイプの説明 は、関係のタイプを説明します。それぞれのストア関係タイプの説明は、1 つの関係タイプのみを説明します。ストア関係タイプの説明は、複数の言語で使用できる可能性があります。

WebSphere Commerce でのストア関係の作成

ストア関係を XML ファイルの形式で作成します。XML ファイルは、ローダー・パッケージを使用して、データベースにロード可能です。ローダー・パッケージの詳細については、 363 ページの『第 10 部 ストアの発行』を参照してください。

注: ストア作成ウィザードを使用してホストされるストアを作成する場合 (詳しくは 76 ページの『「ストア作成 (Store Creation)」ウィザード』を参照してください)、またはサービス契約を使用してディストリビューター・プロキシー・ストアを作成する場合 (詳しくは 77 ページの『プロキシー・ストアの作成』を参照してください)、これらのストア関係の多くが作成されます。

ストア関係資産を作成するには、以下のようにします。

1. 429 ページの『第 38 章 WebSphere Commerce データベース資産グループのロード』に記載されている情報を確認します。
2. サンプル・ストアのストア資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。

   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

   - *WC_installdir*/samplestores

     注: 「*WebSphere Commerce サンプル・ストア・ガイド*」には、サンプル・ストアに含まれる各データ資産についての情報が記載されています。
3. 489 ページの『付録 B. データの作成』に記載されている情報を確認します。

4. サンプル・ストア・アーカイブの storerelation.xml ファイルの 1 つをコピーするか、または新しいファイルを作成することにより、storerelation.xml ファイルを作成します。詳細については、wcs.dtd ファイルを参照してください。DTD ファイルは以下のディレクトリーにあります。
   - *WC_installdir*/schema/xml
5. ストア関係を作成します。
   a. 次の例を参考にして、XML ファイルの STOREREL テーブルにストア・エンティティーを定義します。

```
<storerel
   store_id="@storeent_id_1"
 relatedstore_id="@storeent_id_2"
 streltype="-4"
 sequence="0"
 state="1"
/>
```

   ここで、
   - store_id は、関連したストアのサービスを使用する 1 次ストアです。
   - relatedstore_id は、1 次ストアによって使用されるサービスを提供するストアです。
   - streltype は、関係のタイプです。デフォルトの関係タイプは以下のとおりです。
     - -1 com.ibm.commerce.businessPolicy
     - -3 com.ibm.commerce.campaigns
     - -4 com.ibm.commerce.catalog
     - -5 com.ibm.commerce.command
     - -6 com.ibm.commerce.hostedStore
     - -7 com.ibm.commerce.price
     - -8 com.ibm.commerce.referral
     - -9 com.ibm.commerce.segmentation
     - -10 com.ibm.commerce.URL
     - -11 com.ibm.commerce.view
     - -13 com.ibm.commerce.inventory
     - -14 com.ibm.commerce.storeitem
     - -15 com.ibm.commerce.channelStore
     - -16 com.ibm.commerce.propertyFiles
     - -17 com.ibm.commerce.currency.conversion
     - -18 com.ibm.commerce.currency.format
     - -19 com.ibm.commerce.currency.supported
     - -20 com.ibm.commerce.currency.countervalue
     - -21 com.ibm.commerce.measurement.format
   - sequence は、同一の関係タイプに対して複数の関連ストアが定義されているときの選択順序を定義します。デフォルトは 0 です。
   - state は、関係の状態です (0 = 非アクティブ、1 = アクティブ)。デフォルトは 1 です。

@ と **&** の使用法の詳細については、 489 ページの『付録 B. データの作成』を参照してください。

第 15 章 コマンド、表示、および URL レジストリー・データ

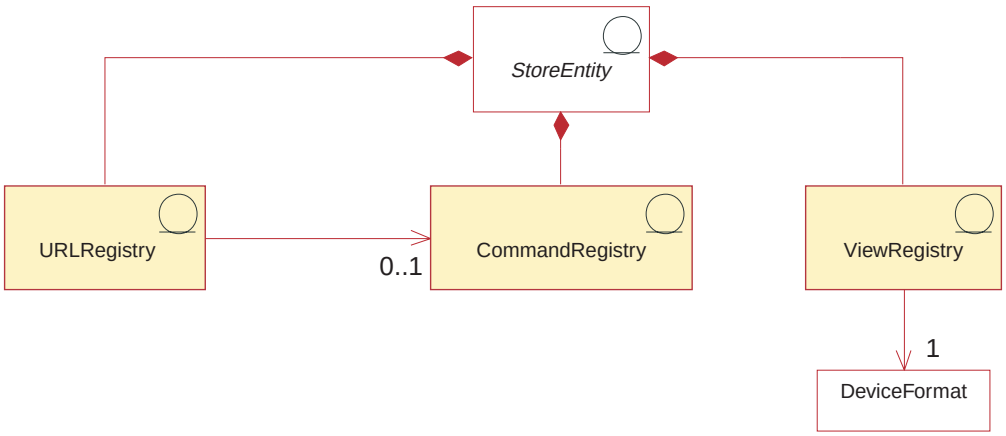
コマンド、表示、および URL レジストリーは、WebSphere Commerce コマンド・フレームワークの一部です。 WebSphere Commerce コマンド・フレームワークについては、「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」の 1 章、『概要』、 2 章『デザイン・パターン』、および 6 章『コマンドのインプリメンテーション』でさらに詳細に説明されています。コマンド、表示、および URL レジストリーをどのように情報モデルに適合させるかを理解するために、ここではその概要を示します。

WebSphere Commerce Server のコマンドおよびビュー資産の構造の詳細については、 WebSphere Commerce オンライン・ヘルプでコマンドおよびビュー・データ・モデルを参照してください。

WebSphere Commerce におけるコマンド、ビュー、および URL レジストリーについて

WebSphere Commerce コマンド・フレームワークは、実行されたコマンドによって戻されたビューに基づいて、どのようにコマンドを実行し、その後応答を戻すかを決定します。コマンドの実行および応答は、ストアに依存しています。これは、同じコマンドはストアごとに異なる仕方でインプリメントされ、ストアごとに異なる応答を戻すということを意味します。

以下の図は、WebSphere Commerce Server のコマンド、表示、および URL レジストリー構造を示しています。



URL レジストリー

URL レジストリーは、実行されるコマンドの実際のインターフェースにコマンド名をマップします。それぞれの URL レジストリー・エントリーは、ストアを区別します。すなわち、各ストアは、同じ URL 値に対して異なるインターフェースを定義できます。 URL レジストリーのストア・バージョンが見つからない場合には、

サイトに対して定義された URL レジストリー (ストア 0) が使用されます。デフォルトでは、すべての URL レジストリーは、サイトに対して定義されています。

Business 1 つのストアで定義され、登録された URL は、他のストアでも使用できます。あるストアが、別のストアで定義された URL を使用するには、ストア間でタイプ com.ibm.commerce.URL のストア関係を作成する必要があります。詳細については、151 ページの『第 14 章 ストア間の関係』を参照してください。

コマンド・レジストリー

コントローラー・コマンドであってもタスク・コマンドであっても、すべてのコマンドは、コマンド・レジストリーで定義できます。コマンドがコマンド・レジストリーで定義された場合、その定義は、コマンドの実行時にコマンド・インプリメンテーションとして使用されます。コマンドがコマンド・レジストリーで定義されなかった場合、デフォルトのインプリメンテーションが代わりに使用されます。すべてのコマンド・インターフェースには、コマンドがコマンド・レジストリーで定義されなかった場合に使用されるデフォルトのインプリメンテーションが割り当てられます。

コマンドがサイト・レベルのコマンド (ストア 0) としてコマンド・レジストリーに定義されている場合、コマンドの別のインプリメンテーションが定義されているストアにコマンドを実行した場合を除いて、サイト・レベルのインプリメンテーションが使用されます。

コマンド・レジストリーは、異なるストアが同じコマンドを使用しても、コマンドの元の流れを変更せずにインプリメンテーションの一部またはすべてを拡張できます。

Business 1 つのストアで定義されて登録されたコマンドは、他のストアでも使用できます。あるストアが、別のストアで定義されたコマンドを使用するには、ストア間でタイプ com.ibm.commerce.command のストア関係を作成する必要があります。詳細については、151 ページの『第 14 章 ストア間の関係』を参照してください。

ビュー・レジストリー

コマンドの実行後は、ほとんどの場合に、コマンド要求発行者に応答が戻されることが求められます。応答を決定する時に、コマンド・フレームワークは以下のエレメントを考慮します。

- コマンドの実行後に応答プロパティーで検出されるビュー。
- コマンドを実行する対象となったストア。
- 要求がなされた時の、要求のデバイス形式。

応答を戻すすべてのビューは、ストアごとに、またはデフォルトではサイトにおいて、ビュー・レジストリーで定義される必要があります。各ストアは通常、着信要求が取ることができるデバイス形式ごとにビューを定義します。ただし、ビューがストアによって定義されていない場合、サイトのデフォルトのビューが使用されます。要求を処理するアダプターは、どのビューを呼び出すかを判断する場合に、どのデバイス形式およびデフォルト・デバイス形式を使用するかを決定します。単一

の汎用デバイス形式がないため、WebSphere Commerce により受け入れられる要求のさまざまなタイプに基づいて、デバイス形式ごとに定義されるビューがある場合もあります。

Business 1 つのストアで定義され、登録されたビューは、他のストアでも使用できます。あるストアが、別のストアで定義されたビューを使用するためには、ストア間でタイプ com.ibm.commerce.view のストア関係を作成する必要があります。詳細については、151 ページの『第 14 章 ストア間の関係』を参照してください。

新規コマンド、ビュー、および URL の作成

WebSphere Commerce Server インスタンスを作成すると、WebSphere Commerce に付属するデフォルトのコマンド、ビュー、および URL が、WebSphere Commerce Server データベースの対応するテーブル (CMDREG、VIEWREG、および URLREG) に登録されます。これらのコマンド、ビュー、および URL は、インスタンス内に存在するすべてのストアで使用することができます。

WebSphere Commerce は、デフォルト・ビューを表示するデフォルト JSP ファイルも提供します。これらの JSP ファイルは、VIEWREG テーブルのビューと関連しています。

新しいコマンド、ビュー、または URL を作成する場合、または既存のものをカスタマイズする場合は、対応するデータベース・テーブル (CMDREG、VIEWREG、および URLREG) にそれらを登録しないとストアで使用可能になりません。自分のストアで使用するために新しい JSP ファイルを作成する場合は、それらのファイルを VIEWREG テーブルの中の対応するビューと関連付ける必要があります。

注: 新しい JSP ファイルを作成するものの、それにビューと関連したデフォルトの JSP ファイルと同じ名前を付ける場合は、新しい JSP ファイルを VIEWREG テーブルに登録する必要はありません。

注: 新規ビューを作成する場合は、必ずアクセス制御ポリシーをそれぞれの新規ビューに関連付けてください。詳細については、331 ページの『ストアへのアクセス制御の追加』を参照してください。

コマンド、ビュー、または URL の作成またはカスタマイズの詳細については、「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」を参照してください。「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」には、コマンド、ビュー、URL、および JSP ファイルを登録する方法と、いつ登録するべきかに関する情報も含まれています。

WebSphere Commerce でのコマンド、ビュー、および URL の登録

ストアのために複数の新しいコマンド、ビュー、URL、または JSP ファイルを作成あるいはカスタマイズしたなら、それを登録するために XML ファイルを使用することも (XML ファイルは、後でローダー・パッケージを使用してデータベースにロードできます)、それをストア・アーカイブの一部として登録することもできます。ストア・アーカイブは、管理コンソールにある発行ユーティリティーを使用して発行できます。ローダー・パッケージの詳細については、363 ページの『第 10 部 ストアの発行』を参照してください。

注: 新規またはカスタマイズ済みコマンドをロードするための XML ファイルを作成する前に、「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」に載せられているコマンドの動作に関する詳細情報を参照してください。

コマンド、ビュー、および URL の登録用 XML ファイルの作成

ストア用の新しいコマンド、ビュー、および JSP ファイルを登録するための XML ファイルを作成するには、次の手順に従います。

1. 429 ページの『第 38 章 WebSphere Commerce データベース資産グループのロード』に記載されている情報を確認します。
2. サンプル・ストア用のコマンド、ビュー、JSP ファイルを登録するのに使用される XML ファイルを確認します。各サンプル・ストアには、command.xml ファイルが入っており、このファイルには登録情報が入っています。ストア・アーカイブ・ファイルは以下のディレクトリーにあります。
   - *WC_installdir*/samplestores

     注: 「*WebSphere Commerce サンプル・ストア・ガイド*」には、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

   ストア・アーカイブの内容を表示するには、解凍プログラムを使います。command.xml ファイルは、データ・ディレクトリーにあります。
3. 489 ページの『付録 B. データの作成』に記載されている情報を確認します。
4. サンプル・ストア・アーカイブの command.xml ファイルの 1 つをコピーするか、または新しいファイルを作成することにより、command.xml ファイルを作成します。詳細については、wcs.dtd ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。
   - *WC_installdir*/schema/xml
5. コントローラー・コマンドは、URLREG テーブルと CMDREG テーブルに登録する必要があります。 URLREG テーブルに新規またはカスタマイズ済みのコントローラー・コマンドを登録するには、次の例を指針として、新しいカスタマイズ済みのコントローラー・コマンドごとに、XML ファイルにエントリーを作成します。

   ```
   <urlreg
   url="MyProductDisplay"
   storeent_id="@storeent_id_1″
   interfacename="com.mystore.commerce.catalog.commands.ProductDisplayCmd"
   https="0"
   description="Product display command for my store"
   authenticated="0"
   internal="0" />
   ```

   ここで、
   - urlreg は、この情報を読み込むデータベース・テーブル (URLREG) の名前です。
   - url は URI 名です。
   - storeent_id はストア・エンティティー ID です。@ 記号を使用するのは、内部別名解決法と呼ばれます。内部別名解決法を使用する場合は、 XML 文

書内で基本キー (ID) の代わりに別名が用いられます。これで別名は、そのエレメントを参照するために、XML ファイル内の他の場所で使用できます。したがって、XML ファイルを構築するのに必要な固有索引を知っている必要はありません。発行中、ID リゾルバーは @ 記号を固有な値に置き換えます。詳細については、 489 ページの『付録 B. データの作成』を参照してください。

- interfacename はコントローラー・コマンド・インターフェース名です。
- https はこの URL 要求に必要なセキュア HTTP です。セキュア HTTP が必要な場合は 1、必要ない場合は 0 を使用します。
- authenticated は、この URL 要求にユーザー・ログオンが必要かどうかを示します。認証が必要な場合は 1、必要ない場合は 0 を使用します。
- internal は、コマンドが WebSphere Commerce にとって内部的なものかどうかを示します。内部 URL は、WebSphere Commerce のツールで使用されます。内部的なものである場合は 1、外部的なものである場合は 0 を使用します。自分で作成する URL は外部的なものでなければなりません。

6. CMDREG テーブルに新規のコントローラー・コマンド、または新規のタスク・コマンドを登録するには、次のタスク・コマンドの例 (ToolTech サンプル・ストア command.xml ファイルからの例) を参考にして、新規の、またはカスタマイズ済みのコントローラーまたはタスク・コマンドごとに、XML ファイルにエントリーを作成します。

```
< cmdreg
storeent_id="@storeent_id_1"
interfacename="com.ibm.commerce.payment.commands.DoPaymentCmd"
classname="com.ibm.commerce.payment.commands.DoPaymentMPFCmdImpl"/>
```

ここで、

- cmdreg は、この情報を読み込むデータベース・テーブル (CMDREG) の名前です。
- storeent_id はストア・エンティティー ID です。@ 記号を使用するのは、内部別名解決法と呼ばれます。内部別名解決法を使用する場合は、 XML 文書内で基本キー (ID) の代わりに別名が用いられます。これで別名は、そのエレメントを参照するために、XML ファイル内の他の場所で使用できます。したがって、XML ファイルを構築するのに必要な固有索引を知っている必要はありません。発行中、ID リゾルバーは @ 記号を固有な値に置き換えます。詳細については、 489 ページの『付録 B. データの作成』を参照してください。
- interfacename はコマンド・インターフェース名です。
- classname はコマンド・インプリメンテーション・クラス名です。通常、この名前は、インターフェース名の末尾に Impl を付加したものとなっています。

7. 新しいビューを登録する場合、または新しい JSP ファイルをビューに関連付ける場合は、次の例 (ToolTech サンプル・ストア command.xml ファイルからの例) を参考にして、VIEWREG テーブルにエントリーを作成します。

```
<viewreg
viewname="OrderOptionsView"
devicefmt_id="-1"
```

```
storeent_id="@storeent_id_1"
interfacename="com.ibm.commerce.command.ForwardViewCommand"
classname="com.ibm.commerce.command.HttpForwardViewCommandImpl"
properties="docname=Shipping.jsp"
internal="0"
https="0"/>
```

ここで、

- viewreg は、この情報を読み込むデータベース・テーブル (VIEWREG) の名前です。
- viewname はビューの名前です。
- devicefmt_id は、このビューが使用されるデバイスのタイプ (ブラウザーなど) です。
- storeent_id はストア・エンティティー ID です。@ 記号を使用するのは、内部別名解決法と呼ばれます。内部別名解決法を使用する場合は、 XML 文書内で基本キー (ID) の代わりに別名が用いられます。これで別名は、そのエレメントを参照するために、XML ファイル内の他の場所で使用できます。したがって、XML ファイルを構築するのに必要な固有索引を知っている必要はありません。発行中、ID リゾルバーは @ 記号を固有な値に置き換えます。詳細については、 489 ページの『付録 B. データの作成』を参照してください。
- interfacename はビュー・コマンド・インターフェース名です。デフォルト・オプションは ForwardView、DirectView、および RedirectView です。
- classname はビュー・インプリメンテーション・クラス名です。通常、この名前は、インターフェース名の末尾に Impl を付加したものとなっています。
- properties はコマンドへの入力プロパティーとして設定される、デフォルトの名前と値の対です。同じページが常に表示される場合は、このプロパティーに docname=Shipping.jsp などと JSP ファイル名を設定します。
- internal は、ビューが WebSphere Commerce にとって内部的なものかどうかを示します。内部ビューは、WebSphere Commerce のツールによって使用されます。内部的なものである場合は 1、外部的なものである場合は 0 を使用します。自分で作成するビューは外部的なものでなければなりません。
- https はこの URL 要求に必要なセキュア HTTP です。セキュア HTTP が必要な場合は 1、必要ない場合は 0 を使用します。

@ と **&** の使用法の詳細については、 489 ページの『付録 B. データの作成』を参照してください。

第 16 章 カタログ資産

従来のカタログと同様、オンライン・カタログも、販売する商品やサービスから構成されています。オンライン・カタログのサイズや構造は、購入用の商品取引のタイプや金額によって、ストアごとにかなり異なることもありますが、カタログに必要なものは次のとおりです。

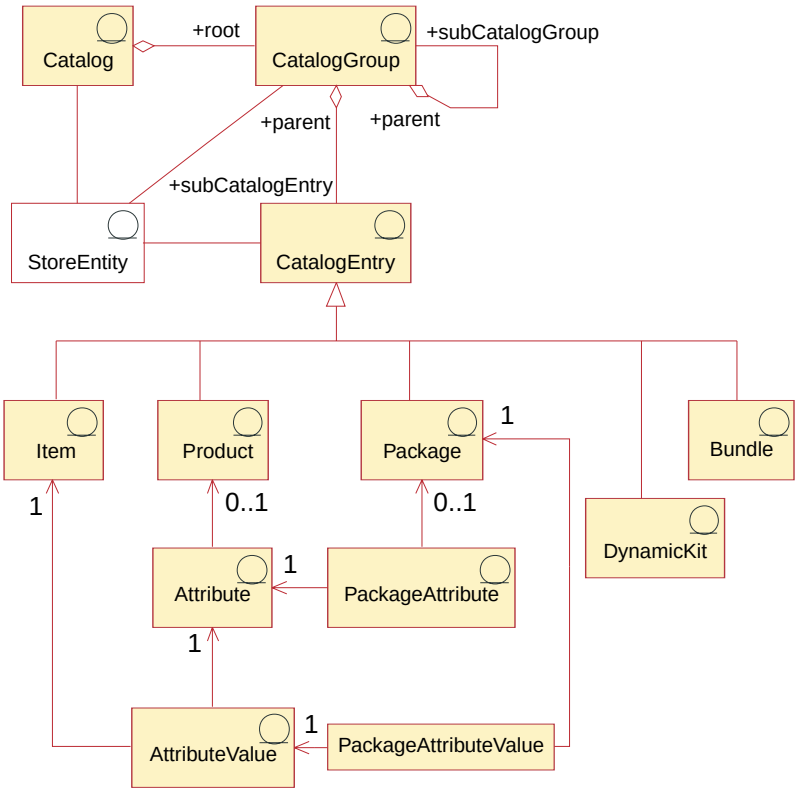
- 販売するもの。次のものが含まれます。
  - 販売価格。必ずと言ってよいほどオンライン・カタログに掲載されます。
  - 商品データ。商品取引の詳細やイメージなど。
  - カテゴリー。大部分のカタログ (ただしすべてではない) が、顧客のナビゲーションの便宜を図って商品取引をカテゴリーに分類しているのと同様です。
- 商品の表示方法。カタログ表示ページは、ページが顧客にどのように表示されるかの概要を決定し、さまざまなカタログ・ページのルック・アンド・フィールを統一の取れたものとします。カタログの構造は、扱う商品取引によって異なります。

WebSphere Commerce のカタログについて

WebSphere Commerce では、ストアのオンライン・カタログにいくつかの要件が課されています。 WebSphere Commerce システムのすべてのストアには、マスター・カタログ がなければなりません。これは単にカタログとも言います。マスター・カタログは、ストアの商品取引を管理する中心です。これは、すべての商品、アイテム、関係、およびストアで販売されるものすべての標準価格を含む 1 つのカタログです。

マスター・カタログは複数のストアで共有することができ、また必要な数のストアを定義できます。カタログ管理用のマスター・カタログを作成することに加えて、表示の目的で 1 つ以上のセールス・カタログ を作成することもできます。セールス・カタログにはサブセットまたはマスター・カタログと同じカタログ・エントリーを含めることができますが、カスタマーに表示する目的で、セールス・カタログはマスター・カタログよりずっと柔軟なカテゴリー構造になっています。マスター・カタログは 1 つしかありませんが、セールス・カタログはいくつでも必要なだけ作成することができます。ただし、オンラインの商品取引を管理するためにマスター・カタログを使用する必要があるので、マスター・カタログをセールス・カタログとして使用して、メインテナンスのオーバーヘッドを最小限に抑えるようお勧めします。

WebSphere Commerce ストア用に新しいマスター・カタログを作成する場合、または ToolTech などの WebSphere Commerce サンプル・ストアで使用可能な既存のマスター・カタログを変更する場合は、自分のカタログがこれらの要件を確実に満たすようにする必要があります。次の図は、WebSphere Commerce におけるマスター・カタログの基本構造の概観を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce 情報モデルの一部です。情報モデルの詳細については、111 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、487 ページの『付録 A. UML の凡例』を参照してください。

カタログ

カタログ は情報モデルの出発点です。カタログには、オンライン・カタログ用のすべての階層情報およびナビゲーション情報が入っています。カタログは、オンライン・ストアで表示され、また購入可能なカタログ・グループとカタログ・エントリーの集合です。

WebSphere Commerce では、カタログは、データベースにおいて、*カタログ・エンティティー* で表されています。カタログ・エンティティーは、固有のカタログ ID とカタログの説明 (カタログ名など) から構成されています。各カタログは個別かつ固有のエンティティーであるため、簡単に 1 つ以上のストアと関連付けることができます。 WebSphere Commerce システムのストアはどれも、最低 1 つのカタログ・エンティティーと関連付けなければなりません。

カタログ・グループ

カタログ・グループ とは、区分化の目的で作成される、カタログ・エントリーの一般的なグループ分けのことです。 1 つのカタログ・グループは 1 つのカタログに所属し、複数のカタログ・グループまたはカタログ・エントリーを含むことができ

ます。カタログ・グループは複数のカタログと関連付けることができます。カタログ・グループはカテゴリー とも呼ばれます。

フラット・カタログは、商品がカテゴリーごとにグループ分けされず、代わりに商品のリストが表示されるカタログです。 WebSphere Commerce ではフラット・カタログを作成することも可能ですが、構造上およびナビゲーション上の理由から、カタログ・グループを作成することをお勧めします。

カタログ・グループを作成するときには、まずカタログを階層、つまり逆さまにした木の形に整理する必要があります。ツリーは、一般カタログ・グループ (ルート・カテゴリーまたは最上位カテゴリーと呼ばれる) から始まり、それ以上分割できなくなるまで、特定のサブカテゴリーへと徐々に分岐していきます。商品しか入っていない最低レベルのカタログ・グループは、リーフと呼ばれます。カタログ・グループは、そのすぐ下のカテゴリーに対しては親、1 つ上のカテゴリーに対しては子になります。例として、メンズ・ファッションは紳士服のカテゴリーのグループであり、パンツやシャツといったカタログ・グループは商品のグループです。

カタログ・エントリー

各カタログ・グループには*カタログ・エントリー* が入っています。カタログ・エントリーとは、オンライン・カタログでオーダー可能な商品のことです。エントリーには通常、部品番号、説明、1 つ以上のオファー価格、イメージ、およびその他詳細情報があります。カタログ・エントリーは、商品、アイテム、バンドル、パッケージ、スタティック・パッケージ、またはダイナミック・パッケージのいずれかになります。必要なら、6 つの既存のモデルのうちいずれにもあてはまらない、新しいカタログ・エントリー・タイプを作成することもできます。カタログ・エントリーの各タイプの詳細については、以下で説明します。

商品

商品 は、カタログ・エントリーの 1 タイプです。商品は、同じ属性を表すアイテム (または SKU) のグループのテンプレートとなります。たとえば、シャツはカタログに記載されている商品です。シャツに属性と属性値を追加すると、それぞれのバリエーションが 1 つのアイテムとなります。たとえば、S サイズの黒のシャツなどです。

アイテム

アイテム は、特定の名前、部品番号、および価格を持つ具体的な商品取引の単位です。たとえば、S サイズの黒のシャツはアイテムですが、シャツは商品です。特定の商品に関連するすべてのアイテムは同じ属性セットを示し、それらの属性値によって区別されます。

注: WebSphere Commerce アクセラレーターのユーザーにとって、アイテムと *SKU* は同義語です。 WebSphere Commerce アクセラレーターの商品管理ツールを使用する場合、オーダー可能なアイテムを SKU と呼びます。 WebSphere Commerce データベース・スキーマでは、この特定のタイプのカタログ・エントリーをアイテムと呼びます。

バンドル

バンドル は、カタログ・エントリーのコレクションで、これにより顧客は一度に複数のアイテムを購入することができます。たとえば、コンピューターのバンドルは

中央演算処理装置、モニター、ハード・ディスク、および CD-ROM ドライブなどで構成されています。バンドルにできるのは、アイテムのグループ、あるいは商品、アイテム、および完全解体パッケージの組み合わせです。アイテムだけを含むバンドルを選択する場合、バンドルは個別にオーダーできる SKU に分解され、これは個別にショッピング・カートに追加されます。ただし、商品を含むバンドルを選択する場合、これらの商品を SKU 解決によってアイテムに解体してからショッピング・カートに追加する必要があります。いずれの場合でも、いったんバンドルが分解されてそのコンポーネント・アイテムがショッピング・カートに追加されると、各アイテムを変更または除去できます。

パッケージ

パッケージ は、カタログ・エントリーの不可分のコレクションです。たとえば、コンピューター・パッケージには、別売りできない特定の中央演算処理装置、モニター、およびハード・ディスクが含まれます。商品と同様、パッケージには定義されている属性があり、パッケージは完全解体パッケージのコンテナーにあたります。完全解体パッケージは、SKU に相当します。パッケージはそれ自身の価格を持った実際にオーダー可能な SKU であり、ショッピング・カートに追加できます。パッケージは、ナビゲーション中もショッピング・カート内に置かれた後も分解、変更することはできません。

注: WebSphere Commerce アクセラレーターのユーザーにとって、パッケージ (package) とパッケージ *(prebuilt kit)* は同義語です。 WebSphere Commerce アクセラレーターの商品管理ツールを使用する場合、パッケージ (package) はパッケージ (prebuilt kit) として知られています。 WebSphere Commerce データベース・スキーマでは、この特定のタイプのカタログ・エントリーをパッケージと呼びます。

ダイナミック・パッケージ

ダイナミック・パッケージ は、顧客が動的に構成できるカタログ・エントリーのタイプです。この商品の構成 (またはグループ分け) は顧客の要件に基づき、1 単位で販売されます。ダイナミック・パッケージのコンポーネントは、一連の定義済み規則とユーザーの対話による外部商品コンフィギュレーターで構成され、オーダー入力時に提供されます。オーダーにダイナミック・パッケージを追加することは、パッケージを追加することに似ています。パッケージと同じように、ダイナミック・パッケージの個々のコンポーネントは変更できず、構成は、全体として行う必要があります。しかし、ダイナミック・パッケージ・コンポーネントは、外部商品コンフィギュレーターを使用して再構成を行うことによって変更することができます。

スタティック・パッケージ

スタティック・パッケージ は、1 つの単位としてオーダーされる商品のグループです。スタティック・パッケージに含まれている商品に関する情報は、WebSphere Commerce 内で事前定義され、制御されます。オーダー内の個々のコンポーネントは変更できず、まとめて実行する必要があります。スタティック・パッケージは、そのコンポーネントのいずれかが利用できない場合にバックオーダーします。

スタティック・パッケージは、まずパッケージとして作成され、その後は管理者により構成されます。

商品セット

商品セット は、発行済みのカタログ・エントリーと関連付けられます。商品セットは、カタログを論理サブセットに区分化するためのメカニズムを提供します。この区分化によって、さまざまなユーザーにカタログの異なる部分を示すことができます。契約を作成して、契約の参加者だけが、事前定義された商品セットに当たる商品を購入できるように指定できます。 WebSphere Commerce は、マスター・カタログに対する契約および権利フィルター規則を作成して管理するツールを提供します。

属性

属性 は、オンライン・ストアの商品のプロパティーです。属性には 2 つのタイプがあります。

- 定義属性は、カラーやサイズなどのプロパティーです。属性値は、特定のカラー (青または黄色) やサイズ (M) などの、属性のプロパティーになります。属性値をアイテムに割り当てる前に、それを定義しておく必要があります。属性値は暗黙的にその属性と関連しています。それぞれの属性と属性値の可能な組み合わせは、新しいアイテムと等しくなります。属性とその値を作成したら、名前、説明、およびタイプ (テキスト、整数、または 10 進数) などの情報を更新できます。定義属性は、SKU 解決のために使用されます。 SKU 解決では、属性と属性値との可能な組み合わせが、それぞれ 1 つのアイテムを定義します。
- 対照的に、説明属性は単に追加説明を提供します。たとえば、ドライ・クリーニングのみ行うべきで洗濯はできない衣類があり、説明属性は、このドライ・クリーニングのみの条件を指定できます。記述属性は SKU 解決には使用されず、商品説明を拡張したり、ビジネス固有の情報を容易にカスタマイズできるようにするためのものであることに注意してください。

属性値

属性値 は、属性のプロパティーです。たとえば、特定のカラー (青または黄) やサイズ (S、M、L) などです。属性値をアイテムに割り当てる前に、それを定義しておく必要があります。定義している属性値の相互間の可能な組み合わせが、それぞれ 1 つのアイテムを定義します。

パッケージの属性

パッケージの属性 は、パッケージに含まれている商品の属性から作成される必要があります。アイテムのみを含むパッケージには、パッケージの属性はありません。

パッケージの属性値

パッケージの属性値 は、パッケージの属性に割り当てられた値です。パッケージの属性値は、パッケージに含まれている商品の属性値から作成される必要があります。

WebSphere Commerce でのカタログ資産の構造に関する詳細については、WebSphere Commerce オンライン・ヘルプの『カタログ・データ・モデル』を参照してください。

WebSphere Commerce でのカタログ資産の作成

ストア用にカタログ資産を作成するには、複数の WebSphere Commerce データベース・テーブルに情報を追加することによって、マスター・カタログを作成する必要があります。カタログは XML ファイルを使用して作成できます。XML ファイルは、ローダー・パッケージを使用してデータベースにロードされます。グローバル化されたカタログを作成する場合、ストアがサポートする各ロケールごとに個別の XML ファイルが必要です。それぞれのロケール固有の XML ファイルでは、カタログ、カタログ・グループ、およびカタログ・エントリーについての翻訳可能な情報 (説明など) が追加されます。

カタログ作成プロセスの概要は、次のとおりです。

1. WebSphere Commerce では、XML ファイルを使用してカタログを作成します。カタログの作成は、まずカタログ・エンティティーから始めます。カタログ・エンティティーとは、一般の (紙の) カタログのデータベース版とも言うべきものです。
2. カタログの構造とナビゲーションを作成します。そのためには、カタログ・グループを追加して、商品取引のカテゴリーとレイアウトを決めます。
3. カタログ・エントリーのベースとして在庫情報を作成します。
4. カタログ・エントリーの形で商品取引を追加します。カタログ・エントリーの種類には、商品、SKU、バンドル、パッケージ、スタティック・パッケージ、およびダイナミック・パッケージがあります。
5. カタログの商品に属性と属性値を追加して、それぞれの SKU を区別します。
6. 販売促進のために、パッケージやバンドルを作成して、特定のカタログ・エントリーをグループ化できます。
7. 次に、カタログ・グループとカタログ・エントリーの間の関係を作成します。この関係によって、どのエントリーがどのカタログ・グループに属するかを決めます。
8. 商品をお勧めするための戦略として、カタログ・エントリーについて、取引管理上の関連付け (アソシエーション) を作成できます。
9. カタログ、カタログ・グループ、カタログ・エントリーを WebSphere Commerce ストアに関連付けます。
10. 最後の段階で次の項目を作成します。
    a. 商品取引にかかる税。
    b. 配送方法。
    c. 在庫の倉庫や発送/受取センターとしての役割を果たす配送センター。ストアには複数の配送センターを定義できます。
    d. 商品取引の価格。

マスター・カタログの作成

複数のカテゴリー・レベルを含むマスター・カタログを作成する場合は、次のタスクを完了します。

パート 1: カタログ作成の準備

1. カタログ情報と、それに対応する WebSphere Commerce のオブジェクト・モデルとデータ・モデルを確認します。カタログ情報は WebSphere Commerce Server のコンポーネントの 1 つであり、オーダー可能な商品取引のために、オンライン・カタログのナビゲーション、区分化、カテゴリー化、および関連を提供します。
2. WebSphere Commerce ローダー・パッケージ情報を確認します。ローダー・パッケージは、主に WebSphere Commerce データベースを準備して、そこにデータをロードするユーティリティーで構成されます。ローダー・パッケージを使用して、データベースに大量のデータをロードしたり、データを更新したりすることができます。ローダー・パッケージの詳細については、 363 ページの『第 10 部 ストアの発行』を参照してください。
3. 489 ページの『付録 B. データの作成』に記載されている情報を確認します。
4. カタログ所有者とするため、管理コンソールで組織を作成します。詳細については、WebSphere Commerce オンライン・ヘルプのトピック『組織の作成』を参照してください。
5. 既存の XML エントリー、および ToolTech サンプル・ストアの `catalog.xml` ファイルを参考にして、マスター・カタログ用の新しい XML ファイルを作成します。グローバル化されたカタログを作成する場合、ストアがサポートする各ロケールごとに個別の `catalog.xml` ファイルを作成します。説明情報はすべてロケール固有のファイルに指定されるので、これを翻訳するのは容易です。この例では、翻訳の不要なすべての情報のために 1 つの catalog.xml ファイルが使用され、さらにストアがサポートするロケールごとに第 2 の catalog.xml が使用されて翻訳の必要な情報が入れられます。または、ToolTech サンプル・ストアから既存の XML ファイルを使用して、必要に応じて情報を変更することもできます。 ToolTech サンプル・ストアの catalog.xml ファイルがストア・アーカイブ・ファイルにあります。 catalog.xml ファイルを表示するには、ZIP プログラムを使用して、ストア・アーカイブを解凍します。 catalog.xml ファイルは以下のデータ・ディレクトリーにあります。

   - *WC_installdir*/samplestores

   注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

   `catalog.dtd` ファイルは以下のディレクトリーにあります。

   - *WC_installdir*/xml/sar

パート 2: カタログ・エンティティーの作成

1. ToolTech サンプル・ストアの次の例を参考にして、CATALOG および CATALOGDSC テーブルに情報を追加することによって、カタログ・エンティティーを作成します。カタログ・エンティティーはデータベースの中のカタログを表します。

```
<catalog
catalog_id="@catalog_id_1"
member_id="@seller_b2b_mbr_id"
```

```
identifier="ToolTech"
description="ToolTech Catalog"
tpclevel="0"
/>
```

ここで、

- catalog_id は、内部参照番号です。
- member_id は、カタログの所有者を識別する内部参照番号です。
- identifier は、カタログの外部名です。
- description は、カタログの説明です。

2. ToolTech サンプル・ストアの次の例を参考にして、後で翻訳することを意図したロケール固有の XML ファイルにカタログの説明を追加します。

```
<catalogdsc
catalog_id="@catalog_id_1"
language_id="&en_US;"
name="Store master catalog"
/>
```

ここで、

- catalog_id は、この言語特定情報をカタログに関連付ける内部参照番号です。
- language_id は、言語の ID です。
- name は、言語に依存するカタログの名前です。

パート 3: カタログ・グループの作成

1. ToolTech サンプル・ストアの次の例を参考にして、CATGROUP および CATGRPDESC テーブルに情報を追加することによって、カタログ・グループを作成します。カタログ・グループ (カテゴリーとも呼ばれる) は、他のカタログ・グループまたは製品をグループにしたものです。カタログの各カタログ・グループごとにこのタスクを完了します。

```
<catgroup
catgroup_id="@catgroup_id_1"
member_id="@seller_b2b_mbr_id"
identifier="Woodworking"
markfordelete="0"
/>
```

ここで、

- catgroup_id は、カタログ・グループの内部参照番号です。
- member_id は、カタログの所有者を識別する内部参照番号です。
- identifer は、カタログの外部名です。
- markfordelete は、カタログ・グループが削除の対象としてマークされているかどうかを示します。
  - 0 = マークされていない。
  - 1 = マークされている。

2. ToolTech サンプル・ストアの次の例を参考にして、後で翻訳することを意図したロケール固有の XML ファイルにカタログ・グループの説明を追加します。カタログの各カタログ・グループごとにこのタスクを完了します。

```
<catgrpdesc
language_id="&en_US;"
catgroup_id="@catgroup_id_1"
name="Woodworking"
shortdescription="Woodworking"
longdescription="Woodworking"
published="1"
/>
```

ここで、

- language_id は、言語の ID です。
- catgroup_id は、カタログ・グループの内部参照番号です。
- name は、言語に依存するカタログの名前です。
- shortdescription は、カタログ・グループの要旨です。
- longdescription は、カタログ・グループの詳細記述です。
- published は、このカタログ・グループを、 language_id によって示された言語で表示するかどうかを指示します。
  - 0 = 表示しない。
  - 1 = 表示する。

注: カタログ・グループとその説明を作成するたびに、新しいカタログ・グループを表すよう、catgroup_id が変わります。たとえば、catgroup_id="@catgroup_id_2"、catgroup_id="@catgroup_id_3"、および catgroup_id="@catgroup_id_4"、などです。

3. カタログ・グループを作成したら、CATTOGRP テーブルに情報を追加することによって、最上位のカタログ・グループをカタログに割り当てます。このカタログ・グループは、そのすぐ下のカタログ・グループの親になります。カタログの中の各最上位カタログ・グループごとにこのタスクを完了します。 ToolTech サンプル・ストアの次の例を参考にしてください。

```
<cattogrp
catalog_id="@catalog_id_1"
catgroup_id="@catgroup_id_1"
/>
```

ここで、

- catalog_id は、カタログの参照番号です。
- catgroup_id は、カタログ・グループの参照番号です。

注: 最上位のカタログ・グループをカタログに割り当てるたびに、新しいカタログ・グループの関連を表すよう、catgroup_id が変更されます。たとえば、catgroup_id="@catgroup_id_2"、catgroup_id="@catgroup_id_3"、および catgroup_id="@catgroup_id_4"、などです。

4. カタログ・グループの親と子の構造が決定されたら、CATGRPREL テーブルに情報を追加することによって、カタログ・グループ間の関係を作成します。カタログの中の、親子のカタログ・グループ構造のそれぞれについて、このタスクを完了します。 ToolTech サンプル・ストアの次の例を参考にしてください。

```
<catgrprel
catgroup_id_parent="@catgroup_id_1"
catgroup_id_child="@catgroup_id_11"
catalog_id="@catalog_id_1"
sequence="0"
/>
```

ここで、

- catgroup_id_parent は、この関係のソース・カタログ・グループです。
- catgroup_id_child は、この関係のターゲット・カタログ・グループです。
- catalog_id は、カタログの参照番号です。
- sequence は、カタログ・グループの内容の表示順序を決定する番号です。

注: 各カタログ・グループの関係ごとに、新しい関係を表すよう、catgroup_id_child と sequence が変更されます。たとえば、後に続く関係は、 catgroup_id_child="@catgroup_id_12" と sequence="1"、catgroup_id_child="@catgroup_id_13" と sequence="2" (以下同様) などと表示されるかもしれません。カタログの中で誘導型構造を使用していない場合は、CATGRPREL 関係を除去できます。

パート 4: 在庫情報の作成

1. ToolTech サンプル・ストアの次の例を参考にして、BASEITEM、BASEITEMDSC、ITEMSPC、ITEMVERSN、VERSIONSPC、DISTARRANG および STOREITEM テーブルに情報を追加することによって、在庫情報を作成します。最初に、BASEITEM テーブルに情報を追加することにより、基本アイテムを作成します。基本アイテムとは、名前と説明を共有する、商品の一般的なファミリーを表します。カタログの中の在庫アイテムのグループごとにこのタスクを完了します。

```
<baseitem
baseitem_id="@baseitem_id_102"
member_id="@seller_b2b_mbr_id"
markfordelete="0"
partnumber="tooltech_sku_102"
itemtype_id="ITEM"
quantitymeasure="C62"
quantitymultiple="1.0"
/>
```

ここで、

- baseitem_id は、生成される固有キーです。
- member_id は、基本アイテムの所有者です。
- markfordelete は、基本アイテムが削除の対象としてマークされているかどうかを示します。
  - 0 = マークされていない。
  - 1 = マークされている。
- partnumber は、基本アイテムを所有者ごとに固有に識別します。
- itemtype_id は、基本アイテムのタイプです。
  - ITEM = アイテム、パッケージ、またはバンドル
  - DNKT = ダイナミック・キット
  - STKT = スタティック・パッケージ。
- quantitymeasure は、数量の倍数の計測単位です。
- quantitymultiple は、整数単位で計測される、基本アイテムの量です。quantitymeasure と共に、各整数単位がどれだけの量を表すかを示します。

注: カタログの中に作成するあらゆる商品に対して、基本アイテムを作成する必要があります。基本アイテムを作成するたびに、baseitem_id と partnumber の番号が変わって、新しい基本アイテムが作成されます。たとえば、ある新しい基本アイテムのエントリーは baseitem_id="@baseitem_id_147" と partnumber="tooltech_sku_147" になり、次の基本アイテムのエントリーは baseitem_id="@baseitem_id_192" と partnumber="tooltech_sku_192" になるかもしれません。

2. ToolTech サンプル・ストアの次の例を参考にして、指定されたアイテムについての情報をデータベースに追加します。指定されたアイテムはすべての属性に値が指定されているアイテムであり、カタログ内のアイテム、パッケージ、バンドル、またはダイナミック・パッケージを表します。カタログの中の指定されたアイテムごとに、このタスクを完了します。

```
<itemspc
itemspc_id="@itemspc_id_106"
baseitem_id="@baseitem_id_102"
markfordelete="0"
partnumber="T0000106"
member_id="@seller_b2b_mbr_id"
discontinued="N"
/>
```

ここで、

- itemspc_id は、生成される固有キーです。
- baseitem_id は、商品基本アイテムです。
- markfordelete は、指定されたアイテムが削除の対象としてマークされているかどうかを示します。
  - 0 = マークされていない。
  - 1 = マークされている。
- partnumber は、指定されたアイテムを所有者ごとに固有に識別します。
- member_id は、指定されたアイテムの所有者です。
- discontinued は、指定されたアイテムが製造中止になったかどうかを示します。
  - Y = 製造中止されました。十分な在庫がある場合はオーダー可能ですが、バックオーダーはできません。
  - N = アクティブです。在庫切れの場合はバックオーダーできます。

注: カタログの中に作成するアイテムごとに、指定されたアイテムを作成する必要があります。指定されたアイテムを定義するたびに、itemspc_id="@itemspc_id_107"、baseitem_id="@baseitem_id_102"、partnumber="T0000107" の各番号が変わり、指定された新しいアイテムが作成されます。たとえば、指定された新しいアイテムのエントリーは itemspc_id="@itemspc_id_108"、baseitem_id="@baseitem_id_102"、および partnumber="T0000108" になり、指定された次のアイテムのエントリーは itemspc_id、baseitem_id、および partnumber などとなるかもしれません (以下同様)。

3. ToolTech サンプル・ストアの次の例を参考にして、あるアイテム・バージョンと基本アイテムとの関係に関する次の情報をデータベースに追加します。カタログの中のこのような関係ごとに、このタスクを完了します。

```
<itemversn
itemversn_id="@itemversn_id_102"
baseitem_id="@baseitem_id_102"
expirationdate="2010-01-01 00:00:00.000000"
versionname="version"
/>
```

ここで、

- itemversn_id は、アイテム・バージョンを識別するために生成された参照番号です。
- baseitem_id は、基本アイテムです。
- expirationdate は、アイテム・バージョンの有効期限です。
- versionname は、基本アイテムのアイテム・バージョンを固有に識別します。

注: アイテム・バージョンと基本アイテムとの間の関係を作成するたびに、itemversn_id と baseitem の番号が変わって、新しい関係が作成されます。 baseitem_id は、既存の基本アイテムに一致します。たとえば、新しい関係のエントリーは itemversn_id="@itemversn_id_107" と baseitem_id="@baseitem_id_107" になり、次の関係のエントリーは itemversn_id="@itemversn_id_108" と baseitem_id="@baseitem_id_108" などとなるかもしれません (以下同様)。

4. ToolTech サンプル・ストアの次の例を参考にして、商品バージョンと指定されたアイテムの関係に関する次の情報をデータベースに追加します。カタログの中のこのような関係ごとに、このタスクを完了します。

```
<versionspc
versionspc_id="@versionspc_id_106"
itemspc_id="@itemspc_id_106"
itemversn_id="@itemversn_id_102"
/>
```

ここで、

- versionspc_id は、生成される固有 ID です。
- itemspc_id は、カタログ・エントリーと関連のある指定されたアイテムです。
- itemversn_id は、アイテムのバージョンを示します。

注: 商品バージョンと指定済みアイテムの関係を作成するたびに、versionspc_id と itemspc_id の番号が変わって、新しい基本アイテムが作成されます。 itemspc_id は、指定された既存のアイテムに一致します。たとえば、新しい関係のエントリーは versionspc_id="@versionspc_id_107" と itemspc_id="@itemspc_id_107" になり、次の関係のエントリーは versionspc_id="@versionspc_id_108" と itemspc_id="@itemspc_id_108" などとなるかもしれません (以下同様)。

5. ToolTech サンプル・ストアの次の例を参考にして、配送手配をデータベースに追加します。分散配置によって、ストアは独自の在庫を販売できるようになります。カタログの中のこのような配送手配ごとに、このタスクを完了します。

```
<distarrang
distarrang_id="@distarrang_id_102"
wholesalestore_id="@storeent_id_1"
merchantstore_id="@storeent_id_1"
baseitem_id="@baseitem_id_102"
```

```
pickingmethod="F"
startdate="2000-12-25 00:00:00.000000"
enddate="2010-01-01 00:00:00.000000"
/>
```

ここで、

- distarrang_id は、配送手配の参照番号です。
- wholesalestore_id は、マーチャント・ストアで販売可能な在庫を所有する卸売ストアです。この卸売ストアは merchantstore_id と同じでなければなりません。
- merchantstore_id は、卸売ストアの在庫から販売できるマーチャント・ストアです。このマーチャント・ストアは wholesalestore_id と同じでなければなりません。
- baseitem_id は、配送手配の対象となる商品です。
- pickingmethod は、この配置により RECEIPT テーブルから在庫がピッキングされる順序を決定します。
  - F = FIFO (先入れ先出し) : 古いものから順に在庫から取り出します。
  - L = LIFO (後入れ先出し) : 新しいものから順に在庫から取り出します。
- startdate は、この配送手配の開始が有効になる時刻です。
- enddate は、この配送手配の停止が有効になる時刻です。

注: 配送手配を作成するたびに、 distarrang_id と baseitem の番号が変わって、新しい配送手配が作成されます。たとえば、第 2 の配送手配には distarrang_id="@distarrang_id_147" と baseitem_id="@baseitem_id_147" が含まれ、第 3 の配送手配には distarrang_id="@distarrang_id_192" と baseitem_id="@baseitem_id_192" が含まれる、というようになります。

6. ToolTech サンプル・ストアの次の例を参考にして、特定のストアが特定の基本アイテムのうち指定されたアイテムの在庫をデータベースに割り振る方法に影響する属性を追加します。カタログの中のこのような基本アイテムごとに、このタスクを完了します。

```
<storeitem
baseitem_id="@baseitem_id_102"
storeent_id="@storeent_id_1"
trackinventory="Y"
forcebackorder="N"
releaseseparately="N"
returnnotdesired="N"
backorderable="Y"
creditable="Y"
minqtyforsplit="0"
/>
```

ここで、

- baseitem_id は、基本アイテムです。
- storeent_id は、ストアまたはストア・グループです。
- trackinventory は、 RECEIPT テーブルの中で在庫をトラッキングするかどうかを指定します。
  - N = 在庫をトラッキングせず、 RECEIPT テーブルにはエントリーがありません。
  - Y = RECEIPT テーブルの中で在庫をトラッキングします。

- forcebackorder は、基本アイテムに対して指定されたアイテムの割り振りを一時的に中断します。
  - N = 在庫を割り振ることができます (通常の動作)。
  - Y = 十分な在庫があっても在庫の割り振りができません。
- releaseseparately は、基本アイテムに対して指定されたオーダー・アイテムをリリースする方法を指定します。
  - N = オーダー・アイテムを他のオーダー・アイテムと共にリリースします。
  - Y = オーダー・アイテムは別個にリリースする必要があります (専用の梱包で)。
- returnnotdesired は、顧客が返品したい場合、あるいは返品可能な場合であっても、アイテムの返品が望ましくないということを指定します (腐りやすい食品などの場合)。
  - N = アイテム返品に関する顧客の要望に基づいて、クレジット評価を要求しますが、返品は予期されません。
  - Y = 返品が予期されているかのようなクレジット評価を要求します。
- backorderable は、基本アイテムに対して指定されたアイテムのバックオーダーができないことを指定します。
  - N = アイテムのバックオーダーはできません。
  - Y = アイテムのバックオーダーが可能です。
- creditable は、このアイテムに関してマーチャントがオーバーライドなしでクレジットを発行するかどうかを指定します。
  - N = 現金販売。
  - Y = クレジット可能。
- minqtyforsplit は、新しいオーダー・アイテムの残りの未割り振り数量が、最小数量として指定された数量より少ない場合、在庫の割り振りにおいてオーダー・アイテムを自動的には分割しないことを指定します。

注: ストア・アイテムの在庫割り振り規則を定義するたびに、 baseitem_id 番号が、新しい基本アイテムを表す番号に変わります。たとえば、新しい割り振りには baseitem_id="@baseitem_id_147" が含まれていて、第 3 のものには baseitem_id="@baseitem_id_192" が含まれている、などです。

7. ToolTech サンプル・ストアの次の例を参考にして、後で翻訳することを意図したロケール固有の XML ファイルに基本アイテムの説明を追加します。カタログの中のこのような基本アイテムの説明ごとに、このタスクを完了します。

```
<baseitmdsc
baseitem_id="@baseitem_id_102"
language_id="&en_US;"
shortdescription="Circular Saw"
longdescription="Light on weight but not in quality. The Circular Saw
weighs a maximum of 10.9lbs., with a choice of a 12 or 14 amp motor,
and speeds of up to 600 rpms! Low friction 220V aluminum alloy shoe
will ensure the job gets done on time."
/>
```

ここで、

- baseitem_id は、生成される固有キーです。
- language_id は、この情報の言語です。

- shortdescription は、基本アイテムの要旨です。
- longdescription は、基本アイテムの詳細記述です。

パート 5: カタログ・エントリーの作成

1. ToolTech サンプル・ストアの次の例を参考にして、CATENTRY および CATENTDESC テーブルに情報を追加することにより、カタログ・エントリーを作成します。各タイプのカタログ・エントリー (商品、アイテム、パッケージ、バンドル、およびダイナミック・パッケージ) は、カタログの中で販売対象となる、オーダー可能な商品取引を表します。各商品カタログ・エントリーごとに 1 つの基本アイテムを定義する必要があります。カタログの中の各商品カタログ・エントリーごとにこのタスクを完了します。

```
<catentry
catentry_id="@product_id_102"
baseitem_id="@baseitem_id_102"
member_id="@seller_b2b_mbr_id"
catenttype_id="ProductBean"
partnumber="T0000102"
mfpartnumber="Sprain-Tools-102"
mfname="Sprain Tools"
markfordelete="0"
buyable="1"
/>
```

ここで、

- catentry_id は、商品カタログ・エントリーの内部参照番号です。
- baseitem_id は、カタログ・エントリーと関連のある基本アイテムです。
- member_id は、カタログ・エントリーを識別する参照番号です。
- catenttype_id は、カタログ・エントリーのタイプを識別します。
  - ItemBean = アイテムを識別する。
  - ProductBean = 商品を識別する。
  - PackageBean = パッケージを識別する。
  - BundleBean = バンドルを識別する。
  - DynamicKitBean = ダイナミック・リンクを識別する。
- partnumber は、カタログ・エントリーの部品番号を識別する参照番号です。
- mfpartnumber は、カタログ・エントリーを識別するためにメーカーが使用する部品番号です。
- mfname は、カタログ・エントリーのメーカーの名前です。
- markfordelete は、カタログ・エントリーが削除の対象としてマークされているかどうかを示します。
  - 0 = マークされていない。
  - 1 = マークされている。
- buyable は、カタログ・エントリーを個別に購入できるかどうかを示します。
  - 0 = 購入できない。
  - 1 = 購入できる。

注: 基本アイテムをカタログ・エントリーに追加するたびに、新しいカタログ・エントリーを表すよう、catentry_id と baseitem_id のシーケンスが変わります。 catenttype_id は、カタログ・エントリーのタイプに応じて変わります。

- ToolTech サンプル・ストアの次の例を参考にして、カタログ・エントリーごとに指定されたアイテムを定義します。カタログの中の各カタログ・エントリーごとにこのタスクを完了します。

```
<catentry
catentry_id="@catentry_id_106"
itemspc_id="@itemspc_id_106"
member_id="@seller_b2b_mbr_id"
catenttype_id="ItemBean"
partnumber="T0000106"
mfpartnumber="Sprain-Tools-106"
mfname="Sprain Tools"
markfordelete="0"
buyable="1"
/>
```

  ここで、

  – catentry_id は、カタログ・エントリーの内部参照番号です。
  – itemspc_id は、カタログ・エントリーが属する指定されたアイテムです。
  – member_id は、カタログ・エントリーを識別する参照番号です。
  – cattentype_id は、カタログ・エントリーのタイプを識別します。
    - ItemBean = アイテムを識別する。
    - ProductBean = 商品を識別する。
    - PackageBean = パッケージを識別する。
    - BundleBean = バンドルを識別する。
    - DynamicKitBean = ダイナミック・リンクを識別する。
  – partnumber は、カタログ・エントリーの部品番号を識別する参照番号です。
  – mfpartnumber は、カタログ・エントリーを識別するためにメーカーが使用する部品番号です。
  – mfname は、カタログ・エントリーのメーカーの名前です。
  – markfordelete は、カタログ・エントリーが削除の対象としてマークされているかどうかを示します。
    - 0 = マークされていない。
    - 1 = マークされている。
  – buyable は、カタログ・エントリーを個別に購入できるかどうかを示します。
    - 0 = 購入できない。
    - 1 = 購入できる。

  注: 指定されたアイテムをカタログ・エントリーに追加するたびに、catentry_id と itemspc_id のシーケンスが新しいカタログ・エントリーを表すように変わります。 catenttype_id は、カタログ・エントリーのタイプに応じて変わります。マスター・カタログの構造に由来する制限のため、 1 つのカタログ・エントリーが複数のカテゴリーに属することは

不可能です。カタログ・エントリーを複数のカテゴリーに入れるには、セールス・カタログを使用する必要があります。

- ToolTech サンプル・ストアの次の例を参考にして、ロケール固有の XML ファイルに説明を追加します。カタログの中の各カタログ・エントリーの説明ごとに、このタスクを完了します。

```
<catentdesc
catentry_id="@product_id_102"
language_id="&en_US"
name="Circular"
shortdescription="Circular Saw"
longdescription="Light on weight but not in quality. The Circular Saw
weighs a maximum of 10.9lbs., with a choice of a 12 or 14 amp motor,
and speeds of up to 600 rpms! Low friction 220V aluminum alloy shoe
will ensure the job gets done on time."
thumbnail="images/circular_saw_sm.gif"
fullimage="images/circular_saw.gif"
available="1"
published="1"
/>
```

  ここで、

  – catentry_id は、この言語特定情報が関連するカタログ・エントリーを示す内部参照番号です。
  – language_id は、言語の ID です。
  – name は、言語に依存するカタログ・エントリーの名前です。
  – shortdescription は、カタログ・エントリーの要旨です。
  – longdescription は、カタログ・エントリーの詳細記述です。
  – thumbnail は、サムネール・イメージのパスです。
  – fullimage は、完全なイメージのパスです。
  – available は、カタログ・エントリーの可用性に対する時間の長さを指示します。
  – published は、このカタログ・エントリーを、language_id によって示された言語で表示するかどうかを指示します。
    - 0 = 表示する。
    - 1 = 表示しない。

パート 6: 属性と属性値の作成

1. ToolTech サンプル・ストアの次の例を参考にして、後で翻訳することを意図したロケール固有の XML ファイルの ATTRIBUTE および ATTRVALUE テーブルに情報を追加することにより、商品の属性と属性値を作成します。カタログの中の各商品ごとに属性の特定のセットがあります。たとえば、シャツやパンツのサイズおよびカラーなどです。アイテムは、属性値によって定義されます。たとえば、シャツは商品ですが、M サイズの黒のシャツはアイテムです。カタログの中の属性ごとにこのタスクを完了します。

```
<attribute
attribute_id="@attribute_id_103"
language_id="&en_US"
attrtype_id="STRING"
name="Amps"
sequence="0"
```

```
description="Amps"
catentry_id="@product_id_102"
description2="Amps"
/>
```

ここで、

- attribute_id は、属性の内部参照番号です。
- language_id は、この属性値が関係する言語です。
- attrtype_id は、対応する属性値のタイプです。
- name は、属性の名前です。
- sequence は、指定の商品の属性の表示順序を決定する順序番号です。
- description は、属性の説明です。
- catentry_id は、この属性が属する商品の参照番号です。
- description2 は、属性の追加説明です。

注: catentry_id によって定義される商品に属性を追加するたびに、新しい属性を表すよう、attribute_id のシーケンスが変わります。

2. ToolTech サンプル・ストアの次の例を参考にして、属性値を追加します。カタログの中の属性値ごとにこのタスクを完了します。

```
<attrvalue
attrvalue_id="@attrvalue_id_114"
language_id="&en_US"
attribute_id="@attribute_id_103"
name="12.0amps"
attrtype_id="STRING"
stringvalue="12.0amps"
sequence="0"
usage="1"
catentry_id="@catentry_id_106"
/>
```

ここで、

- attrvalue_id は、属性値の内部参照番号です。
- language_id は、この属性値が関係する言語です。
- attribute_id は、値と関連した属性の内部参照番号です。
- name は、属性値の名前です。
- attrtype_id は、属性値のタイプです。
- stringvalue は、属性値です。
- sequence は、指定の属性の属性値の表示順序を決定する順序番号です。
- usage は、以下に示す属性のタイプです。
  - 1 は、SKU 解決に使用される定義属性を示します。
  - 0 (または別の値) は、説明属性を示します。
- catentry_id は、この属性値が記述するアイテム ID です。

注: 属性値を属性に追加するたびに、異なる値を表すよう、attrvalue_id のシーケンスが変わります。 attribute_id シーケンスも変わって、異なる属性を表します。新しい属性値が追加されるたびに、sequence は大きくなります。たとえば、後に続く属性値は sequence="1"、sequence="2"、および sequence="3" (以下同様) などになるかもしれません。

パート 7: 商品とアイテムの間の関係の作成

1. カタログに商品とアイテムを作成したら、CATENTREL テーブルに情報を追加することによって、商品とアイテムの間の関係を定義します。 ToolTech サンプル・ストアの次の例を参考にしてください。カタログの中の商品とアイテムの関係ごとにこのタスクを完了します。

```
<catentrel
catentry_id_parent="@product_id_147"
catreltype_id="PRODUCT_ITEM"
catentry_id_child="@catentry_id_152"
sequence="2"
quantity="1"
/>
```

ここで、

- catentry_id_parent は、この関係のソース・カタログ・エントリー (商品) の参照番号です。
- catreltype_id は、関係のタイプ (PRODUCT_ITEM) です。
- catentry_id_child は、この関係のターゲット・カタログ・エントリー (アイテム) の参照番号です。
- sequence は、表示順序を判別するために使用されるシーケンス番号です。
- quantity は、関係と関連している可能性がある数量です。

注: 商品とアイテムの間の関係を追加するたびに、 catentry_id_parent と catentry_id_child の番号が変わり、 catreltype_id に基づいてさまざまな関係が作成されます。新しい関係はそれぞれ、sequence 番号が異なります。たとえば、sequence="2" がある場合、次の関係の番号は sequence="3" になり、さらに sequence="4" (以下同様) と続きます。

パート 8: パッケージとバンドルの作成

1. 商品とアイテムを作成したら、CATENTRY、CATENTDESC、および CATENTREL の各テーブルに情報を追加することによって、パッケージとバンドルを作成します。たとえば、以下のコード例を使用して、CATENTRY テーブルに情報を追加することによりパッケージまたはバンドルを作成します。カタログの中の各パッケージおよびバンドルごとに、このタスクを完了します。

```
<catentry
catentry_id="@package_id_102"
member_id="@seller_b2b_mbr_id"
catenttype_id="PackageBean"
partnumber="sku-@package_id_102"
mfpartnumber="sku-@package_id_102"
mfname="ToolTech"
markfordelete="0"
buyable="1"
/>
```

ここで、

- catentry_id は、カタログ・エントリーの参照番号です。
- member_id は、カタログ・エントリーの所有者を識別する参照番号です。
- catenttype_id は、カタログ・エントリーのタイプを識別します。
  - PackageBean = パッケージを識別する。
  - BundleBean = バンドルを識別する。
- partnumber は、カタログ・エントリーの部品番号を識別する参照番号です。

- mfpartnumber は、カタログ・エントリーを識別するためにメーカーが使用する部品番号です。
- mfname は、カタログ・エントリーのメーカーの名前です。
- markfordelete は、カタログ・エントリーが削除の対象としてマークされているかどうかを示します。
  - 0 = マークされていない。
  - 1 = マークされている。
- buyable は、カタログ・エントリーが個別に購入できるかどうかを示します。
  - 0 = 購入できない。
  - 1 = 購入できる。

  注: パッケージまたはバンドルを作成するたびに、 catentry_id、partnumber、および mfpartnumber の番号が、異なるパッケージまたはバンドルを作成するように変わります。たとえば、新しいパッケージを作成する場合、エントリーをパッケージとして識別するため、catentry_id="@package_id_103"、 partnumber="sku-@package_id_103"、および mfpartnumber="sku-@package_id_103" (catenttype_id="PackageBean" を含む) を使用できます。新しいバンドルを作成するためには、エントリーをバンドルとして識別するために、catentry_id="@package_id_110"、 partnumber="sku-@package_id_110"、および mfpartnumber="sku-@package_id_110" (catenttype_id="BundleBean" を含む) を使用する、というようになります。
- たとえば、以下のコード例を使用して、後で翻訳することを意図したロケール固有の XML ファイルの CATENTDESC テーブルに情報を追加することにより、パッケージまたはバンドルの説明を追加します。カタログの中の各パッケージおよびバンドルの説明ごとに、このタスクを完了します。

```
<catentdesc
catentry_id="@catentry_id_102"
language_id="-1"
name="computer"
shortdescription="Computer"
longdescription="A combination of a central processing unit, monitor,
 hard drive, and color printer. An ideal starter system."
thumbnail="images/package_system_sm.gif"
fullimage="images/package_system.gif"
available="1"
published="1"
/>
```

  ここで、
  - catentry_id は、この言語特定情報が関連するカタログ・エントリーを示す内部参照番号です。
  - language_id は、言語の ID です。
  - name は、言語に依存するカタログ・エントリーの名前です。
  - shortdescription は、カタログ・エントリーの要旨です。
  - longdescription は、カタログ・エントリーの詳細記述です。
  - thumbnail は、カタログ・エントリーのサムネール・イメージのパスです。

– fullimage は、カタログ・エントリーの完全なイメージのパスです。
– available は、カタログ・エントリーの可用性に対する時間の長さを指示します。
– published は、カタログ・エントリーを、language_id によって示された言語で表示するかどうかを指示します。
  - 0 = カタログ・エントリーを表示しない。
  - 1 = カタログ・エントリーを表示する。

• たとえば、以下のコード例を使用して、CATENTREL テーブルに情報を追加し、パッケージまたはバンドルとそのコンポーネントとの間の関係を作成します。カタログの中のパッケージまたはバンドルのコンポーネントの関係ごとにこのタスクを完了します。

```
<catentrel
catentry_id_parent="@catentry_id_102"
catreltype_id="PACKAGE_COMPONENT"
catentry_id_child="@catentry_id_97"
sequence="1.0"
quantity="1.0"
/>
```

ここで、

– catentry_id_parent は、この関係のソース・カタログ・エントリー (パッケージまたはバンドル) の参照番号です。
– catreltype_id は、この関係のタイプです。
  - PACKAGE_COMPONENT は、パッケージとそのコンポーネントの間の関係を表します。
  - BUNDLE_COMPONENT は、バンドルとそのコンポーネントの間の関係を表します。
– catentry_id_child は、この関係のターゲット・カタログ・エントリー (コンポーネント) の参照番号です。
– sequence は、表示順序を判別するために使用されるシーケンス番号です。
– quantity は、関係と関連している可能性がある数量です。

注: パッケージとバンドル間の関係を作成するたびに、既存のカタログ・エントリーに合わせて catentry_id_parent と catentry_id_child の番号が変わります。新しい関係はそれぞれ、sequence 番号が異なります。たとえば、sequence="1.0" から始まる場合、次の関係は sequence="2.0" になり、さらに sequence="3.0" (以下同様) と続きます。

パート 9: カタログ・グループとカタログ・エントリーの間の関係の作成

1. カタログの中にカタログ・グループとカタログ・エントリーを作成したら、CATGPENREL テーブルに情報を追加することによって、カタログ・グループとカタログ・エントリーとの間の関係を定義します。マスター・カタログの構造に由来する制限のため、 1 つのカタログ・エントリーが複数のカテゴリーに属することは不可能です。カタログ・エントリーを複数のカテゴリーに入れるには、セールス・カタログを使用する必要があります。 ToolTech サンプル・ストアの次の例を参考にしてください。カタログの中のカタログ・グループとカタログ・エントリーとの関係ごとにこのタスクを完了します。

```
<catgpenrel
catgroup_id="@catgroup_id_11"
catalog_id="@catalog_id_1"
catentry_id="@product_id_102"
sequence="0"
/>
```

ここで、

- catgroup_id は、この関係のソース・カタログ・グループです。
- catalog_id は、この関係が含まれているカタログです。
- catentry_id は、この関係のターゲット・カタログ・グループです。
- sequence は、カタログ・グループの内容の表示順序を決定するシーケンス番号です。

注: カタログ・グループとカタログ・エントリーと間の関係を作成するたびに、catgroup_id と catentry_id の番号が変わり、さまざまなカタログ・グループとカタログ・エントリーの新しい関係が形成されます。新しい関係はそれぞれ、sequence 番号が異なります。たとえば、sequence="0" から始まる場合、次の関係は sequence="1" になり、さらに sequence="2" (以下同様) と続きます。

パート 10: 取引管理アソシエーションの作成

1. たとえば、以下のコード例を使用して、MASSOCECE テーブルに情報を追加し、カタログ・エントリー同士の取引管理アソシエーションを作成します。カタログの中の取引管理アソシエーションごとにこのタスクを完了します。

```
<massoccece
massoccece_id="@relationship_id_100"
massoctype_id="X-SELL"
catentry_id_from="@product_id_1"
catentry_id_to="@product_id_15"
massoc_id="REQUIRES"
quantity="2.0"
rank="1.00000"
/>
```

ここで、

- massoccece_id は、このエントリーの参照番号です。
- massoctype_id は、アソシエーション・タイプの ID です。
  - X-SELL = 関連商品販売。
  - UPSELL = 上位商品販売。
  - ACCESSORY = アクセサリー。
  - REPLACEMENT = 交換。
- catentry_id_from は、アソシエーションのソースであるカタログ・エントリーです。
- catentry_id_to は、アソシエーションのターゲットであるカタログ・エントリーです。
- massoc_id は、セマンティック指定子の ID です。
  - REQUIRES
  - COMES_WITH
  - TEMP

– NONE

- quantity は、このアソシエーションに関連した数量です。
- rank は、表示順序に使用されるシーケンス番号です。

注: 取引管理アソシエーションを追加するたびに、新しい関係を表すよう、massoccece_id の番号が変わります。 catentry_id_from と catentry_id_to の番号が変わり、アソシエーションの新しい商品取引内容が作成されます。

パート 11: ストアへのカタログの関連付け

1. ToolTech サンプル・ストアの既存の store-catalog.xml ファイルを参考にして、カタログ、そのカタログ・グループ、およびカタログ・エントリーを、データベースの中のストアに割り当てることによって、カタログをストアに関連付けます。また、表示ページをカタログ・グループおよびカタログ・エントリーに割り当てることも必要です。この情報を、STORECAT、STORECENT、STORECGRP、 DISPCGPREL、および DISPENTREL の各テーブルに追加します。グローバル化されたカタログを作成する場合、ストアがサポートする各ロケールごとに別個のストア/カタログ関係 XML ファイルを作成します。

```
<storecat
catalog_id="@catalog_id_1"
storeent_id="@storeent_id_1"
mastercatalog="1"
/>
```

ここで、

- catalog_id は、カタログの参照番号です。
- storeent_id は、データベースの中のストア・エンティティーの参照番号です。
- mastercatalog は、ストアのマスター・カタログを指定します。値 1 は、このカタログをマスター・カタログとして指定することを示します。

2. ToolTech サンプル・ストアの次の例を参考にして、ストア/カタログ関係にカタログ・エントリーを追加します。カタログの中の各カタログ・エントリーごとにこのタスクを完了します。

```
<storecent
storeent_id="@storeent_id_1"
catentry_id="@product_id_102"
/>
```

ここで、

- storeent_id は、データベースの中のストア・エンティティーの参照番号です。
- catentry_id は、カタログ・エントリーの参照番号です。

注: catentry_id をストア・エンティティーに追加するたびに、参照番号が既存のカタログ・エントリーに合わせて変化します。

3. ToolTech サンプル・ストアの次の例を参考にして、ストア・エンティティーにカタログ・グループを追加します。カタログの各カタログ・グループごとにこのタスクを完了します。

```
<storecgrp
storeent_id="@storeent_id_1"
catgroup_id="@catgroup_id_1"
/>
```

ここで、

- `storeent_id` は、データベースの中のストア・エンティティーの参照番号です。
- `catgroup_id` は、カタログ・グループの参照番号です。

注: `catgroup_id` をストア・エンティティーに追加するたびに、参照番号が既存のカタログ・グループに合わせて変化します。

.

パート 12: カタログへの税の関連付け

税を、特定のストアのカタログの商品およびサービスに関連付けます。この情報を CATENCALCD テーブルに追加することによって、税額計算コードをカタログ・エントリーと関連付けることが必要です。詳細については、 284 ページの『WebSphere Commerce での税資産の作成』を参照してください。

パート 13: カタログへの配送方法の関連付け

配送方法をカタログの商品およびサービスに関連付けるには、配送計算コードをカタログ・エントリーと関連付けることが必要です。この情報を CATENCALCD テーブルに追加してください。詳細については、 265 ページの『WebSphere Commerce での配送資産の作成』を参照してください。

パート 14: カタログへの配送センターの関連付け

製品を顧客に配送するために、カタログを配送センターに関連付けます。配送センターは、ストアの商品の在庫および配送を管理します。この情報を FFMCENTER テーブルに追加してください。詳細については、 228 ページの『WebSphere Commerce での配送資産の作成』を参照してください。

パート 15: カタログ・エントリーの価格の作成

カタログ・エントリーの価格設定を行います。価格設定は、カタログ・エントリーの価格範囲、およびその価格を使用するために満たす必要のある基準を表します。機能的なカタログを作成するには、オファリング情報をデータベースに追加することが必要です。この情報を、TRADEPOSCN、TDPSCNCNTR、 MGPTRDPSCN、OFFER、および OFFERPRICE テーブルに追加してください。詳細については、 201 ページの『WebSphere Commerce での価格設定資産の作成』を参照してください。あるいは、WebSphere Commerce アクセラレーターの商品管理ツールを使用して、カタログ・エントリーの価格設定の作成または更新を行えます。

パート 16: XML ファイルのロード

データを作成したら、ローダー・パッケージを使用するか、発行ユーティリティーを使って、XML ファイルをデータベースにロードします。ローダー・パッケージの詳細については、 363 ページの『第 10 部 ストアの発行』を参照してください。

注: WebSphere Commerce アクセラレーターの商品管理ツールを使用することによっても、マスター・カタログのカタログ資産を作成できます。商品管理ツールについては、WebSphere Commerce オンライン・ヘルプを参照してください。

ストア・カタログ資産の表示

カタログ、カタログ・グループ、およびカタログ・エントリーをストアに関連付けたなら、データベース中にこれらの関係を作成することによって、カタログ・エントリーとカタログ・グループを表示する JSP テンプレートを割り当てます。これらの関係を XML ファイルの形式で作成します。XML ファイルは、ローダー・パッケージを使用して、データベースにロードできます。

ToolTech サンプルの store-catalog.xml ファイルがストア・アーカイブ・ファイルにあります。 store-catalog.xml ファイルを表示するには、ZIP プログラムを使用して、ストア・アーカイブを解凍します。 store-catalog.xml ファイルは以下のデータ・ディレクトリーにあります。

- *WC_installdir*/samplestores

store-catalog.dtd ファイルは、以下のディレクトリーにあります。

- *WC_installdir*/xml/sar

ストア - カタログ関係を作成する前に、ストア・データ資産を作成してあることを確認してください。以下のタスクを実行します。それぞれのタスクを実行すると、store-catalog.xml ファイルにエントリーが作成されます。

1. ストアの中のカタログ・グループ (カテゴリー) を表示するには、JSP テンプレートをカタログ・グループに割り当てることが必要です。カタログ・グループに特定の表示ページのテンプレートを割り当てることもできますし、すべてのカタログ・グループを表示するためのデフォルト・テンプレートを割り当てることもできます。 ToolTech サンプル・ストアの次の例を参考にして、DISPCGPREL テーブルに情報を追加し、カタログ・グループ・テンプレートを割り当てます。カタログ・グループに割り当てる各テンプレートごとにこのタスクを完了します。

```
<dispcgprel
catgroup_id="@catgroup_id_1"
devicefmt_id="-1"
dispcgprel_id="@dispcgprel_id_1"
mbrgrp_id="0"
pagename="CategoryDisplay.jsp"
storeent_id="@storeent_id_1"
rank="0"/>
```

ここで、

- catgroup_id は、このページ名が表示されるカタログ・グループの参照番号です。値 0 は、このページ名がすべてのカタログ・グループに使用されることを示します。
- devicefmt_id は、ページが表示されるデバイス・タイプの参照番号です。値 -1 は、このテンプレート・ページが HTTP ブラウザーに使用されることを示します。
- dispcgprel_id は、このエントリーの参照番号です。
- mbrgrp_id は、このテンプレート・ページが表示されるメンバー・グループの参照番号です。値 0 は、このテンプレート・ページがすべてのメンバー・グループに使用されることを示します。
- pagename は、表示テンプレート・ページの名前です。
- rank は、複数のページが選択基準を満たすときに、関係を切るために使用されるシーケンス番号です。

注: JSP テンプレートをカタログ・グループに関連付けるたびに、 catentry_id のシーケンスが既存のカタログ・エントリーに合わせて変化します。

2. ストアのカタログ・エントリー (商品、アイテム、パッケージ、スタティック・パッケージ、バンドル、およびダイナミック・キット) を表示するには、 JSP テンプレートをカタログ・エントリーに割り当てることが必要です。すべてのカタログ・エントリーを表示するためのデフォルト・テンプレートを割り当てることもできますし、カタログ・エントリーのタイプごとにデフォルトを割り当てることもできます。たとえば、製品、アイテム、あるいは特定のカタログ・エントリーに、それぞれ別個のテンプレートを割り当てるという方法です。 ToolTech サンプル・ストアの次の例を参考にして、DISPENTREL テーブルに情報を追加し、テンプレートを割り当てます。カタログ・エントリーに割り当てる各テンプレートごとにこのタスクを完了します。

```
<dispentrel
auctionstate="0"
catentry_id="0"
catenttype_id="ProductBean"
devicefmt_id="-1"
dispentrel_id="@dispentrel_id_1"
mbrgrp="0"
pagename="ProductDisplay.jsp"
storeent_id="@storeent_id_1"
rank="0"/>
```

ここで、

- auctionstate は、このテンプレート・ページが、オークション対象のカタログ・エントリーを表示することを示します。
  - 0 = オークション・テンプレートではない。
  - 1 = オークション・テンプレート。
- catentry_id は、このページ名が表示されるカタログ・エントリーの参照番号です。値 0 は、このページ名がすべてのカタログ・エントリーに使用されることを示します。
- catenttype_id は、このページを使用して表示されるカタログ・エントリーのタイプです。
  - ProductBean = 商品を表示する。
  - ItemBean = アイテムを表示する。
  - PackageBean = パッケージを表示する。
  - BundleBean = バンドルを表示する。
  - DynamicKitBean = ダイナミック・パッケージを表示する。
- devicefmt_id は、ページが表示されるデバイス・タイプの参照番号です。値 -1 は、このテンプレート・ページが HTTP ブラウザーに使用されることを示します。
- dispentrel_id は、カタログ・エントリーの参照番号です。
- mbrgrp は、このテンプレート・ページが表示されるメンバー・グループの参照番号です。値 0 は、このテンプレート・ページがすべてのメンバー・グループに使用されることを示します。
- pagename は、表示テンプレート・ページの名前です。
- storeent_id は、このページが表示されるストアの参照番号です。

- rank は、複数のページが選択基準を満たすときに、関係を切るために使用されるシーケンス番号です。

注: JSP テンプレートをカタログ・エントリーに関連付けるたびに、catentry_id のシーケンスが既存のカタログ・エントリーに合わせて変化します。

セールス・カタログの作成

WebSphere Commerce ストアでは、マスター・カタログとセールス・カタログという 2 種類のカタログを使用できます。セールス・カタログは、マスター・カタログに必要な構造上の制約を受けません。セールス・カタログは、柔軟性の高い表示構造を用意して、各ストアの要件にマッチしたカタログを作成するためのカタログです。

特に、セールス・カタログは、マスター・カタログに必要な以下の構造上の制約を受けません。

- マスター・カタログは、適切なツリーでなければならない。つまり、循環が存在していてはならず、親カテゴリー A にサブカテゴリー B があるとき、B および B のいかなるサブカテゴリーも A の親カテゴリーになることはできません。
- 1 つの商品が複数のカテゴリーに属することはできません。

FashionFlow サンプル・ストア・カタログを変更して、セールス・カタログを作成するための手順をこれから見ていきます。ここでは一部の商品を複数のカテゴリーに分類するので、結果として生成されるカタログはマスター・カタログの分類からは外れることになります。標準的なセールス・カタログは、カテゴリー関係テーブル (サブカテゴリー関係が含まれる CATGRPREL と、カテゴリーと商品の関係が含まれる CATGPENREL) に情報を追加することによって作成します。例では FashionFlow を使用していますが、独自のマスター・カタログでも、カタログ情報、構造、および設計に合わせて適切な調整を行うことにより、以下の基本的なステップを行うことができます。

2 番目のカテゴリーへの商品の追加

この例では、元の構造を保持しながら、1 つのカテゴリーから別のカテゴリーに商品をコピーする方法を示します。**「ホーム・ページ販売」**カテゴリーに含まれる **「サマー・ナイトガウン」** 商品は、最上位カテゴリー**「レディース・ファッション」**の**「スリープ・ウェア」** サブカテゴリーにも所属することができます。以下の説明では、**「サマー・ナイトガウン」**商品とその SKU を**「スリープ・ウェア」** カテゴリーにコピーする方法を示します。

商品を 2 番目のカテゴリーに追加して FashionFlow サンプル・ストア・マスター・カタログをセールス・カタログに変更するには、以下のようにします。

1. FashionFlow ストア・アーカイブを発行して FashionFlow サンプル・ストアを作成する。 WebSphere Commerce には、米国英語版と他の 9 つの各国語版の FashionFlow が用意されています。 FashionFlow_en_US_*locale*.sar ファイルの 1 つを発行用に選択してください。
2. catalog.xml ファイルをエディターで開く。このファイルは、以下の WebSphere Commerce ディレクトリーにあります。

- *WC_installdir*/samplestores/FashionFlow/*locale*/data

3. catalog.xml ファイルで CATGPENREL データ・セクションを見つける。**「サマー・ナイトガウン」**の新しい商品エントリーを作成します。これはもともと、**「ホーム・ページ販売」**カテゴリーの下にある商品です。 CATGPENREL セクションの下に、以下の部分を追加してこの商品を含めます。

```
<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@product_id_2692"
sequence="2"
/>
```

ここで、

- catgroup_id は、FashionFlow サンプル・ストアによって定義された、カタログ・グループ内部参照番号です。この例では、@catgroup_id_18 は**「レディース・スリープ・ウェア」**カテゴリーです。
- catalog_id は、FashionFlow サンプル・ストアによって定義された、カタログの内部参照番号です。
- catentry_id は、FashionFlow サンプル・ストアによって定義された、カタログ・エントリー内部参照番号です。この例では、@catentry_id_2692 は**「サマー・ナイトガウン」**カテゴリーです。
- sequence は、FashionFlow サンプル・ストアによって定義された、カタログ・グループの内容の表示順序を決定する番号です。この例では、**「サマー・ナイトガウン」**商品が最後に表示されます。

4. **「サマー・ナイトガウン」**商品エントリーを追加したら、 FashionFlow サンプル・ストアで定義されているように、商品の SKU エントリーを CATGPENREL セクションの下に追加する。現在、**「サマー・ナイトガウン」**商品には、10 個の定義済み SKU が含まれています。 CATGPENREL セクションの下に、以下の部分を追加してこれらの SKU を含めます。

```
<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2695"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2696"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2697"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2698"
sequence="2"
/>
```

```
<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2699"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2700"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2701"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2702"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2703"
sequence="2"
/>

<catgpenrel
catgroup_id="@catgroup_id_18"
catalog_id="@catalog_id_1"
catentry_id="@catentry_id_2704"
sequence="2"
/>
```
ここで、

- `catgroup_id` は、FashionFlow サンプル・ストアによって定義された、カタログ・グループ内部参照番号です。この例では、`@catgroup_id_18` は**「レディース・スリープ・ウェア」**カテゴリーです。
- `catalog_id` は、FashionFlow サンプル・ストアによって定義された、カタログの内部参照番号です。
- `catentry_id` は、FashionFlow サンプル・ストアによって定義された、カタログ・エントリー内部参照番号です。この例では、`@catentry_id_2695` ～ `@catentry_id_2704` は、**「サマー・ナイトガウン」**商品用に定義されている 10 個の SKU を表しています。
- `sequence` は、FashionFlow サンプル・ストアによって定義された、カタログ・グループの内容の表示順序を決定する番号です。この例では、**「サマー・ナイトガウン」**の SKU が最後に表示されます。

5. catalog.xml ファイルを保管します。

6. 変更内容を表示するには、変更された FashionFlow ストア・アーカイブを管理コンソールによって発行するか、 437 ページの『データベース資産グループのロード』の説明に従い、ローダー・パッケージを使用して catalog.xml ファイルをロードします。

WebSphere Commerce でのカタログ資産の管理

時間とともに、マスター・カタログのデータベース資産情報を更新する必要が生じます。カタログの管理は継続的なプロセスであり、継続的に商品取引の追加や除去を行ったり、カテゴリーやカタログ・グループの作成や関連づけを行ったり、説明や価格などの商品情報を更新したりしなければなりません。

カタログ資産を変更するには、ストアの既存のデータベース・エントリーと catalog.xml ファイルを使って、WebSphere Commerce の XML データを編集します。 WebSphere Commerce のサンプル・ストアの XML ファイルを参考にしてください。それらのファイルは、以下のデータ・ディレクトリーにあります。

- *WC_installdir*/samplestores

注: これらのサンプル・ファイルは、 FashionFlow サンプル・ストアに基づいており、カタログ資産情報を変更するために編集しなければならない XML 要素を示しています。

カタログ・グループ

WebSphere Commerce カタログにカタログ・グループを作成するには、データベース・テーブルの CATGROUP と CATGRPDESC を使用します。 catalog.xml ファイルでは、典型的なカタログ・グループが次のようになっています。

```
<catgroup
catgroup_id="@catgroup_id_1"
member_id="&MEMBER_ID"
identifier="Accessories"
markfordelete="0"
/>
```

catgroup_id は、カタログ・グループの内部参照番号です。 WebSphere Commerce では、各カタログ・グループに内部参照番号が割り当てられており、カタログ・エントリーを追加するときには、その番号でカタログ・グループを識別することになります。 identifer は、カタログ・グループの外部名です。このいずれも、データベース資産内で一意であり、重複は認められません。

ロケール固有の catalog.xml ファイルには、名前と説明が記述されています。ストアがサポートしている各ロケールには、そのロケール固有のファイルが必要です。翻訳可能な情報を含んだ典型的なカタログ・グループは、次のようになっています。

```
<catgrpdesc
language_id="&en_US"
catgroup_id="@catgroup_id_1"
name="Accessories"
shortdescription="Accessories"
longdescription="Accessories"
published="1"
/>
```

language_id は、このカタログ情報の言語です。この ID は、ストアがサポートしている言語に合わせて変更する必要があります。 name は、顧客に表示される名前、 shortdescription は、顧客に表示されるカタログ・グループの簡単な説明、longdescription は、顧客に表示されるカタログ・グループの詳しい説明です。

新しいカタログ・グループを作成するときには、上記の構造に合わせて情報を指定してください。

注:

1. 上記の例では、 identifer と name の指定内容が同じですが、もちろん同じ指定にする必要はありません。たとえば、カタログ・グループの名前を **Complementary Additions** に変更するとしましょう。その場合は、name の情報だけを変更すればよいわけで、 identifer の情報を変更する必要はありません。
2. カタログ・グループを削除する場合は、それに合わせて catgroup_id の情報を更新する必要があります。たとえば、カタログ・グループを削除するときに、その中のカタログ・エントリーも一緒に削除するのであれば、 XML エントリー全体を削除します。しかし、カタログ・エントリーを残すのであれば、catgroup_id を別のグループに変更する必要があります。

カタログ・エントリー

WebSphere Commerce カタログにカタログ・エントリーを作成するには、データベース・テーブル CATENTRY と CATENTDESC の情報を使用します。カタログ・エントリーは、商品、アイテム、パッケージ、バンドル、スタティック・パッケージ、またはダイナミック・パッケージのいずれかになります。 catalog.xml ファイルでは、典型的なカタログ・エントリーが次のようになっています。

```
<catentry
catentry_id="@product_id_102"
baseitem_id="@baseitem_id_102"
member_id="&MEMBER_ID"
catenttype_id="ProductBean"
partnumber="product-sku-nf-102"
mfpartnumber="product-sku-nf-102"
mfname="FashionFlow"
markfordelete="0"
buyable="1"
/>
```

catentry_id は、商品カタログ・エントリーの内部参照番号です。 baseitem_id は、カタログ・エントリーと関連のある基本アイテムです (これは、在庫管理のための情報です)。 partnumber は、カタログ・エントリーの部品番号を識別する参照番号です。 mfpartnumber は、カタログ・エントリーを識別するためにメーカーが使用する部品番号です。このいずれも、データベース資産内で一意であり、重複は認められません。

catenttype_id は、カタログ・エントリーのタイプを識別します。タイプは、ItemBean、ProductBean、PackageBean、StaticBean、BundleBean、DynamicKitBean のいずれかです。

ロケール固有の catalog.xml ファイルには、名前と説明が記述されています。ストアがサポートしている各ロケールには、そのロケール固有のファイルが必要です。こ

のファイルには、商品取引イメージも含まれています。翻訳可能な情報を含んだ典型的なカタログ・グループは、次のようになっています。

```
<catentdesc
catentry_id="@product_id_102"
language_id="&en_US"
name="Belt"
shortdescription="Classic belt"
longdescription="This classic belt looks great with your favorite jeans,
or takes you to work in style. 1 1/2 inches wide in full-grain leather
with a solid nickel buckle."
thumbnail="images/mens_accessories_belt_sm.gif"
fullimage="images/mens_accessories_belt.gif"
available="1"
published="1"
/>
```

language_id は、このカタログ情報の言語です。この ID は、ストアがサポートしている言語に合わせて変更する必要があります。 name は、顧客に表示される名前、 shortdescription は、顧客に表示されるカタログ・エントリーの簡単な説明、 longdescription は、顧客に表示されるカタログ・エントリーの詳しい説明です。

新しいカタログ・エントリーを作成するときには、上記の構造に合わせて情報を指定してください。

注:

1. カタログ・エントリーを削除する場合は、それに合わせて固有のエレメントの情報を更新する必要があります。たとえば、カタログ・グループを削除するときに、その中のカタログ・エントリーも一緒に削除するのであれば、 XML エントリー全体を削除します。しかし、カタログ・エントリーを残すのであれば、 catgroup_id を別のグループに変更する必要があります。
2. 商品をまず作成しなければ、他の種類のカタログ・エントリーを作成することはできません。

XML ファイルを手作業で変更したくない場合は、商品管理ツールを使用します。

商品管理ツール

WebSphere Commerce アクセラレーターの商品管理ツールを使用すれば、さまざまなウィザードやノートブックを使用してストアのマスター・カタログにある商品を管理できます。さらに、カタログ・エントリー情報を直接更新することができる、商品管理動的テーブルを使用することもできます。以下のように、カタログの内容を更新したり、新しいカタログ・データを作成したりできます。

- ウィザードやノートブックを使用して、商品や商品詳細情報を作成、更新、および削除する。商品は、SKU、つまり最終的に顧客に販売される個々のアイテムのテンプレートとして機能します。商品詳細情報には、商品コード (商品を一意的に識別する)、商品の名前と説明、取引管理オプション (商品を顧客に表示したり、商品が特別な販売促進品であることを示したりする) 、商品イメージ、税と配送の仕様、商品に割り当てられる割引、および製造元情報が含まれます。
- 購入用の SKU (またはアイテム) を生成、更新、および削除する。 SKU は販売用の商品取引におけるそれぞれのオーダー可能アイテムを表します。特定の商品に関連するすべての SKU は同じ属性セットを示し、それらの属性値によって区

別されます。 SKU への追加や変更には、商品と同じ情報が含まれます。ただしこれは、オーダー可能かどうかに基づきます。

- カテゴリー (またはカタログ・グループ) を作成、更新、および削除する。カテゴリーは、ストアが提供する商品やサービスを編成するために使用される同様のプロパティーを持つオブジェクトのグループです。カテゴリーやカテゴリー詳細 (たとえば、カテゴリー・コード、名前、および親カテゴリーやイメージなどの説明) を作成、変更、および削除することにより、マスター・カタログのカテゴリー階層を管理できます。
- 親カテゴリーを選択したり、商品と SKU を 1 つのカテゴリーから別のカテゴリーに移動したりすることにより、商品と SKU をカテゴリーに関連付ける。
- 商品の属性と属性値を作成する。属性と属性値との可能な組み合わせが、それぞれ新しい SKU となります。属性値を SKU に割り当てる前に、それを定義しておく必要があります。属性とその値を作成したら、名前、説明、(テキスト、整数または 10 進数)、および属性と属性値が表示される順序などの情報を作成または更新できます。
- カタログの価格設定の作成、更新、削除、および商品への関連付けを行う。商品や SKU の価格を 1 つ以上の通貨で定義し、それとともに、その価格を使用するために満たさなければならない条件 (たとえば、1 つだけ買ったときとまとめ買いしたときの価格設定) のセットを定義できます。

各タスクの詳細については、オンライン・ヘルプの『Product Management (商品管理)』セクションを参照できます。

注:

1. 商品管理ツールは、小さい変更を行う場合にのみ使用をお勧めします。カタログの大きな更新 (季節ごとの商品取引の追加や削除、または在庫一掃セールの準備など) では、ローダー・パッケージを使用してください。
2. カタログ・データの変更内容は、キャッシングを使用不可にするか、現在キャッシングされている JSP ページを削除しない限り、ストアで表示できません。詳細については、WebSphere Commerce オンライン・ヘルプで CacheDelete コマンドを参照してください。 CacheDelete コマンドを実行すると、動的ページ・キャッシュのリモート・クリーンアップが開始され、ファイル・システムへの直接アクセスなしにキャッシュを管理できるようになります。このコマンドを使用する前に、「自動ページ無効」が使用可能になっていることを確かめてください。このコマンドを使用するには管理者としてログインしなければならないことに注意してください。

ローダー・パッケージ

ローダー・パッケージ (以前は Catalog Manager の一部として知られていた) を使用してカタログを保守することもできます。既存の商品情報をデータベースに大量にインポートする場合は、ローダー・パッケージが理想的です。これは、WebSphere Commerce でカタログ情報の作成や管理を行うための主なツールです。このパッケージは主に、データを準備したり WebSphere Commerce データベースにロードしたりするためのコマンド・ユーティリティーで構成されています。また、ローダー・パッケージでは、データベースからデータを XML 文書として抽出したり、XML データを代替 XML 形式に変換したり、文字区切り変数形式と XML データ形式の間でデータを変換したりすることもできます。

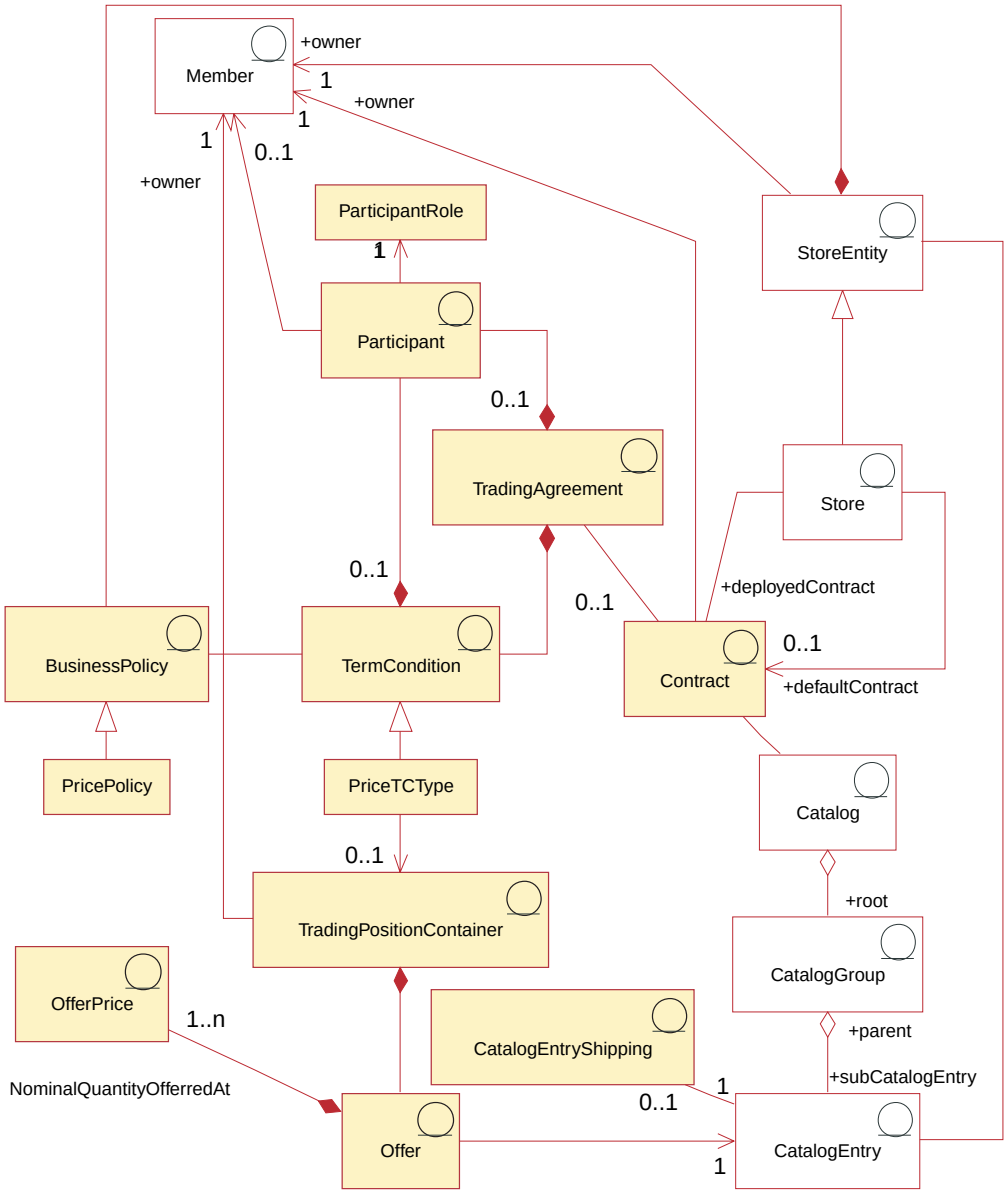
詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

第 17 章 価格設定資産

価格設定は、カタログ・エントリーの価格、およびその価格を使用するために満たす必要のある基準を表します。機能カタログを作成するには、価格設定情報をデータベースに追加することが必要です。価格設定情報を、ローダー・パッケージを使用して、データベースにロード可能な XML ファイルの形式で作成できます。または、価格設定データの量が少ない場合は、 WebSphere Commerce アクセラレーターの商品管理ツールを使用することもできます。

WebSphere Commerce の価格設定について

以下の図は、WebSphere Commerce Serverにおける価格設定資産を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、111 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、487 ページの『付録 A. UML の凡例』を参照してください。

オファー

オファー、つまり価格設定は、同じ商品またはアイテムについて、顧客または組織によって異なる価格を提供するものです。オファーは、カタログ・エントリーの価格と、その価格で購入するために顧客が満たす必要のある基準 (購入数量など) を表します。たとえば、商品取引またはサービスは、子供、学生、大人、および高齢者によって価格が異なることがよくあります。 WebSphere Commerce では、これは取引位置とも呼ばれており、取引位置コンテナーの一部です。

オファー価格

オファー価格 は、取引条件または契約によってストアがカタログ・エントリーをオファーする価格です。 1 つのオファーでは、複数の通貨で定義された 1 つ以上のオファー価格を指定できます。

取引位置コンテナー

オファーは、メンバーが所有する、*取引位置コンテナー* の一部です。取引位置コンテナーには、取引位置があります。これは、すべての顧客が使用できるようにするか、あるいは取引条件や契約、そして契約の条項によって特定のグループ内の顧客だけが使用できるようにすることができます。契約において、取引位置コンテナーは、価格ビジネス・オブジェクトの 1 つであり、それは複数の価格ビジネス・ポリシーで参照したり、 1 つのストアで、または 1 つのストア・グループ中のすべてのストアで共用したりできます。取引位置コンテナーは、*価格表* とも呼ばれます。

契約条件

契約条件 は、取引条件の振る舞いとプロパティーを定義します。ストアの操作のいくつかの局面はビジネス・ポリシーによって定義されているので、契約条件の多くはビジネス・ポリシーを参照します。

価格設定条件の種類

Professional Business *価格設定条件 は、契約に基づいて購入できる商品と顧客がその商品に支払う価額を定義します。契約には、以下の価格設定条件のうち、少なくとも 1 つが必要です。 WebSphere Commerce では以下の価格設定条件が利用できます。*

カスタマイズ済み価格表

この条件は、販売用の商品リストとその価格の両方を、契約内での販売用にカスタマイズすることを指定します。アイテムは、ストア・カタログの特定のセクションに限定されることはなく、ストア・カタログのどこにあるものでも可能です。

価格調整付き総合カタログ

この条件は、販売用ストア・カタログの購入可能な商品すべてに、ストア・カタログで定義された基本価格からのパーセント調整 (利掛けまたは割引) を提供します。価格調整が指定されていないなら、アイテムは基本価格で販売されます。

価格調整付き価格表

この条件は、販売用価格表の購入可能な商品すべてに、ストア・カタログで定義された基本価格からのパーセント調整 (利掛けまたは割引) を提供します。価格調整が指定されていないなら、アイテムは基本価格で販売されます。

選択的価格調整付き価格表

この条件は、価格調整付き価格表と類似していますが、価格調整が価格表全体に適用されない点が異なります。価格調整は、価格表のサブセットに対してなされます。価格表のそのサブセットは、商品セットビジネス・ポリシーか、またはカスタマイズ済み商品セットのいずれかです。商品セットのタイプ間の違いについての情報は、 WebSphere Commerce オンライン・ヘルプにある『契約条件』のトピックを参照してください。

フィルター付きカタログ

この条件は、販売用ストア・カタログの購入可能な商品すべてに、ストア・カタログで定義された基本価格からのパーセント調整 (利掛けまたは割引) を提供します。この条件は、販売用カテゴリー (または特定の商品およびアイテムのリスト) の購入可能な商品すべてに、この条件により参照される価格表で定義された基本価格からのパーセント調整 (利掛けまたは割引) も提供します。さらにこの条件で、契約の中でどのカテゴリー、商品、およびアイテムが販売用でどれがそうでないかも指定できます。カテゴリー商品セットは、商品セットのビジネス・ポリシーとして振る舞います。アイテム商品セットは、カスタマイズされた商品セットとなります。

取引条件

Business *取引条件* は、契約、RFQ、ビジネス・アカウント、またはオークションです。取引条件は、セラーとバイヤーの間でネゴシエーションされた合意事項であり、それによりバイヤーは、特定のアイテムを、契約の中で指定されている条件およびビジネス・ポリシーで購入できます。たとえば、それにより顧客は、価格設定条件に従って、指定された期間に指定された価格でストアから商品を購入することができます。 WebSphere Commerce において、すべての顧客は、契約に従ってストアでの買い物をしなければなりません。ストアでは 1 つ以上の契約をデプロイすることができ、そのうちの 1 つをデフォルトの契約として指定できます。デフォルト契約には、一連のストア・デフォルト・ポリシーに関連する一連の条件が含まれます。取引条件には、それぞれ役割の異なる 0 または複数の参加者が含まれます。

参加者

参加者 は、取引条件または契約条件の一部となることができます。参加者は、それ自身メンバー・グループや組織などであり得る 1 メンバーです。契約においてバイヤーの役割の参加者が指定されている場合、バイヤーがその契約に従って買い物をするためには、バイヤーはバイヤー参加者のメンバーでなければなりません。契約の条件にも、0 または複数の参加者を含めることができます。

参加者の役割

ある参加者の*参加者の役割* は、以下のうちのいずれか 1 つです。

- クリエーター
- セラー
- バイヤー
- サプライヤー
- 承認者
- アカウント・ホルダー
- バイヤー連絡先
- セラー連絡先
- 代理人
- 管理者

契約

商品のオファー価格を含む*契約*。 WebSphere Commerce では、すべての顧客は契約の下でショッピングを行わなければなりません。契約により、顧客は、指定された価格で、指定された期間に渡り、契約に定められている契約条件とビジネス・ポリシーに基づいて、ストアから商品を購入できます。ストアは 0 またはそれ以上の契約を所有し、最低 1 つのデフォルト契約を所有しています。

ビジネス・ポリシー

Business *ビジネス・ポリシー* とは、ストアまたはストアのグループが従う一連のルールであり、ビジネス・プロセス、業界慣例、およびストアやストア・グループのオファリングの範囲や特性を定義するものです。ビジネス・ポリシーの規則をインプリメントする 1 つ以上のビジネス・ポリシー・コマンド、規則が適用されるビジネス・オブジェクトへの参照、およびビジネス・ポリシー・コマンドの操作を構成する一連のプロパティーの組み合わせでビジネス・ポリシーは施行されます。

価格設定ポリシー

価格設定ポリシー には、価格表への参照が含まれており、価格表に対するビジネス・ポリシーの適用方法を定義する複数のビジネス・ポリシー・コマンドに関連付けることができます。ポリシーは、ストアに対して定義したり、ストア・グループに対して定義したりできます。ストア・グループに関してポリシーを登録すると、そのポリシーはそのグループに含まれるすべてのストアで使用されます。

カタログ・エントリー配送

商品の配送用の包装方法についての情報を含む、*カタログ・エントリー配送* 情報。各カタログ・エントリーには、さまざまな配送情報を定義できます。たとえば、包装時の商品の高さ、重量、長さなどです。

他の価格設定資産

以下の資産が価格設定と関連付けられています。

- *member* は取引位置コンテナーです。取引位置コンテナーの所有者は 1 人だけです。
- WebSphere Commerce Server データベース内のストアを表すストア・エンティティー。
- *カタログ* には、契約の中で参照されるカタログ・エントリーが含まれます。カタログには、オンライン・カタログ用のすべての階層情報およびナビゲーション情報が入っています。カタログは、オンライン・ストアで表示および購入可能な、カタログ・グループとカタログ・エントリーの集合です。
- *カタログ・グループ*、つまりカテゴリーとは、ナビゲーション上の目的およびカタログの区分化の目的で作成される、カタログ・エントリーの一般的なグループ分けのことです。 1 つのカタログ・グループは 1 つのカタログに所属し、複数のカタログ・グループまたはカタログ・エントリーを含むことができます。カタログ・グループは複数のカタログと関連付けることができます。
- *カタログ・エントリー* とは、オンライン・カタログでオーダー可能な商品のことです。カタログ・エントリーはカタログ・グループに所属します。オファーは常に 1 つのカタログ・エントリーに関連付けられます。

WebSphere Commerce Server での価格設定資産の構造の詳細については、WebSphere Commerce オンライン・ヘルプの『価格設定のオブジェクト・モデル』と『データ・モデル』を参照してください。

WebSphere Commerce での価格設定資産の作成

価格設定資産を作成するためには、以下の 2 つの選択肢があります。

- WebSphere Commerce アクセラレーターの商品管理ツールを使用して価格設定を作成する。 WebSphere Commerce アクセラレーターのツールを使用する方法は、極めて小さなカタログの価格設定を作成する場合に最適です。
- WebSphere Commerce ローダー・パッケージでロード可能な XML ファイルで価格設定を作成するか、管理コンソールを介して発行できるストア・アーカイブの一部として価格設定を作成する。この方法は、大量のデータを作成するのに最適です。

WebSphere Commerce アクセラレーターの商品管理ツールを使用して価格設定を作成する場合の詳細については、 WebSphere Commerce オンライン・ヘルプを参照してください。 XML ファイルで価格設定を作成する場合の詳細については、『XML ファイルによる価格資産の作成』を参照してください。

XML ファイルによる価格資産の作成

価格設定資産を XML ファイルの形式で作成します。XML ファイルは、ローダー・パッケージを使用して、データベースにロード可能です。ローダー・パッケージの詳細については、 363 ページの『第 10 部 ストアの発行』を参照してください。

1. サンプル・ストアの価格設定資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。

   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

- *WC_installdir*/samplestores

注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

各サンプル・ストアには、offering.xml ファイルが 2 つ組み込まれています。これには価格設定情報が含まれています。ストア・アーカイブの offering.xml ファイルを表示するには、 ZIP プログラムを使用してこれを解凍します。offering.xml ファイルはデータ・ディレクトリーにあります。言語ごとの offering.xml は、データ・ディレクトリーのロケールごとのサブディレクトリーにあります。

2. 489 ページの『付録 B. データの作成』に記載されている情報を確認します。
3. サンプル・ストア・アーカイブの offering.xml ファイルの 1 つをコピーするか、または新しいファイルを作成することによって、offering.xml ファイルを作成します。詳細については、offering.xml に対応する DTD ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。
   - *WC_installdir*/xml/sar
4. 取引位置コンテナーを作成します。ストアの品物に価格を付けるには、まず取引位置コンテナーを作成しなければなりません。取引位置コンテナーを作成するためには、情報を TRADEPOSCN テーブルに追加します。
   a. 次の例を参考にして、XML ファイルの TRADEPOSCN テーブルに取引位置コンテナーを作成します。

```
<tradeposcn
tradeposcn_id="@tradeposcn_id_101"
member_id="@seller_b2b_mbr_id"
markfordelete="0"
name="ToolTech"
precedence="0"

/>
```

   ここで、
   - tradeposcn_id は、生成される固有キーです。
   - @seller_b2b_mbr_id は取引位置コンテナーの所有者です。 FashionFlow サンプル・ストアでは、これを @Member_ID; に置き換えてください。
   - markfordelete は次のとおりです。
     - 0 = TradingPositionContainer は使用可能。
     - 1 = TradingPositionContainer は削除対象としてマークされており (DBClean ユーティリティーを参照)、使用できない。
   - name は、特定の所有者ごとに固有の、取引位置コンテナーの簡略名です。
   - precedence は、特定の時点に複数の取引位置コンテナーが適格であった場合に、 PRECEDENCE の一番高いコンテナーが使用されることを示します。
5. CATGRPTPC テーブルに情報を追加することによって、マスター・カタログを取引位置コンテナーに関連付けます。マスター・カタログを取引位置コンテナーに関連付ける場合、マスター・カタログのすべてのカタログ・エントリーに標準価格がなければなりません。マスター・カタログの作成の詳細については、 187 ページの『ストア・カタログ資産の表示』を参照してください。

a. 次の例を参考にして、CATGRPTPC テーブルに情報を追加することにより、マスター・カタログを取引位置コンテナーに関連付けます。

```
<catgrptpc
catalog_id="@catalog_id_1"
tradeposcn_id="@tradeposcn_id_101"
/>
```

ここで、

- catalog_id は、マスター・カタログです。
- tradeposcn_id は、取引位置コンテナーです。

6. OFFER および OFFERPRICE テーブルに情報を追加することによって、カタログ・エントリーにオファーおよびオファー価格を作成します。
   a. 次の例を参考にして、OFFER テーブルに情報を追加することにより、カタログ・エントリーにオファーを作成します。価格を作成する前に、カタログ・エントリーを作成しておかなければならないことに注意してください。カタログ・エントリーの作成の詳細については、 187 ページの『ストア・カタログ資産の表示』を参照してください。

```
offer
offer_id="@offer_id_138"
startdate="2000-06-19 00:00:00.000000"
catentry_id="@product_id_102"
precedence="0"
published="1"
identifier="1"
flags="1"
tradeposcn_id="@tradeposcn_id_101"
/>
```

ここで、

- offer_id は、生成される固有キーです。
- startdate は、このオファーが有効である時刻範囲の開始日です。
- catentry_id は、販売用にオファーされるカタログ・エントリーです。
- precedence は、特定の時点で複数のオファーが有効な場合に、PRECEDENCE の一番高いオファーが使用されることを示します。
- published は次のとおりです。
  - 0 = 発行されない (一時的に使用不可)
  - 1 = 発行される
  - 2 = 削除のマーク (発行されない)
- identifier は、指定されたカタログ・エントリーと取引位置コンテナーと一緒に、このオファーを固有に識別する番号です。
- flags は次のとおりです。
  - 1 = shiptoAddressRequired - 1 の場合には、 OrderPrepare は OrderItem がこのオファーを参照していて配送先住所がなければ、エラーを返します。
- tradeposcn_id は、このオファーが含まれている取引位置コンテナーです。

   b. 次の例を参考にして、OFFERPRICE テーブルに情報を追加することにより、カタログ・エントリーにオファー価格を作成します。オファー価格は、カタ

ログ・エントリーが販売用にオファーされる際の実際の価格です。価格を作成する前に、カタログ・エントリーを作成しておかなければならないことに注意してください。カタログ・エントリーの作成の詳細については、 187 ページの『ストア・カタログ資産の表示』を参照してください。

```
<offerprice
offer_id="@offer_id_138"
currency="USD"
price="590.00"
/>
```

ここで、

- `offer_id` は、この価格と関連したオファーです。
- `currency` は、価格がオファーされる際の通貨です。
- `price` は、オファーで参照される商品の名目数量 (CATENTSHIP.NOMINALQUANTITY を参照) の価格です。

注: ストアに複数の通貨を表示するには、 OFFERPRICE テーブルの中で通貨ごとに別個の XML エントリーを作成してください。たとえば、カナダ・ドルの通貨を表示するには、新しい XML エントリーの中で `currency="CAD"` を使用します。 `price` の値が変更され、カナダ・ドルでの価格が反映されます。 あるいは、変換を使用することにより、顧客が選択する通貨に基づき、さまざまに異なるレートで表示することもできます。詳細については、 252 ページの『XML ファイルを使用した通貨資産の作成』を参照してください。

c. カタログのすべてのカタログ・エントリーに関して、ステップ a と b を繰り返します。

@ と **&** の使用法の詳細については、 489 ページの『付録 B. データの作成』を参照してください。

第 18 章 契約資産

WebSphere Commerce では、すべての顧客は契約の下でショッピングを行わなければなりません。契約により、顧客は、指定された価格で、指定された期間に渡り、指定された契約条件で、ストアから商品を購入できます。ストアのカタログをブラウズするときに顧客が表示できるのは、ストアと結んだ契約範囲内の商品だけです。

ストアと契約していない顧客 (ゲスト・ショッパーなど) がストアでショッピングできるようにしたり、顧客の契約の範囲外の商品を購入できるようにしたりするには、ストアにデフォルト契約 が必要です。

重要

WebSphere Commerce Professional Edition および WebSphere Commerce - Express はストアのデフォルト契約しかサポートしません。

ストアのデフォルト契約以外の契約をサポートするのは WebSphere Commerce Business Edition だけです。

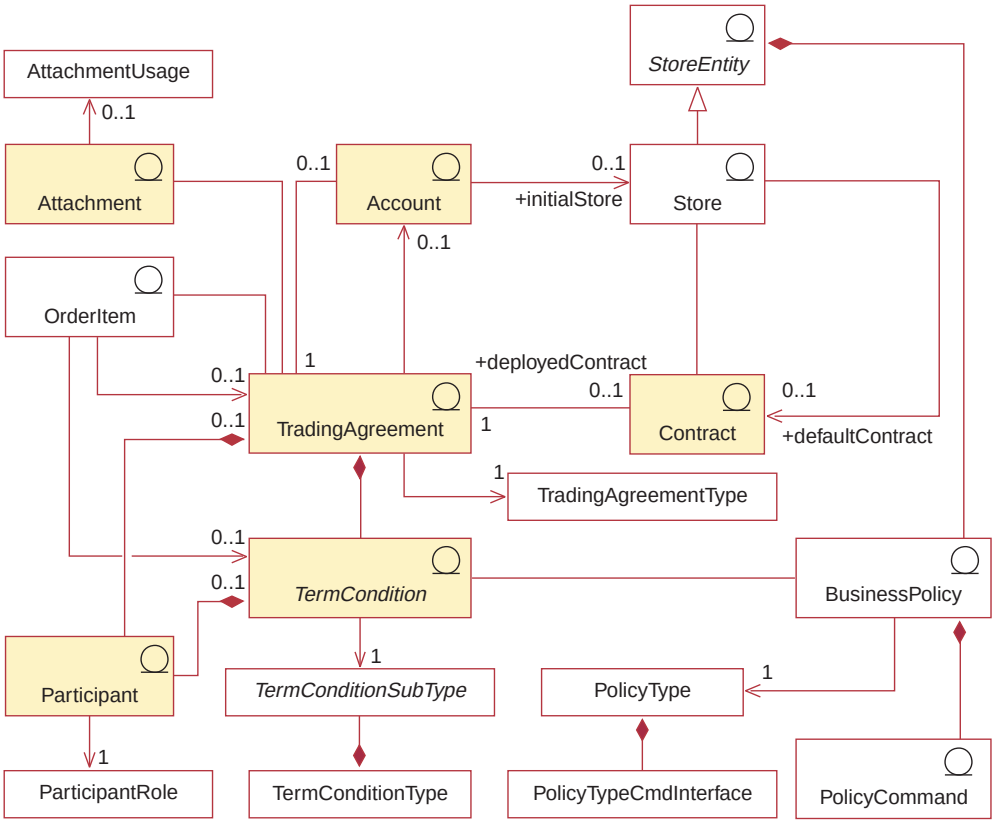
すべての顧客がストアでショッピングできるようにするには、 WebSphere Commerce で作成されたストアに次のものを含めることが必要です。

- ビジネス・ポリシー
- デフォルト契約

ビジネス・ポリシーはデフォルト契約によって参照され、すべての顧客がストアでショッピングすることを許可します。

WebSphere Commerce の契約について

次の図は、WebSphere Commerce における契約の構造を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、111 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、487 ページの『付録 A. UML の凡例』を参照してください。

アカウント (ビジネス・アカウント)

Business

ビジネス・アカウントは、バイヤー組織とセラー組織との間の関係を表します。ビジネス・アカウントは、さまざまな取引条件をまとめたり、バイヤーとセラーの間の関係に関連した契約条件を指定するために使用できます。たとえば、送り状のカスタマイズ、注文の検査、またはセラーに対するバイヤーの貸付限度額などです。

契約は、バイヤーとセラー間の合意事項を表すので、ビジネス・アカウントと関連付けられます。この例外は、ビジネス・アカウントと関連付けられないストア・デフォルト契約です。ビジネス・アカウントは、多数の契約をアカウントに関連付けることができます。

ビジネス・アカウントは、取引条件の 1 タイプです。取引条件の詳細については、207 ページの『取引条件』を参照してください。

重要: ビジネス・アカウントは、WebSphere Commerce Business Edition でのみサポートされます。

契約

ストアに関連付けられるアクティブな契約のタイプが 2 つあります。デプロイ契約とデフォルト契約です。デプロイ契約は、特定のバイヤー組織または個々のバイヤーと結ばれるもので、ストアを作成した後に、WebSphere Commerce アクセラレーターを使用して作成できます。デプロイ契約は、1 つのビジネス・アカウントと関連付けられます。デフォルト契約は、ストアと他の契約を結んでいないバイヤーの場合のストアのデフォルトの振る舞いを定義するものです。デフォルト契約を作成するには XML を使用する必要があり、1 つのストアに作成できるデフォルト契約は 1 つだけです。契約の詳細については、WebSphere Commerce オンライン・ヘルプ情報を参照してください。デフォルト契約資産の作成の詳細については、213 ページの『WebSphere Commerce でのデフォルト契約資産の作成』を参照してください。

典型的な契約は、以下のエレメントから構成されます。

プロファイル
契約プロファイルには、契約の識別情報が含まれます。この情報には、契約の固有の名前、簡略説明、および契約の有効期間が含まれます。

参加者 契約参加者は、契約の当事者となる組織です。バイヤー組織、セラー組織、および両方の組織の連絡先があります。

契約条件
契約条件は、契約の実際のインプリメンテーションを対象とする規則です。契約条件は、商品価格設定、返品とリファンド、支払い、配送、およびオーダー承認などの情報を対象とします。

添付 契約添付は、ファイル添付などの、前述のエレメント以外の情報を対象とするもので、契約についての追加情報や契約についての一般的な注釈を提供します。 WebSphere Commerce は、実際の添付ではなく、契約添付の Universal Resource Identifiers (URI) を保存します。

参照 契約は他の契約を参照して、その契約条件を共用することができます。たとえば契約 A が契約 B を参照している場合、契約 A の適用を受けるバイヤーは、契約 A のすべての契約条件の適用を受けることに加えて、契約 B のすべての契約条件の適用も受けます。

取引条件

契約は、取引条件の 1 タイプです。 WebSphere Commerce には、バイヤーとセラーの間の相互作用を制御する取引メカニズムが多数用意されています。以下の取引メカニズムが WebSphere Commerce の異なるエディションでサポートされています。

- オークション (Business Edition と Professional Edition の両方でサポートされます)
- Business ビジネス・アカウント
- 契約 (この章の前述の制約事項を参照)

- Business 見積依頼 (RFQ)

これらの取引メカニズムすべてに共通する特性があります。たとえば、どの取引メカニズムにも参加者がいますし、取引メカニズムの振る舞いを制御する規則もあります。取引メカニズムの振る舞いを制御する規則のことを、WebSphere Commerce では*使用条件* と呼びます。

取引条件は、取引メカニズムのインスタンスを表し、取引メカニズムのそのインスタンスのプロパティーを記録します。 WebSphere Commerce の契約、ビジネス・アカウント、および Business RFQ は、それぞれ取引条件に示されます。WebSphere Commerce では、すべてのオークションを制御する単一の取引条件があります。

取引条件を構成するのは、TRADING テーブルに保管されるプロファイル、PARTICIPNT テーブルに保管される参加者、TERMCOND テーブルに保管される契約条件、および ATTACHMENT テーブルに汎用リソース ID (URI) として保管されるオプションの添付です。取引条件は複数の添付を持つことができるので、添付は TRDATTACH テーブルを介して取引条件に関連付けられます。添付は Business RFQ ではサポートされていないことに注意してください。

一般の取引条件に加えて、各タイプの取引条件に特有の追加情報が、それぞれ独自のテーブルに保管されます。CONTRACT には契約固有の情報、 Business RFQ には RFQ 固有の情報、ACCOUNT にはビジネス・アカウント固有の情報が保管されます。

参加者

契約参加者は、それぞれの契約内で特定の役割を持ちます。参加者は、バイヤー組織の連絡先およびセラー組織の連絡先がなることもできます。バイヤーの参加者がヌルであることを契約が指定している場合、ゲストを含むすべてのユーザーに対して契約が適用されます。どの契約も、ヌルのバイヤー参加者を指定することが可能です。

契約条件

契約条件は、取引条件の振る舞いとプロパティーを定義します。契約の場合、契約条件は、バイヤー組織のために契約がどのようにインプリメントされるかを定義します。契約では、契約で販売されるもの、販売されるアイテムの価格、オーダーの支払方法、アイテム返品の処理方法、オーダーの承認方法、およびオーダーの配送元が定義されます。

ストアの操作のいくつかの局面はビジネス・ポリシーによって定義されているので、契約条件の多くはビジネス・ポリシーを参照します。契約条件は、自分が参照するビジネス・ポリシーに対してパラメーターを提供します。ビジネス・ポリシーにパラメーターを提供することによって、各契約ごとにビジネス・ポリシーの振る舞いを変更することができます。 WebSphere Commerce は、以下の契約条件をサポートします (ビジネス・ポリシーを参照する契約条件は、アスタリスク (*) で示されています)。

配送センター

このオプションの契約条件を使用して、契約に基づくオーダーの配送元になる配送センターのリストを指定できます。このリストは、ストアに定義された配送センターのサブセットでなければなりません。配送センターの優先順位はストアによって定義され、契約条件によってオーバーライドされることはありません。

オーダーの承認

この条件は、オーダーを配送する前にそのオーダーが顧客組織から承認される必要があるかどうかを指定します。オプションで税と配送料を含む金額を指定して、その金額より小さな値のオーダーであれば、顧客組織の承認がなくても配送できるようにすることもできます。オーダー合計価格がこの金額よりも大きければ、承認が必要です。複数の契約が適用されるオーダー・アイテムのオーダーをバイヤーが発行する場合、オーダー内の 1 つのアイテムにこの条件の指定された契約が適用されるのであれば、オーダー全体に対してそのアイテムに適用されるオーダー承認条件が適用されます。

支払いメソッド*

このオプションの条件は、契約の下で発行されるオーダーに関して使用可能な支払いメソッドを指定します。支払いメソッドには、クレジット・カードのタイプなどの支払いタイプのように汎用性のあるメソッドを指定することも、支払いに使用するクレジット・カード番号のように特定のメソッドを指定することもできます。契約で支払いメソッド条件を指定しない場合、その契約の下で発行されるオーダーに関してはストアで使用可能なすべてのメソッドによる支払いが可能となります。

価格設定条件

価格設定条件は、契約に基づいて購入できる商品と顧客がその商品に支払う価額を定義します。契約ごとに、最低 1 つの価格設定条件が必要になります。 WebSphere Commerce では、以下の価格設定条件を選択できます。

カスタマイズ済み価格表

この条件は、販売用の商品リストとその価格の両方を、契約内での販売用にカスタマイズすることを指定します。アイテムは、ストア・カタログの特定のセクションに限定されることはなく、ストア・カタログのどこにあるものでも可能です。

価格調整付き総合カタログ

この条件は、販売用ストア・カタログの購入可能な商品すべてに、ストア・カタログで定義された基本価格からのパーセント調整 (利掛けまたは割引) を提供します。価格調整が指定されていないなら、アイテムは基本価格で販売されます。

価格調整付き価格表*

この条件は、販売用価格表の購入可能な商品すべてに、ストア・カタログで定義された基本価格からのパーセント調整 (利掛けまたは割引) を提供します。価格調整が指定されていないなら、アイテムは基本価格で販売されます。

選択的価格調整付き価格表*

この条件は、価格調整付き価格表と類似していますが、価格調整が価格表全体に適用されない点が異なります。価格調整は、価格表のサブセットに対してなされます。価格表のそのサブセットは、商品

セットビジネス・ポリシーか、またはカスタマイズ済み商品セットのいずれかです。商品セットのタイプ相互間の違いについては、WebSphere Commerce オンライン・ヘルプを参照してください。

フィルター付きカタログ*

この条件は、販売用ストア・カタログの購入可能な商品すべてに、ストア・カタログで定義された基本価格からのパーセント調整 (利掛けまたは割引) を提供します。この条件は、販売用カテゴリー (または特定の商品およびアイテムのリスト) の購入可能な商品すべてに、この条件により参照される価格表で定義された基本価格からのパーセント調整 (利掛けまたは割引) も提供します。さらにこの条件で、契約の中でどのカテゴリー、商品、およびアイテムが販売用でどれがそうでないかも指定できます。カテゴリー商品セットは、商品セットのビジネス・ポリシーとして振る舞います。アイテム商品セットは、カスタマイズされた商品セットとなります。

商品制約条件

商品制約条件は、契約下での販売の対象商品および対象外商品はどれかを制御します。商品制約条件は、オプションです。契約で指定された商品制約の条件がない場合、契約の価格条件に指定されたすべての商品は、その契約下で販売可能です。 WebSphere Commerce では、以下の商品制約条件を選択できます。

カスタマイズ済み商品セットの除外

この条件は、カスタマイズ済み商品セット内のアイテムは、契約下で販売されないことを示します。

カスタマイズ済み商品セットの取り込み

この条件は、カスタマイズ済み商品セット内のアイテムは、契約下で販売されることを示します。

商品セットの除外*

この条件は、商品セット・ビジネス・ポリシー内のアイテムは、契約下で販売されないことを示します。

商品セットの取り込み*

この条件は、商品セット・ビジネス・ポリシー内のアイテムは、契約下で販売されることを示します。

除外条件は取り込み条件に優先します。つまり、ある商品が契約の下で取り込み条件および除外条件の両方の適用を受けている場合、その商品はその契約下では購入できません。カスタマイズ済み商品セットと商品セット・ビジネス・ポリシーとの違いについては、 WebSphere Commerce オンライン・ヘルプを参照してください。

返品条件

返品条件は、この契約下で返品が処理される方法を指定します。返品条件を指定しない場合、返品を作成することはできません。返品条件を指定する場合、それは契約全体に適用される 1 セットだけとします。 WebSphere Commerce では、以下の返品条件を選択できます。

リファンド支払いメソッド*

この条件は、顧客にリファンドを支払う際の支払いメソッドを指定

します。返品課金条件を指定した場合、1 つ以上のリファンド支払いメソッドも同じく指定する必要があります。契約下で返品が許可されていない場合、この条件は指定できません。

返品課金*

この条件は、返品が自動的に承認される方法、および再仕入れ料など返品処理のためにリファンドから差し引かれる金額を指定します。

購入限度額

この条件は、契約下で発行されるすべてのオーダーの税と配送量を含む合計価格を制限します。契約下で発行されるすべてのオーダーの価格は、指定の金額以下でなければなりません。オーダーが発行される際にこの限度を超えていると、そのオーダーに対する支払い承認は失格します。

配送条件

配送条件は、オーダーの配送方法、配送先、および配送料の支払い者を指定します。 WebSphere Commerce では、以下の配送条件を選択できます。

配送モード*

このオプションの条件は、契約下で作成されたオーダーを配送する方法を定義します。この条件が契約で指定されていない場合、オーダーはストアで使用可能な任意のモードで配送されます。配送モードは配送業者としても知られています。配送業者は、運送会社と配送サービスとの組み合わせです。たとえば、XYZ Courier や Overnight は配送業者です。

配送先住所

このオプションの条件は、契約下で購入された商品を配送する方法を定義します。この条件を指定すると、オーダーを配送可能な地域を制限できます。配送先住所条件を指定しないと、契約下でオーダーを作成するたびに配送先住所を指定しなければなりません。この条件が指定されている場合、バイヤーはオーダーを発行するときに新規の配送先を指定するのではなく、配送先住所のリストから配送先住所を選択する必要があります。

配送料タイプ*

この条件は、オーダーの配送料を誰が支払うかを定義します。以下のタイプの配送料がサポートされています。

- 配送料はバイヤーからセラーに支払われます。セラーはオーダーを受けると配送料を計算して、配送料金をオーダーの合計価格の一部とします。
- 配送料はバイヤーから運送会社に支払われます。運送会社は配送料金を計算して、バイヤーから料金を回収する役割を引き受けます。オーダーの受け取り時には、配送料金は計算されません。

参照インターフェース*

この条件は、ストアとリモート・ストアとの関係を指定します。これはリモート・ストアによってサポートされる機能、およびリモート・ストアに送るメッセージ内で使用されるパラメーターを定義します。

ビジネス・ポリシー

ビジネス・ポリシーとは、ストアまたはストアのグループが従う一連のルールのことです。ビジネス・ポリシーは、ビジネス・プロセス、経営施策、およびストアまたはストアのグループのオファリングの有効範囲と特性を定義します。これは、ストアまたはストアのグループ内で許容され、サポートされる施策の中心的なソースおよび参照テンプレートです。

WebSphere Commerce では、ビジネス・ポリシーの規則をインプリメントする 1 つ以上のビジネス・ポリシー・コマンド、規則が適用されるビジネス・オブジェクトへの参照、およびビジネス・ポリシー・コマンドの操作を構成する一連のプロパティーの組み合わせでビジネス・ポリシーは施行されます。契約条件は、自分が参照するビジネス・ポリシーに対してパラメーターを提供する場合があります。これにより、ビジネス・ポリシーの振る舞いを、ビジネス・ポリシーを参照する契約条件に応じて変更することができます。

Business ビジネス・ポリシーは、共用可能なリソースです。契約で使用可能なビジネス・ポリシーをリストするとき、リストするビジネス・ポリシーは、契約が作成されるストアによって所有されるビジネス・ポリシー、および com.ibm.commerce.businessPolicy ストア関係のあるストアによって所有されるビジネス・ポリシーとします。サイト内の複数のストア間で資産を共用する方法の詳細については、 151 ページの『第 14 章 ストア間の関係』を参照してください。

ビジネス・ポリシーの以下のカテゴリーが、WebSphere Commerce には備えられています。

カタログ・ビジネス・ポリシー
カタログ・ビジネス・ポリシーは、ストアのカタログにある商品の価格およびカテゴリーを含む、ストアで販売する商品のカタログの範囲と特性を定義します。

支払いビジネス・ポリシー
送り状発送、支払い、およびリファンドのビジネス・ポリシーは、ストアが支払いの受け取りおよびリファンドの支払いを行う方法、およびストアの送り状の書式を定義します。

返品ビジネス・ポリシー
返品ビジネス・ポリシーは、リファンドが受け入れられるかどうか、受け入れられる期間、および返品に適用される在庫補充料を定義します。

配送ビジネス・ポリシー
配送ビジネス・ポリシーは、ストアが使用可能な配送業者、および各タイプに関連した課金を定義します。

参照インターフェース・ビジネス・ポリシー
参照インターフェース・ビジネス・ポリシーは、プロキシー・ストアとリモート・ストアとの関係を定義します。

多くの契約条件は、ビジネス・ポリシーを参照します。これにより、契約条件を作成する際の柔軟性を備えながら、ストアが締結した契約の性質を制御する手段が備えられます。ビジネス・ポリシーの詳細については、「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」を参照してください。

添付

添付は、取引条件の他のエレメントが扱っていない、取引条件に関する付加的な情報を提供します。一例としては、RFQ の要件に関する付加的な情報と、 Business RFQ に関するあらゆる一般的な所見とを記したファイルが挙げられます。 1 つの取引条件に複数の添付がある場合もあります。添付は WebSphere Commerce の外側に保管され、取引条件は添付の Universal Resource Identifiers (URI) を保管します。以下に URI の例を挙げます。

- http://www.mycompany.com/information/document1.txt
- file:///home/joeuser/mydocs/document1
- ftp://ftp.mycompany.com/information/attachment.txt

すべての添付には、添付の目的を示す、添付使用法を割り当てることができます。添付使用法は、添付のオプションのプロパティーです。

オーダー・アイテム

オーダー・アイテムとは、オーダーに含まれる商品またはアイテムのことです。 1 つのオーダーに含まれる各オーダー・アイテムを、それぞれ異なる取引条件の下で購入することができます。バイヤーは、ストアの設計に応じて、ショッピング・フローを開始するときか、またはアイテムをオーダーに追加するときのいずれかの時点で、ショッピングを行う際に使用する契約取引条件を選択できます。異なる契約取引条件の下でアイテムを購入する場合、以下の規則が適用されます。

- オーダー内のすべてのアイテムの契約取引条件は、少なくとも 1 つの支払い方式を共有していなければなりません。アイテムの契約が支払い方式を共有していないと、バイヤーはそのアイテムをオーダーに追加できません。オーダーの支払いでは、オーダー内のすべてのアイテムで共有される支払い方式だけが使用できます。
- オーダー内のすべてのアイテムは、同じビジネス・アカウントまたはストア・デフォルト契約に属する契約取引条件からのものでなければなりません。

WebSphere Commerce の契約資産の構造の詳細については、 WebSphere Commerce オンライン・ヘルプの『契約データ・モデル』を参照してください。

WebSphere Commerce でのデフォルト契約資産の作成

デフォルト契約は、ストアのデフォルトの振る舞いを定義します。すべての契約と同様に、購入可能な商品、価格、支払い方式、配送方法、および他のストアの振る舞いを設定することができます。

WebSphere Commerce サンプル・ストアに付属しているストアのデフォルト契約には、以下の事柄を指定する契約条件が含まれています。

- 顧客はストアのマスター・カタログにある購入可のすべての商品を、マスター・カタログの標準価格設定で購入できる (割引や値上げはなし)。
- 配送料がセラー (ストア) に支払われる。
- 顧客は一定の日数以内であれば手数料なしで、購入したものを返品できる。

- 顧客は元の購入で使用したのと同じ支払い方式で、リファンドを受け取ることができる。

また、ストアのデフォルト契約の最も一般的なバージョンでは、バイヤーが使用できる支払い方式と配送方法を制限する契約条件が省略されています。これらの条件が省略されているため、バイヤーは、購入代金の支払いに、ストアがサポートするデフォルトの支払い方式をどれでも使用でき、ストアで用意している配送方法をどれでも使用できます。

デフォルト契約のプロパティーは、その契約の条件の中で定義されます。一部の契約条件は、ビジネス・ポリシーを参照します。ビジネス・ポリシーと契約条件の詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

デフォルト契約資産を作成するには、以下のようにします。

1. 契約条件、契約、デフォルト契約、およびビジネス・ポリシーに関する、オンライン情報を確認します。
2. wcs.bootstrap.xml ファイルで定義されるビジネス・ポリシーを確認します。wcs.bootstrap.xml ファイルの詳細については、オンライン情報を参照してください。
3. サンプル・ストアのデフォルト契約資産を作成するために使用されるファイルを確認します。サンプル・ストアのファイルはすべて、対応するアーカイブ・ファイルの中にあります。各サンプル・ストアには、businesspolicy.xml および contract.xml が含まれ、これらには追加のビジネス・ポリシー情報とデフォルト契約情報が含まれます。ストア・アーカイブ・ファイルは、*WC_installdir*/samplestores ディレクトリーにあります。

   注:

   a. WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

   b. ストア・アーカイブの businesspolicy.xml および contract.xml ファイルを表示するには、 ZIP プログラムを使用してこれを解凍します。ファイルは、data ディレクトリーにあります。

   c. WebSphere Commerce Business Edition に付属している ToolTech サンプル・ストアの契約資産ファイルには、ストアのデフォルト契約以外の契約の情報も含まれています。
4. 489 ページの『付録 B. データの作成』に記載されている情報を確認します。
5. サンプル・ストア・アーカイブの businesspolicy.xml ファイルの 1 つをコピーするか、新しいファイルを作成することにより、businesspolicy.xml ファイルを作成します。新しいファイルの作成方法の説明は、215 ページの『ビジネス・ポリシー XML ファイルの作成』にあります。説明されているビジネス・ポリシーと異なるビジネス・ポリシーを作成する場合は、 businesspolicy.xml に対応する DTD ファイルを参照してください。 DTD ファイルは、*WC_installdir*/xml/sar ディレクトリーにあります。
6. ローダー・パッケージを使用して businesspolicy.xml ファイルをロードします。ローダー・パッケージの詳細については、 363 ページの『第 10 部 ストアの発行』を参照してください。多文化ストアを作成する場合は、ストアでサポートさ

れているロケールごとに別々の XML ファイルを作成することもできます。説明情報はすべてロケール固有のファイルに指定されるので、これを翻訳するのは容易です。

7. サンプル・ストア・アーカイブの contract.xml ファイルの 1 つをコピーするか、新しいファイルを作成することにより、contract.xml ファイルを作成します。新しいファイルの作成方法の説明は、216 ページの『デフォルト契約ファイルの作成』にあります。より複雑なデフォルト契約を作成する場合は、契約ファイルの構造を定義する B2BTrading.dtd または Package.xsd ファイルを確認してください。 B2BTrading.dtd ファイルは、 *WC_installdir*/xml/trading/dtd ディレクトリーにあります。 Package.xsd ファイルは、 *WC_installdir*/xml/trading/xsd ディレクトリーにあります。
8. ContractImportApprovedVersion コマンドを使用して契約を発行します。詳細については、443 ページの『第 39 章 ビジネス・アカウントと契約の発行』を参照してください。 ContractImportApprovedVersion コマンドについての情報は、WebSphere Commerce オンライン・ヘルプの中にもあります。

WebSphere Commerce Business Edition ユーザーは、WebSphere Commerce アクセラレーターを使用して特定の顧客用の契約を定義できます。特定の顧客用の契約の作成の詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

ビジネス・ポリシー XML ファイルの作成

ストアのデフォルト契約の条件が参照できるよう、WebSphere Commerce には多数のビジネス・ポリシーが備わっていますが、自分で定義しなければならないビジネス・ポリシーもあります。ストアのデフォルト条件が参照する、返品の課金、返品の承認、および価格設定に関するビジネス・ポリシーは、自分で定義しなければなりません。これらのビジネス・ポリシー用のコマンドが用意されているので、これらを変更せずに使用できます。独自のビジネス・ポリシーを作成したい場合は、「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」を参照してください。

自分のストア用にビジネス・ポリシーを作成するには、ビジネス・ポリシーを作成して、そのビジネス・ポリシーに 1 つ以上のコマンドを関連付ける必要があります。ビジネス・ポリシーを作成するには、POLICY テーブルに情報を追加します。コマンドをビジネス・ポリシーに関係付けるには、POLICYCMD テーブルに情報を追加します。

ビジネス・ポリシーを作成して、そのポリシーにコマンドを関連付けるには、次のようにします。

1. POLICY テーブルに情報を追加することによって、ビジネス・ポリシー XML ファイルにビジネス・ポリシーを作成します。次の例を参考にしてください。

```
<policy
policy_id="@policy_id_10"
policyname="MasterCatalogPriceList"
policytype_id="Price"
storeent_id="@storeent_id_1"
properties="name=&STORE_IDENTIFIER;&orgentity_dn=ORGANIZATION_DN
/>
```

ここで、

- policy_id は、ビジネス・ポリシーに対する固有な数値 ID です。
- policyname は、このビジネス・ポリシーに対する固有名です。
- policytype_id は、定義するポリシーのタイプです。有効な Valid policytype_ids は以下のとおりです。
  - InvoiceFormat
  - Payment
  - Price
  - ProductSet
  - ReturnApproval
  - ReturnCharge
  - ReturnPayment
  - ShippingCharge
  - ShippingPayment
  - ReferralInterface
- storeent_id は、ストアまたはストア・グループです。
- properties は、ビジネス・ポリシー・コマンドに送信される名前と値の対のリストです。

2. POLICYCMD テーブルに情報を追加することによって、ビジネス・ポリシー XML ファイルでコマンドをビジネス・ポリシーに関連付けます。次の例を参考にしてください。

```
<policycmd
policy_id="@policy_id_10"
businesscmdclass=
  "com.ibm.com.commerce.price.commands.RetrievePricesCmdImpl"
/>
```

ここで、

- policy_id は、コマンドに関連付けられているビジネス・ポリシーの数値 ID です。
- businesscmdclass は、ビジネス・ポリシーをインプリメントしている Java クラスの名前です。

businesscmdclass 属性が改行されているのは、表示上の理由に過ぎません。

@ と **&** の使用法の詳細については、 489 ページの『付録 B. データの作成』を参照してください。

デフォルト契約ファイルの作成

デフォルト契約を作成するには、契約、契約所有者、契約の説明、契約の参加者、および契約の条件を定義することが必要です。契約情報は、CONTRACT、PARTICIPNT、TRADING、および TERMCOND の 4 つのテーブルに保管されます。

デフォルト契約は、STOREDEF データベース・テーブルを使用してストアと関連付けられます。 WebSphere Commerce Business Edition ユーザーの場合、デフォルト契約以外の契約は STORECNTR データベース・テーブルを使用してストアと関連付けられます。

XSD または DTD の 2 つの形式のいずれかに基づいて、XML でデフォルト契約を作成することができます。各タイプを作成するための詳しい方法については、以下のセクションを参照してください。

XSD でのデフォルト契約ファイルの作成

XSD 形式でのデフォルト契約を作成するには、以下のようにします。

1. XML ファイルにデフォルト契約を定義します。デフォルト契約は、次のように XML ファイルの先頭で定義されます。

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <Package xmlns="http://www.ibm.com/WebSphereCommerce"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.ibm.com/WebSphereCommerce Package.xsd">
   <BuyerContract state="Active" contractUsage="Default" comment="">
   <ContractUniqueKey majorVersionNumber="1" minorVersionNumber="0"
    name="&STORE_IDENTIFIER; Default Contract" origin="Manual">
   <ContractOwner>
   <OrganizationRef distinguishName="ou=&ORGENTITYNAME;,&ORGANIZATION_DN;" />
   </ContractOwner>
   </ContractUniqueKey>
   ```

   Package および ContractUniqueKey エレメントが改行されているのは、表示上の理由に過ぎません。

2. 契約 XML ファイルに契約の参加者を定義します。次の例を参考にしてください。

   ```
   <Participant role="Buyer">
   </Participant>
   <Participant role="Seller">
     <ParticipantMember>
       <OrganizationRef distinguishName=""ou=&ORGENTITYNAME;,&ORGANIZATION_DN;"/>
     </ParticipantMember>
   </Participant>
   ```

   distinguishName は、この契約のセラーであるユーザーの LDAP 識別名形式の名前で、たとえば、uid=johnsmith,ou=People,o=ibm,o=com などです。多くの場合、これは契約の所有者と同じになります。

   注: バイヤー参加者役割にメンバーは指定されません。契約はバイヤー役割を持つすべてのユーザーに使用可能であるためです。

3. 契約 XML ファイルに契約の説明を定義します。次の例を参考にしてください。

   ```
   <ContractDescription title="This is a store default contract." locale="en_US"/>
   ```

   ここで、

   - title は、契約のテキスト記述です。
   - locale は、タイトルを記述する言語のロケールです。 locale には、以下の値が事前定義されています。
     - en_US (英語)
     - fr_FR (フランス語)

- de_DE (ドイツ語)
- it_IT (イタリア語)
- es_ES (スペイン語)
- pt_BR (ブラジル・ポルトガル語)
- zh_CN (中国語 (簡体字))
- zh_TW (中国語 (繁体字))
- ko_KR (韓国語)
- ja_JP (日本語)

ご使用のサイトの言語資産を更新することによって、`locale` に追加の値を定義することもできます。言語資産の詳細については、245 ページの『第 22 章 言語資産』を参照してください。

4. 契約 XML ファイルに契約条件を定義します。 XML エレメントと属性は、契約条件のタイプによって異なります。各タイプの条件に使用する XML エレメントと属性について学ぶには、`Package.xsd` ファイルを使用します。契約条件を定義する際は、一般に以下の属性が使用されます。

policyName
契約条件が参照するビジネス・ポリシーの名前。この名前は POLICY.POLICYNAME に保管されます。

ポリシー参照
契約条件が参照するビジネス・ポリシーのタイプ。有効な値は、以下のとおりです。

- PricePolicyRef
- ProductSetPolicyRef
- InvoiceFormatPolicyRef
- PaymentPolicyRef
- ReturnApprovalPolicyRef
- ReturnChargePolicyRef
- ReturnPaymentPolicyRef
- ShippingChargePolicyRef
- ShippingModePolicyRef

storeRef
契約条件のストアまたはストア・グループ。

distinguishName
ストアまたはストア・グループを所有しているユーザーの名前。名前は LDAP 識別名形式でなければなりません。たとえば、`uid=wcsadmin,o=Root Organization` などです。

以下に、サンプルの契約条件と、この条件が定義する内容の説明とを列挙します。

- すべてのバイヤーは、ストアのマスター・カタログのすべてのアイテムを、マスター・カタログに設定された価格で購入できます。

```
<PriceTCMasterCatalogWithOptionalAdjustment>
</PriceTCMasterCatalogWithOptionalAdjustment>
```

- バイヤーは配送料をセラーに支払います。

```
<ShippingTCShippingCharge>
<ShippingChargePolicyRef policyName="StandardShippingChargeBySeller">
  <StoreRef name="&STORE_IDENTIFIER;">
    <Owner>
      <OrganizationRef distinguishName="ou=&ORGENTITYNAME;,
       &ORGANIZATION_DN;" />
    </Owner>
  </StoreRef>
</ShippingChargePolicyRef>
  </ShippingTCShippingCharge>
```

- バイヤーは、返品の課金なしで商品を返品することができます。商品は、ApprovalByDays ビジネス・ポリシーで定義された日数以内に返品しなければなりません。

```
 <ReturnTCReturnCharge>
<ReturnChargePolicyRef policyName="NoCharges">
   <StoreRef name="&STORE_IDENTIFIER;">
     <Owner>
       <OrganizationRef distinguishName="ou=&ORGENTITYNAME;,
         &ORGANIZATION_DN;" />
     </Owner>
   </StoreRef>
</ReturnChargePolicyRef>
<ReturnApprovalPolicyRef policyName="ApprovalByDays">
   <StoreRef name="&STORE_IDENTIFIER;">
     <Owner>
       <OrganizationRef distinguishName="ou=&ORGENTITYNAME;,
         &ORGANIZATION_DN;" />
     </Owner>
   </StoreRef>
 </ReturnApprovalPolicyRef>
 </ReturnTCReturnCharge>
```

WebSphere Commerce Business Edition ユーザーへの注意:
ストアのデフォルト契約からこれらの契約条件が省略してあるということは、デフォルトでは、ストアが返品を受け付けないことを示します。ただし、他の契約では、返品の条件を定義することにより、バイヤーが返品を行えるようにすることができます。

WebSphere Commerce Professional Edition ユーザーへの注意:
ストアのデフォルト契約からこれらの契約条件が省略してあるということは、ストアが返品を受け付けないことを示します。

- リファンドは、オーダーの完了時にバイヤーが使用したのと同じ支払い方式で支払われます。

```
 <ReturnTCRefundPaymentMethod>
<ReturnPaymentPolicyRef policyName="UseOriginalPayment">
<StoreRef name="&STORE_IDENTIFIER;">
     <Owner>
       <OrganizationRef distinguishName="ou=&ORGENTITYNAME;,
        &ORGANIZATION_DN;" />
     </Owner>
   </StoreRef>
 </ReturnPaymentPolicyRef>
 </ReturnTCRefundPaymentMethod>
```

5. 以下の方法で XML ファイルを閉じます。

```
</BuyerContract>
</Package>
```

@ と **&** の使用法の詳細については、 489 ページの『付録 B. データの作成』を参照してください。

DTD 形式でのデフォルト契約ファイルの作成

デフォルト契約を作成するには、契約、契約所有者、契約の説明、契約の参加者、および契約の条件を定義することが必要です。契約情報は、CONTRACT、PARTICIPNT、TRADING、および TERMCOND の 4 つのテーブルに保管されます。

デフォルト契約は、STOREDEF データベース・テーブルを使用してストアと関連付けられます。 WebSphere Commerce Business Edition ユーザーの場合、デフォルト契約以外の契約は STORECNTR データベース・テーブルを使用してストアと関連付けられます。

DTD 形式でのデフォルト契約を作成するには、以下のようにします。

1. XML ファイルにデフォルト契約を定義します。デフォルト契約は、次のように XML ファイルの先頭で定義されます。

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <!DOCTYPE Trading SYSTEM "B2BTrading.dtd">
   <Trading>
   <Contract state="Active" origin="Manual"
     name="&STORE_IDENTIFIER; Default Contract" majorVersionNumber="1"
     minorVersionNumber="0" contractUsage="Default">
   ```

   Contract エレメントが改行されているのは、表示上の理由に過ぎません。
2. 契約所有者を定義します。次の例を参考にしてください。

   ```
   <ContractOwner>
     <Member>
       <Organization distinguishName="ou=&ORGENTITYNAME;,&ORGANIZATION_DN;" />
     </Member>
   </ContractOwner>
   ```

   distinguishName は、契約を所有しているユーザーの LDAP 識別名形式の名前です。たとえば、uid=johnsmith,ou=People,o=ibm,o=com などです。
3. 契約 XML ファイルに契約の説明を定義します。次の例を参考にしてください。

   ```
   <ContractDescription title="This is a store default contract." languageId="-1">
   </ContractDescription>
   ```

   ここで、

   - title は、契約のテキスト記述です。
   - languageId はタイトルの言語です。 languageId には、以下の値が事前定義されています。
     - -1 (英語)
     - -2 (フランス語)
     - -3 (ドイツ語)
     - -4 (イタリア語)
     - -5 (スペイン語)
     - -6 (ブラジル・ポルトガル語)

– -7 (中国語 (簡体字))
– -8 (中国語 (繁体字))
– -9 (韓国語)
– -10 (日本語)

ご使用のサイトの言語資産を更新することによって、languageId に追加の値を定義することもできます。言語資産の詳細については、 245 ページの『第 22 章 言語資産』を参照してください。

4. 契約 XML ファイルに契約の参加者を定義します。次の例を参考にしてください。

```
<Participant role="Buyer">
</Participant>
<Participant role="Seller">
  <Member>
    <Organization distinguishName="ou=&ORGENTITYNAME;,&ORGANIZATION_DN;"/>
  </Member>
</Participant>
```

distinguishName は、この契約のセラーであるユーザーの LDAP 識別名形式の名前で、たとえば、uid=erickoeck,ou=People,o=ibm,o=com などです。多くの場合、これは契約の所有者と同じになります。

注: バイヤー参加者役割にメンバーは指定されません。契約はバイヤー役割を持つすべてのユーザーに使用可能であるためです。

5. 契約 XML ファイルに契約条件を定義します。 XML エレメントと属性は、契約条件のタイプによって異なります。各タイプの条件に使用する XML 要素と属性について学ぶには、B2BTrading.dtd ファイルを使用します。契約条件を定義する際は、一般に以下の属性が使用されます。

policyName
契約条件が参照するビジネス・ポリシーの名前。この名前は POLICY.POLICYNAME に保管されます。

policyType
契約条件が参照するビジネス・ポリシーのタイプ。有効な値は、以下のとおりです。

- Price
- ProductSet
- InvoiceFormat
- Payment
- ReturnApproval
- ReturnCharge
- ReturnPayment
- ShippingCharge
- ShippingMode

storeIdentity
契約条件のストアまたはストア・グループ。

distinguishName
ストアまたはストア・グループを所有しているユーザーの名前。名前は LDAP 識別名形式でなければなりません。たとえば、`uid=wcsadmin,o=Root Organization` などです。

以下に、サンプルの契約条件と、この条件が定義する内容の説明とを列挙します。

- すべてのバイヤーは、ストアのマスター・カタログのすべてのアイテムを、マスター・カタログに設定された価格で購入できます。

```
<TermCondition>
  <PriceTC>
    <PriceTCMasterCatalogWithOptionalAdjustment>
    </PriceTCMasterCatalogWithOptionalAdjustment>
  </PriceTC>
</TermCondition>
```

- バイヤーは配送料をセラーに支払います。

```
<TermCondition>
  <ShippingTC>
    <ShippingTCShippingCharge>
      <PolicyReference policyName="StandardShippingChargeBySeller"
        policyType="ShippingCharge" storeIdentity="&STORE_IDENTIFIER;">
        <Member>
          <Organization distinguishName="ou=&ORGENTITYNAME;,
           &ORGANIZATION_DN;" />
        </Member>
      </PolicyReference>
    </ShippingTCShippingCharge>
  </ShippingTC>
</TermCondition>
```

`PolicyReference` エレメントが改行されているのは、表示上の理由に過ぎません。

- バイヤーは、返品の課金なしで商品を返品することができます。商品は、`ApprovalByDays` ビジネス・ポリシーで定義された日数以内に返品しなければなりません。

```
<TermCondition>
  <ReturnTC>
    <ReturnTCReturnCharge>
      <ReturnChargePolicyReference>
        <PolicyReference policyName="NoCharges"
         policyType="ReturnCharge"
         storeIdentity="&STORE_IDENTIFIER;">
          <Member>
            <Organization distinguishName="ou=&ORGENTITYNAME;,
            &ORGANIZATION_DN;" />
          </Member>
        </PolicyReference>
      </ReturnChargePolicyReference>
      <ReturnApprovalPolicyReference>
        <PolicyReference policyName="ApprovalByDays"
         policyType="ReturnApproval"
         storeIdentity="&STORE_IDENTIFIER;">
          <Member>
            <Organization distinguishName="ou=&ORGENTITYNAME;,
            &ORGANIZATION_DN;" />
          </Member>
        </PolicyReference>
```

```
      </ReturnApprovalPolicyReference>
    </ReturnTCReturnCharge>
  </ReturnTC>
</TermCondition>
```

PolicyReference エレメントが改行されているのは、表示上の理由に過ぎません。

WebSphere Commerce Business Edition ユーザーへの注意:
ストアのデフォルト契約からこれらの契約条件が省略してあるということは、デフォルトでは、ストアが返品を受け付けないことを示します。ただし、他の契約では、返品の条件を定義することにより、バイヤーが返品を行えるようにすることができます。

WebSphere Commerce Professional Edition ユーザーへの注意:
ストアのデフォルト契約からこれらの契約条件が省略してあるということは、ストアが返品を受け付けないことを示します。

- リファンドは、オーダーの完了時にバイヤーが使用したのと同じ支払い方式で支払われます。

```
<TermCondition>
  <ReturnTC>
    <ReturnTCRefundPaymentMethod>
      <PolicyReference policyName="UseOriginalPayment"
       policyType="ReturnPayment" storeIdentity="&STORE_IDENTIFIER;">
        <Member>
          <Organization distinguishName="ou=&ORGENTITYNAME;,
           &ORGANIZATION_DN;" />
        </Member>
      </PolicyReference>
    </ReturnTCRefundPaymentMethod>
  </ReturnTC>
</TermCondition>
```

PolicyReference エレメントが改行されているのは、表示上の理由に過ぎません。

6. 以下の方法で XML ファイルを閉じます。

```
</Contract>
</Trading>
```

@ と **&** の使用法の詳細については、 489 ページの『付録 B. データの作成』を参照してください。

第 19 章 フルフィルメント資産

配送センターは、ストアにより、在庫保管庫、および配送受取センターの両方として使用されます。配送センターは、顧客への商品の配送元になる場所を表します。在庫数は、配送センターごとに別個に保守されます。 1 つのストアに対して、1 つ以上の配送センターが関連していることがあります。配送センターは、ストアの商品の在庫と配送を管理します。フルフィルメントには、ピッキング、パッキング、配送が含まれます。ピッキングとは、配送センターからの 1 つ以上のリリースの中から商品を選択することで、パッキングとはそれらの商品を配送コンテナーに入れることで、配送とはそれらを顧客に送ることです。

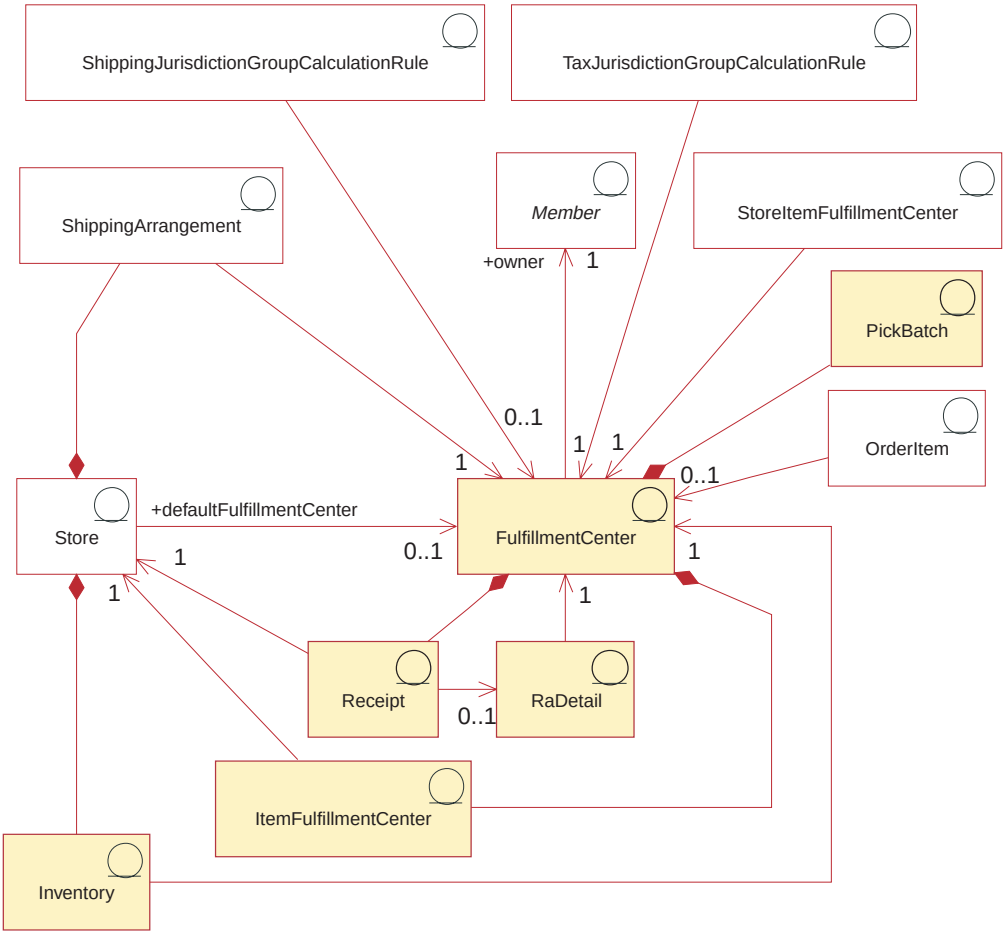
商品のフルフィルメントは、「商品」ウィザードと「商品」ノートブックで構成します。商品の構成には、在庫を追跡記録するオプション、バックオーダーを許可するオプション、バックオーダーを強制するオプション、商品を別々にリリースするオプション、商品を返品不可にするオプションが用意されています。

一般に、配送センターには作業する人が同時に何人かいて、それぞれ 1 つ以上の異なるタスクの実行を担当します。 WebSphere Commerce アクセラレーターは、最も一般的なタスクを役割に分け、これらの役割がユーザーに割り当てられます。WebSphere Commerce アクセラレーターで、フルフィルメントに関する 1 つ以上の役割を割り当てた場合, ログオン時に 1 つの配送センターを選択する必要があります。

注: フルフィルメント、配送センター、および役割の詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

WebSphere Commerce のフルフィルメント資産について

フルフィルメント資産を理解するには、フルフィルメントとストアの間の関係を理解することが必要です。これは、情報モデルを使用して説明できます。次に、ストアおよび他の資産とフルフィルメントとの関係を説明します。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、111 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、487 ページの『付録 A. UML の凡例』を参照してください。

配送センター

上の図では、*配送センター* がフルフィルメントのプロセスの中心にあります。配送センターには、MEMBER テーブルで定義された所有者がいます。各ストアが複数の配送センターに関連していること、および 1 つの配送センターが複数のストアに関連していることが可能です。図に示されているように、ストアと配送センターの間の相互作用がいくつかあります。ストア資産の詳細については、143 ページの『WebSphere Commerce のストア資産について』を参照してください。

受け取り

配送センターは、アイテムの在庫を毎日、毎週、または毎月受け取ります。アイテムの在庫を受け取ると、RECEIPT テーブルに*受け取り* が作成されます。このテーブルは、受け取った数量と、在庫を所有しているストアの情報を記録します。オーダーが処理されると、現在の使用可能な在庫レベルを反映するよう RECEIPT テーブルも更新されます。受け取りの作成の詳細については、 309 ページの『WebSphere Commerce での在庫資産の作成』を参照してください。

RaDetail

RaDetail は、予測在庫の記録におけるアイテムについての詳細情報です。この情報を使用して、いつ配送センターに在庫が到着するかを見積もったり、バックオーダーされたアイテムの配送予定日を顧客に知らせたりすることができます。

在庫

ストアには、配送センターと関連した*在庫* があります。在庫には、物理的に報告することのできる、配送センター内にあるすべてのものが含まれます。在庫には 1 つのストアと 1 つの配送センターが関連付けられます。ストアが配送センターに所有している在庫についての情報も記録されます。たとえば予約済みの数量、バックオーダー中の金額、およびバックオーダーに割り当てられる金額などです。この情報は、ITEMFFMCTR テーブルに保管されます。在庫および在庫資産の詳細については、 305 ページの『第 29 章 在庫資産』を参照してください。

配送調整

配送調整 は、ストアが配送センターを使用できるようにする関係です。配送調整は、配送センターが、配送方式を使用して、ストアの代わりに商品を配送できることを示します。各ストアは配送センターと配送調整を行い、各配送センターはストアと配送調整を行います。配送調整は、SHPARRANGE テーブルでセットアップされます。配送調整の作成の詳細については、 275 ページの『配送フルフィルメント資産の作成』を参照してください。

他のフルフィルメント資産

配送センターとの関係の中には、ストアに直接関係しないものもあります。ピッキング・バッチは、1 つの配送センターと関連する関係です。ピッキング・バッチは、処理するオーダー・リリースを配送センターで 1 つの単位としてグループ化し、ピッキング・スリップとパック・スリップを作成します。アイテムがピッキングされ、パックされると、オーダー・リリースを配送できるようになり、配送を確認することができます。ピッキング・バッチ情報は、PICKBATCH テーブルに保管されます。オーダー・アイテムも、1 つの配送センターと関連する関係です。アイテムは、属性によって定義される商品の特定インスタンスです。オーダー内の各アイテムに関する情報は、ORDERITEMS テーブルに保管されています。オーダー資産の詳細については、 315 ページの『第 30 章 オーダー資産』を参照してください。

他のエンティティーと同様、配送センターにはそのアクションのいくつかを制御する規則があります。各配送センターには、税および配送料に関する規則があります。これらの規則は、それぞれ TAXJCRULE および SHPJCRULE テーブルで定義

されます。税および配送資産の詳細については、 263 ページの『第 26 章 配送資産』、および 281 ページの『WebSphere Commerce の税資産について』を参照してください。

WebSphere Commerce Server の配送資産の構造の詳細については、WebSphere Commerce オンライン・ヘルプの『配送データ・モデル』を参照してください。

WebSphere Commerce での配送資産の作成

品物を提供する 1 つ以上の配送センターを定義しておかないと、ストアは顧客に品物を配送できません。この情報を XML ファイルの形式で作成します。XML ファイルは、ローダー・パッケージを使用して、データベースにロードできます。ローダー・パッケージの詳細については、 363 ページの『第 10 部 ストアの発行』を参照してください。

資産を作成する前に、429 ページの『第 38 章 WebSphere Commerce データベース資産グループのロード』で扱われている資料に精通している必要もあります。

XML ファイルを使用してストアのフルフィルメント資産を作成するには、以下のようにします。

1. サンプル・ストアのフルフィルメント資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。

   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

   - *WC_installdir*/samplestores

     注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

   個々のサンプル・ストアには 1 つの fulfillment.xml ファイルが組み込まれており、このファイルにはフルフィルメント情報が組み込まれています。ストア・アーカイブの fulfillment.xml ファイルを表示するには、 ZIP プログラムを使用してこれを解凍します。 fulfillment.xml ファイルはデータ・ディレクトリーにあります。

2. 489 ページの『付録 B. データの作成』に記載されている情報を確認します。
3. サンプル・ストア・アーカイブの fulfillment.xml ファイルの 1 つをコピーするか、新しいファイルを作成することにより、fulfillment.xml ファイルを作成します。詳細については、fulfillment.xml に対応する DTD ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。

   - *WC_installdir*/xml/sar

4. ストアがサポートする 1 つ以上の配送センターを定義します。

   a. 次の例を参考にして、XML ファイルの FFMCENTER テーブルに配送センターを定義します。

```
<ffmcenter
   ffmcenter_id="@ffmcenter_id_1"
   member_id="@seller_b2b_mbr_id"
```

```
name="ToolTech Home"
defaultshipoffset="0"
markfordelete="0"
/>
```

ここで、

- ffmcenter_id は、生成される固有キーです。
- member_id は、配送センターの所有者です。
- name は、所有者と共にこの配送センターを固有に識別するストリングです。
- defaultshipoffset は、この配送センターからアイテムを配送するのにかかる秒数の見積もりです。この値は、STORITMFFC テーブルでオーバーライドできます。
- markfordelete は、配送センターが削除されるべきかどうかを次のように示します。 0 = 削除しない。 1 = 使用されなくなったら削除する。詳細については、WebSphere Commerce オンライン・ヘルプの『データベース・クリーンアップ情報』を参照してください。

b. 次の例を参考にして、XML ファイルの FFMCENTDS テーブルに配送センターを記述します。多文化ストアを作成する場合は、ロケール固有の XML ファイルにこの情報を組み込む必要があります。

```
<ffmcentds
   ffmcenter_id="@ffmcenter_id_1"
   description="The fulfillment center that supplies products to ToolTech."
   language_id="&en_US"
   displayname="ToolTech Fulfillment"
   staddress_id="@staddress_id_en_US_1"
   />
```

ここで、

- ffmcenter_id は、生成される固有キーです。
- description は、顧客に対して表示するのに適した配送センターの説明です。
- language_id は、この情報の表示に使用される言語です。 (さまざまな言語でのサポートの詳細については、 337 ページの『第 34 章 グローバリゼーション』を参照してください。)
- displayname は、顧客に対して表示するのに適した配送センターの名前です。
- staddress_id は、配送センターの物理的場所です。

c. ストアがサポートするすべての配送センターについて、ステップ a と b を繰り返します。

@ と **&** の使用法の詳細については、 489 ページの『付録 B. データの作成』を参照してください。

ストア・フルフィルメント資産の作成 (非 ATP)

ストアに品物を提供する 1 つ以上の配送センターを定義した後、配送センターを、各商品に関連付けることが必要です。つまり、どの配送センターがどの商品を提供するかを識別しなければなりません。この関係を作成するには、INVENTORY テーブルに情報を追加します。この情報を XML ファイルの形式で作成します。XML

ファイルは、ローダー・パッケージを使用して、データベースにロードできます。ローダー・パッケージの詳細については、 363 ページの『第 10 部 ストアの発行』を参照してください。

注:

1. ストアを配送センターに関連付ける前に、ストア資産を作成することが必要です。ストア資産の作成の詳細については、 145 ページの『XML ファイルによるストア・データ資産の作成』を参照してください。また、ストア・フルフィルメント資産を作成する前に、カタログ資産を作成する必要もあります。詳細については、 187 ページの『ストア・カタログ資産の表示』を参照してください。
2. 非 ATP フルフィルメントをインプリメントしている場合にだけ、ストア・フルフィルメント資産を作成してください。 INVENTORY テーブルは、ATP 機能が組み込まれているストアでは使用されません。

XML ファイルを使用してストア・フルフィルメントの関係を作成するには、以下のようにします。

1. サンプル・ストアのストア・フルフィルメント資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。

   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

   - *WC_installdir*/samplestores

     注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

   個々のサンプル・ストアには 1 つの storefulfill.xml ファイルが組み込まれており、このファイルにはストア・フルフィルメント情報が組み込まれています。ストア・アーカイブの storefulfill.xml ファイルを表示するには、 ZIP プログラムを使用してこれを解凍します。 storefulfill.xml ファイルはデータ・ディレクトリーにあります。
2. 489 ページの『付録 B. データの作成』に記載されている情報を確認します。
3. サンプル・ストア・アーカイブの storefulfill.xml ファイルの 1 つをコピーするか、または新しいファイルを作成することにより、 storefulfill.xml ファイルを作成します。詳細については、*WC_installdir*/schema/xml ディレクトリー中の wcs.dtd ファイル、またはストア・アーカイブ中に組み込まれている DTD を参照してください。
4. 次の例を参考にして、XML ファイルの INVENTORY テーブルに情報を追加することにより、ストアと配送センターの関係を作成します。

   ```
   <inventory
   catentry_id="@catentry_id_1470"
   quantity="100"
   ffmcenter_id="@ffmcenter_id_1"
   store_id="@storeent_id_1"
   quantitymeasure="C62"
   inventoryflags="0"
   />
   ```

   ここで、

   - catentry_id は、この配送センターが提供するカタログ・エントリーです。

- quantity は、この配送センターから使用できる数量です (QUANTITYMEASURE に示された単位による)。
- ffmcenter_id は、在庫を提供する配送センターです。
- store_id は、在庫が供給されるストアです。
- quantitymeasure は、QUANTITY の計測単位です。
- inventoryflags は、QUANTITY の使用法を示す以下のビット・フラグです。
  - 1 = noUpdate. デフォルトの UpdateInventory タスク・コマンドは QUANTITY を更新しません。
  - 2 = noCheck. デフォルトの CheckInventory および UpdateInventory タスク・コマンドは QUANTITY をチェックしません。

5. ストアの中の各カタログ・エントリーごとにステップ 3 を繰り返します。

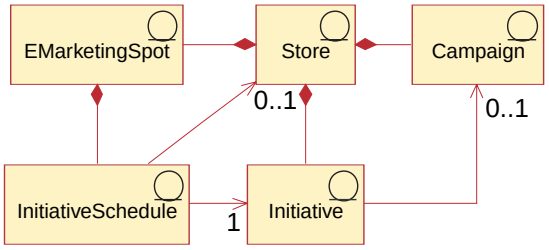
@ と **&** の使用法の詳細については、 489 ページの『付録 B. データの作成』を参照してください。

第 20 章 キャンペーン資産

キャンペーンは、組織的にマーケティング活動を展開するのに役立ちます。多くの場合、キャンペーンは、マーケティング・マネージャーまたは取引管理マネージャーのいずれかによって作成されます。それには、しばしば特定の目標が関係しています。たとえば、「学校へ戻ろう」というキャンペーンであれば、キャンペーン中の子供服の販売を促進するという目標が関係しています。

WebSphere Commerce のキャンペーンについて

以下の図は、WebSphere Commerce Server でのキャンペーン資産を示しています。

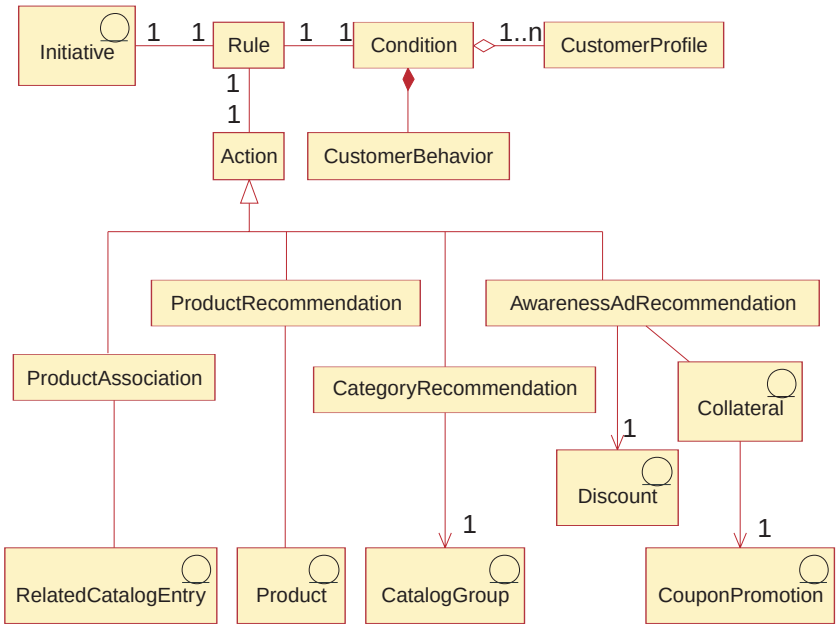


この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です情報モデルの詳細については、111 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、487 ページの『付録 A. UML の凡例』を参照してください。

キャンペーンおよび関連する資産は、ストアを視野に入れています。

WebSphere Commerce の中でのキャンペーンには、任意の数のキャンペーン・イニシアチブが含まれており、それによって 1 つの条件が定義されます。キャンペーン・イニシアチブは、定義された条件が真になった場合に、顧客を対象としたコンテンツを生成します。その結果として、キャンペーンはさまざまなイニシアチブを編成するためのハイレベルなマーケティング・エレメントということになります。

キャンペーン・イニシアチブの情報モデルを以下に示します。



キャンペーン・イニシアチブは、一連のイニシアチブを含むキャンペーンに関連付けられます。その関係を示す例として、オフィス・サプライ品を扱うあるストアで、「学校へ戻ろう」キャンペーンを繰り広げるとします。その場合、イニシアチブは、登録されている顧客のうち職業欄が学生となっている顧客を対象として、ペンの割引を宣伝したりレポート用紙の購入を勧めたりするなどの低レベルのアクションを担当します。

キャンペーン・イニシアチブは、以下の 4 種類の動的コンテンツを表示できます。

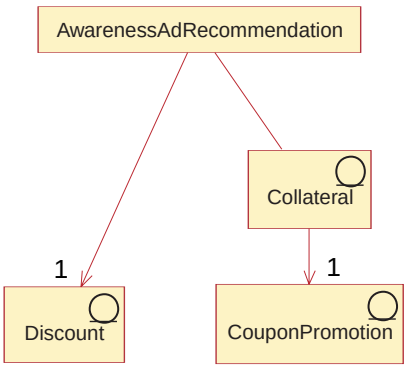
- お勧め商品提示商法
- 協調フィルタリング・ベースの商品提示法
- 顧客キャッチ広告
- 取引管理アソシエーション

お勧め商品提示商法のコンテンツは、特定の顧客を対象として、その顧客のプロファイルやその他の振る舞いに基づいたルール・ベースのカテゴリーおよび商品推奨を提供します。対象にすることのできる顧客の振る舞いには、ショッピング・カートの合計価格、ショッピング・カートの内容、および顧客の購入履歴の内容が含まれます。

協調フィルタリング・ベースの商品提示法も商品推奨を作成しますが、使用する推奨アルゴリズムが少し異なっています。この場合、事前に定義されたルールではなく顧客の振る舞い全体に基づいて対象商品が決められます。

顧客キャッチ広告は、お勧め商品提示商法と同じ基準に基づき、特定の顧客を対象として宣伝コンテンツを提供します。しかし、顧客キャッチ広告は、オンライン・ストアのさまざまなアクティビティーに関する顧客の認知度を高めたり、特別なオファーを強調したり、ブランドの認知度を高めたりすることを意図したものです。

認識広告は、以下に示す情報モデルに従います。



取引管理アソシエーションは、カタログで定義済みの静的アソシエーションをベースにして、上位商品販売および関連商品販売の機会を作ります。このタイプのイニシアチブを作成するためには、アソシエーションでのソース商品の選択方式を定義する必要があります。これによって、e-マーケティング・スポットの呼び出し時に適切なソースが使用され、対象商品を戻します。その方式では、現在のページの内容、ショッピング・カートの内容、またはショッパーの購入履歴のいずれかをベースにして、ソースをアソシエーションのソースとして選択できます。

イニシアチブは、サイトのどのページにも組み込むことができます。サイトの設計時に、 e-マーケティング・スポットと呼ばれる特別なプレースホルダーがサイト上に配置されます。顧客に対して表示される時点でそれらのプレースホルダーは、対象となる特定のコンテンツに置き換えられます。その位置は、 e-マーケティング・スポットの中の希望する位置にイニシアチブを表示するようスケジューリングすることによって割り当てます。ストアに e-マーケティング・スポットを追加するための詳細については、 479 ページの『第 42 章 ストアへの e-マーケティング・スポットの追加』を参照してください。

キャンペーン・イニシアチブには、表示するタイミングと対象を決定する条件が含まれます。その条件は、イニシアチブが作成された時点で定義され、イニシアチブの存続期間中にイニシアチブの可視性や表示内容を調整するために変更することも可能です。顧客プロファイルの詳細については、 323 ページの『第 32 章 顧客プロファイル』を参照してください。

キャンペーン・イニシアチブでは、その使用に関する統計が生成されます。それらの統計情報は、マーチャント、マーケティング・マネージャー、および取引管理マネージャーが、 WebSphere Commerce アクセラレーターを使用して表示できます。統計は、イニシアチブがインプリメントされている各 e-マーケティング・スポットごとに、そのイニシアチブの閲覧率を示します。これらの統計は、イニシアチブの効果や、これが表示されるさまざまな場所における比較成功率についてのフィードバックを示します。

WebSphere Commerce でのキャンペーン資産の作成

多くの場合、キャンペーンとキャンペーン・イニシアチブは、マーケティング・マネージャーによって、または WebSphere Commerce アクセラレーターにおいて「キャンペーン」ウィザードや「キャンペーン・イニシアチブ」ウィザードを使用して取引管理マネージャーによって作成されます。詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

ストアに e-マーケティング・スポットを追加するための詳細については、 479 ページの『第 42 章 ストアへの e-マーケティング・スポットの追加』を参照してください。

第 21 章 決済手段

WebSphere Commerce には、 WebSphere Commerce Payments (以前の IBM Payment Manager) というオプションのコンポーネントが提供されています。

ストアで WebSphere Commerce Payments を使用したい場合は、ストア・アーカイブ中に支払資産ファイルを含める必要があります。ストア・アーカイブの発行前に Payments インスタンスが開始済みであることを確認してください。そうすると、ストア・アーカイブの発行時に、支払資産ファイル (サンプル・ストア・アーカイブの一部として含まれている) により、WebSphere Commerce Payments 中で以下の情報がセットアップされます。

- WebSphere Commerce Payments データベース中の merchant_ID。
- ストアで使用されるカセットのタイプ。
- 支払資産ファイル中でストアによってサポートされていると指定されている通貨ごとの、 WebSphere Commerce Payments データベース中のアカウント。支払資産ファイル中に指定されている通貨がストアでサポートされていない場合は、アカウントは作成されません。
- アカウントごとの 1 つ以上のブランド。

支払資産ファイルをセットアップし、 WebSphere Commerce Payments を使用するようにストアをセットアップするには、以下のようにします。

- ストア発行時に管理コンソールを使ってロードされる支払データを、 XML ファイルの形式 (`paymentinfo.xml`) で作成する。これにより、発行するストアについて指定されているマーチャントおよびブランドのタイプに対応するよう、 WebSphere Commerce Payments が構成されます。詳細については、 238 ページの『XML ファイルを使用した支払資産の作成』を参照してください。

  注: `paymentinfo.xml` は、 WebSphere Commerce Server データベースの中のテーブルにデータを入れるわけではありません。このファイルは WebSphere Commerce Payments を構成します。 `paymentinfo.xml` が適用されるのは、支払い方式としてオフライン・クレジット・カードを使用している場合だけです。

  ストア・アーカイブを発行し終えたら、サンプル・ストア・アーカイブ中にセットアップされた支払い情報を使用して、オーダーを発行できます。新しいブランドを追加する場合は、 WebSphere Commerce Payments を個々のブランドを扱うように構成する必要があります。
- OfflineCard または CustomOffline カセット以外の IBM 決済カセットを使用する場合は、 239 ページの『決済カセット用の環境のカスタマイズ』で説明されているようにサンプル・ストアの Web 資産に変更を加える。
- 管理コンソールまたは WebSphere Commerce Payments のユーザー・インターフェースを使用することによって、 WebSphere Commerce Payments をストアに合わせてセットアップする。管理コンソールを使用する場合には、 Payments のメニューにメニュー項目が表示されます。 WebSphere Commerce Payments のユーザー・インターフェースを使用する場合には、ナビゲーション・フレームの中の

「管理」の下にメニュー項目が表示されます。セットアップのタスクの詳細については、 WebSphere Commerce オンライン・ヘルプの『ストアに合わせた *WebSphere Commerce Payments* のセットアップ』を参照してください。

WebSphere Commerce Payments 以外の支払い手段を使用する場合でも、その支払い手段を使用するためのステップは以下の手順と似ています。

XML ファイルを使用した支払資産の作成

XML ファイルを使用してストアの支払資産を作成するには、以下のようにします。

1. サンプル・ストアの支払資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。

   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

   - *WC_installdir*/samplestores

     注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

   各サンプル・ストアには 1 つの paymentinfo.xml ファイルが組み込まれています。これには支払いに関する情報が入っています。ストア・アーカイブの paymentinfo.xml ファイルを表示するには、 ZIP プログラムを使用してこれを解凍します。 paymentinfo.xml ファイルは、データ・ディレクトリーにあります。

2. サンプル・ストア・アーカイブの paymentinfo.xml ファイルの 1 つをコピーするか、新しいファイルを作成することにより、paymentinfo.xml ファイルを作成します。詳細については、paymentinfo.xml に対応する DTD ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。

   WC_installdir/xml/sar

3. WebSphere Commerce Payments を使用可能または使用不可にします。

   a. 次の例を参考にして、XML ファイルの中で WebSphere Commerce Payments を使用可能または使用不可にし、ストアの受け付ける決済カセット、通貨、およびブランドの種類を指定します。

```
<paymentinfo>
  <PaymentManager enable="yes"/>
  <Cassette type="OfflineCard">
   <Account currency="USD">
    <Brand type="MasterCard"/>
    <Brand type="VISA"/>
    <Brand type="American Express"/>
   <Account/>
   <Account currency="EUR">
    <Brand type="MasterCard"/>
    <Brand type="VISA"/>
    <Brand type="American Express"/>
   </Account>
  </Cassette>
</paymentinfo>
```

   ここで、

   - enable には、WebSphere Commerce Payments を使用可能にするか、それとも使用不可にするかを指定します。 WebSphere Commerce Payments を使用不可にした場合、 Payments ユーザー・インターフェースは機能を続

けますが、ストアは Payments コンポーネントによって決済取引を処理できなくなります。 Payments コンポーネントを使用不可にした場合は、支払い情報エレメントに他の情報を指定する必要はありません。

- `Cassette type` は、サポートされるカセットの種類です。
- `Account currency` は、ストアでサポートする通貨です。 OfflineCard カセット・タイプを使用している場合には、アカウント通貨が必要です。通貨は、ISO 4217 規格に従って 3 文字のコードで指定します。たとえば、米国ドルの場合は ″USD″ です。
- `Brand type` は、指定されたアカウントと通貨でサポートされるクレジット・カードの種類です。

決済カセット用の環境のカスタマイズ

WebSphere Commerce では、決済カセットとして OfflineCard カセットか CustomOffline カセットを使用して決済取引を処理できるサンプル・ストアが提供されています。これらのカセットはサンプル・ストアの支払い方式として使用できるように自動的に構成されます。その他の決済カセットを使用するには、サンプル・ストアの Web 資産に変更を加える必要があります。以下の指示は、 WebSphere Commerce で提供されているその他の IBM 決済カセットを使用するように、環境をカスタマイズする方法について説明しています。

ストアで IBM 決済カセットを使用するには、まずインストールする WebSphere Commerce Payments コンポーネントを選択する必要があります。インストールに関する指示は、ご使用のプラットフォーム版の「*WebSphere Commerce インストール・ガイド*」に記載されています。 WebSphere Commerce インストール・プログラムは、 Payments のフレームワークとカセット・ソフトウェアの両方をインストールします。次に、WebSphere Commerce 構成マネージャーを使用して、 Payments インスタンスの作成やカセットのインスタンスへの追加などの、必要なインストール後タスクを実行しなければなりません。 Payments インスタンスの構成に関する指示は、「*WebSphere Commerce インストール・ガイド*」および構成マネージャーのオンライン・ヘルプを参照してください。

決済カセットを Payments インスタンスに追加し終えたら、以下のカスタマイズのステップを調べて、選択した決済カセットを使用して WebSphere Commerce サンプル・ストアが決済を処理できることを確認してください。

1. store .jsp ファイルに変更を加えて、決済カセットを指定する。
2. カセットのキャッシャー・プロファイルを検査する。
3. クラーク・オーダー (ゲスト・オーダー) 発行ページをサポートする cassette's .jsp ファイルを検査する。
4. マーチャント設定を構成する。

これらのステップについて、以下に説明されています。

store .jsp ファイルの変更

サンプル・ストアで OfflineCard カセットか CustomOffline カセットを使用しない場合は、ストアの jsp ファイルを変更する必要があります。デフォルトでは、ストアの .jsp ファイルは OfflineCard カセットを使用するようにセットアップされてい

ます。したがって、その他のカセットを使用にはこのファイルを変更しなければなりません。 FashionFlow も CustomOffline カセットを使用します。

加えられる変更を確認するための .jsp ファイルのリストについては、 241 ページの表 10を参照してください。

.jsp ファイルを変更するには、以下のステップに従います。

1. FashionFlow などのサンプル・ストアを使用して、 WebSphere Commerce 中にストアを作成します。
2. 以下に示すディレクトリーに進みます。

   WAS_installdir/installedApps/*cell_name*/
   WC_instance_name.ear/Stores.war/

   400 iSeries の場合、パスは以下のとおりです。

   WAS_userdir/*WAS_instance_name*/
   installedApps/*cell_name*/*WC_instance_name*.ear/Stores.war/. 作成したストアの war ディレクトリーには独自のディレクトリーがあります。
3. テキスト・エディターで、ストアのディレクトリーから、 `OrderSubmitForm.jsp` ファイルをオープンします。

   Business WebSphere Commerce アクセラレーターの Contract Tools は、すべての決済カセットをサポートしています。 `OrderSubmitForm.jsp` ファイルは、バイヤー組織とセラーの間でセットアップされた契約の支払い条件に従わなければなりません。
4. `OrderSubmitForm.jsp` ファイル中で、以下のテキストを検索します。

   ```
   if (info[i].getPolicyName().trim().equals("OfflineCard"))
   ```

   支払いポリシーの名前を `OfflineCard` から以下のうち該当する名前に変更します。

   ```
   CustomOffline
   BankServACH
   Paymentech
   VisaNet
   VisaNet_PCard
   ```

   ポリシーの詳細については、 WebSphere Commerce オンライン・ヘルプの『データベース・テーブル: POLICY』を参照してください。

   CustomOffline のポリシーは、現金引換 (COD)、掛け売り、WebSphere Commerce Payments の外部で取り引きされることの多いクーポンによる決済など、カスタム決済取引の処理をサポートします。

   BankServACH のポリシーは、Automated Clearing House Network (ACH) と連結する BankServ 決済ゲートウェイを使用するオンライン電子小切手決済をサポートします。

   Paymentech のポリシーは、クレジット・カードと非 PIN ベースのデビット・カード決済のオンライン与信と清算をサポートします。

   VisaNet のポリシーは、 Vital Processing Services または First Horizon Merchant Services (FHMS) 決済ネットワークを使用するクレジット・カード取引をサポートします。

注: カード・サポートを購入しており VisaNet 用のカセットを使用する場合は、VisaNet ではなく VisaNet_PCard を選択してください。

これらのカセットの詳細については、「カセットの補足」を参照してください。

ストアがクイック・チェックアウト機能を使用する場合、これらの他のファイルでの支払いポリシーの名前も変更してください。

```
ShoppingArea¥CheckoutSection¥QuickCheckoutSubsection¥QuickCheckoutForm.jsp
UserArea¥AccountSection¥QuickCheckoutProfileSubsection¥QuickCheckoutProfileForm.jsp
```

5. (オプション) 決済にクレジット・カード方式を使用していて、ユーザーから追加の情報を収集するためにユーザー・インターフェースにフィールドを追加する必要がある場合は、可能な変更方法について StandardCreditCard.jsp ファイルも参照してください。パスの情報については、表 10を参照してください。

   クレジット・カードの使用が関係する特定の支払いメソッドを使って購入を実行するときに、クレジット・カードのブランドを表示するには、オプション値がその支払いメソッドの .jsp ファイルに存在するようにします。たとえば、Paymentech 支払いメソッドを使って購入を実行するときに、クレジット・カードのブランドを表示するには、<select name="cardBrand"> を検索します。そのテキストの下に新しい行で以下を追加します。

   ```
   <option value="Paymentech">Paymentech</option>.
   ```

表 10. 確認するストア .jsp ファイル

JSP ファイル	ビジネス・モデル/サンプル	変更の目的
/ShoppingArea/CheckoutSection/StandardCheckoutSubsection/OrderSubmitForm.jsp	消費者向け (Fashion Flow または Express)、ホストされる販売店	OfflineCard からの支払ポリシー名の変更
/ShoppingArea/CheckoutSection/QuickCheckoutSubsection/QuickCheckoutForm.jsp /UserArea/AccountSection/QuickCheckoutProfileSubsection/QuickCheckoutProfileForm.jsp	消費者向け (Fashion Flow または Express)	OfflineCard からの支払ポリシー名の変更
/ShoppingArea/CheckoutSection/StandardCheckoutSubsection/StandardCreditCard.jsp	消費者向け (Fashion Flow または Express)	クレジット・カードのブランドを表示可能にする
/ShoppingArea/CheckoutSection/StandardCheckoutSubsection/StandardCreditCardDisplay.jsp	B2B 向け (ToolTech)、バリュー・チェーン - サプライ	クレジット・カードのブランドを表示可能にする

カセットのキャッシャー・プロファイルの検査

WebSphere Commerce で提供されている IBM 決済カセット用の WebSphere Commerce Payments キャッシャー・プロファイルを、使用できるようにする必要があります。キャッシャー・プロファイルは、Payments コンポーネントでオーダーを作成するために使用します。

キャッシャー・プロファイルを編集して、 APPROVEFLAG パラメーターや DEPOSITFLAG パラメーターなどの特定のパラメーターを設定することもできます。カセット・パラメーターが同一でない場合もあるので、パラメーターの設定の詳細については、以下のカセットの補足を参照してください。

- *WebSphere Commerce Payments CustomOffline Cassette Supplement*
- *WebSphere Commerce Payments OfflineCard Cassette Supplement*
- *WebSphere Commerce Payments Cassette for BankServACH Supplement*
- *WebSphere Commerce Payments Cassette for Paymentech Supplement*
- *WebSphere Commerce Payments Cassette for VisaNet Supplement*

IBM 提供の決済カセットに関連したキャッシャー・プロファイルには、以下のものが含まれます。

```
WC51_BankServACH.profile
WC51_CustomOffline_BillMe.profile
WC51_CustomOffline_COD.profile
WC51_OfflineCard.profile
WC51_VisaNet.profile
WC51_VisaNet_PCard.profile
WC_Paymentech.profile
```

カセット・プロファイルは、 WebSphere Commerce インスタンス・プロファイル・ディレクトリーに保管する必要があります。

プロファイルが保管されるディレクトリーを見つけるには、作成したインスタンス用の WebSphere Commerce 構成ファイルを探してください。デフォルトのインスタンス名の「demo」を使用した場合は、構成ファイルは以下のようになります。

WC_installdir/instances/demo/xml/demo.xml

400 iSeries の場合、パスは以下のとおりです。

WC_userdir/instances/demo/xml/demo.xml

次に、構成ファイル中の Payment Manager エレメントの ProfilePath 属性で指定されているディレクトリーを見つけます。この属性は、プロファイルがある場所を指定します。デフォルトのインスタンス名の「demo」を使用した場合は、プロファイルの保管先のディレクトリー・パスは以下のようになります。

WC_installdir/instances/demo/xml/payment

400 iSeries の場合、パスは以下のとおりです。

WC_userdir/instances/demo/xml/payment

カセットのキャッシャー・プロファイルを編集してパラメーターを設定する場合は、 *WC_installdir*/instances/*instance_name*/xml/payment ディレクトリー (*instance_name* は使用しているインスタンスの名前) 中にこのプロファイルを保管したことを確認してください。

400 iSeries の場合、パスは以下のとおりです。

WC_userdir/instances/instance_name/xml/payment

決済ビジネス・ポリシーで実際に使用されるキャッシャー・プロファイルは、決済ビジネス・ポリシーの Properties フィールド中の profileName プロパティー値で指

定されます。ビジネス・ポリシーの詳細については、オンライン・ヘルプの『データベース・テーブル：POLICY』を参照してください。

cassette .jsp ファイルの検査

オーダー・クラークが顧客の代わりにゲスト・オーダーを発行する場合は、決済は WebSphere Commerce アクセラレーターで処理されます。カセットの決済データは、カセットの .jsp ファイルを使用して収集されます。

WebSphere Commerce では、カセットの .jsp ファイルのことを「決済属性ページ」といいます。実際に使用されるページは、決済ビジネス・ポリシーの Properties フィールド中の attrPageName プロパティー値で指定されます。詳細については、WebSphere Commerce オンライン・ヘルプの『データベース・テーブル: POLICY』を参照してください。ストアのフローと WebSphere Commerce アクセラレーターのフローで、決済属性ページを使用する必要があります。

カセットの .jsp ファイルは、以下のディレクトリー中にすでに存在している必要があります。

WC_installdir/wc.ear/CommerceAccelerator.war/tools/order/buyPages/
WAS_installdir/installedApps/*cell_name*/WC_demo.ear/CommerceAccelerator.war/tools/order/buypages

400 iSeries の場合、パスは以下のとおりです。

QIBM/userdata/webas5/base/*WAS_instance_name*/installedApps/*cell_name*/WC_demo.ear/CommerceAccelerator.war/tools/order/buypages

「購買ページ」の情報をカスタマイズする場合は、それに応じて .jsp ファイルを変更します。

WebSphere Commerce Payments でのマーチャント設定の構成

IBM 決済カセットのマーチャントを構成するには、カセットの補足に記載されている指示に従ってください。 WebSphere Commerce 管理コンソールか、 Payments のユーザー・インターフェース (http://*host_name:port*/webapp/PaymentManager) を使用して、マーチャント設定を変更することができます。マーチャント設定を構成するには、 WebSphere Commerce Payments 内で決済管理者権限とマーチャント管理者権限がなければなりません。

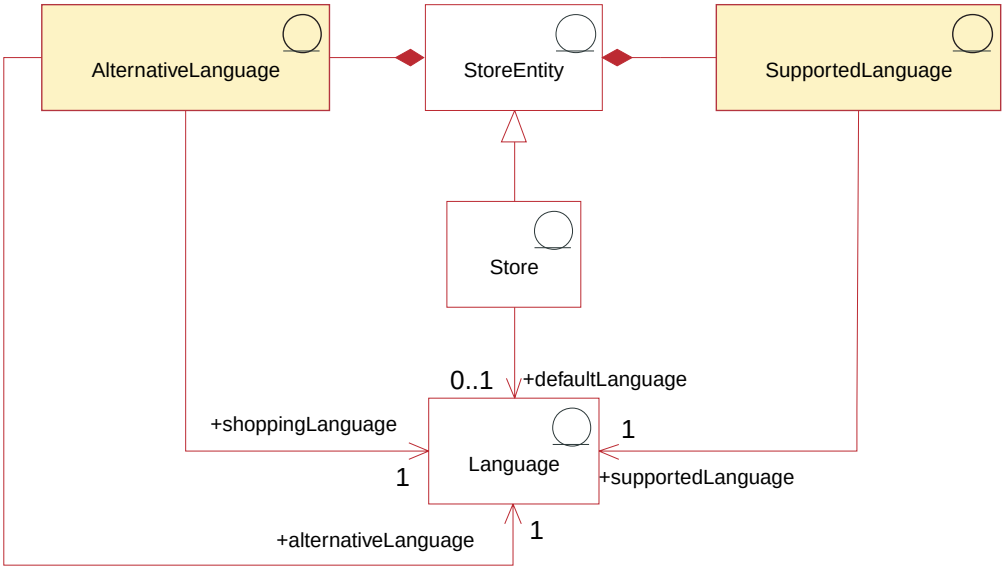
第 22 章 言語資産

WebSphere Commerce では、サイトに多数の使用できる言語を定義することができます。インスタンス作成の際に、LANGUAGE テーブルにはサポートされる 10 言語を含めることができます。すなわち、ドイツ語、中国語 (繁体字)、中国語 (簡体字)、日本語、韓国語、イタリア語、フランス語、スペイン語、ブラジル・ポルトガル語、英語です。サイトでは、さまざまな文化や地域の顧客に対する情報の表示方法を調整するために、さらに付加的な言語や既存の言語の方言を定義できます。

WebSphere Commerce の言語資産について

言語資産を理解するには、言語とストアの間の関係を理解することが必要です。これは、以下の情報モデルを使用して説明できます。次に、ストアおよび他の資産と言語との関係、および関連を説明します。

以下の図は、言語資産情報モデルを示しています。



WebSphere Commerce の言語には 4 つの種別があります。これらの種別は、以下のとおりです。

- デフォルト言語。
- サポートされる言語。
- 代替言語。
- ショッピング言語。

これらの種別のそれぞれは、ストアで異なる役割を果たします。すべての言語は、LANGUAGE テーブルに保管されます。

デフォルト言語

デフォルト言語 は、各ストアに関連付けられます。これは、ストアが主な言語として使用するよう選択した言語であり、ショッピング言語を明示的に選択しない顧客に対して表示される言語になります。ストアのデフォルト言語は、暗黙のうちにストアによってサポートされます。つまりストアでは、デフォルト言語 (または LANGPAIR テーブルで定義されている場合には、その代替言語のいずれか) による情報の表示が常に可能でなければなりません。サポートされている言語または代替言語のいずれかで情報が利用できない場合、情報はデフォルト言語で表示されます。

サポートされる言語

STORELANG テーブルは、各ストアでサポートする言語を指定します。ストアでは、*サポート言語*、または LANGPAIR テーブルで定義されている場合には、その代替言語のいずれかで情報を表示することが可能でなければなりません。またストアは、そのストア・グループでサポートされているすべての言語をサポートする必要があります。

サポートされる言語の追加の詳細については、 351 ページの『ストアへの言語の追加』を参照してください。

代替言語

サポートされる言語のいずれかで情報を表示できない場合、ストアは、*代替言語* による情報の表示を試みます (使用可能な場合)。ストアでは、代替言語を 1 つ 1 つを試す順序を指定できます。ストアの代替言語には、そのストア・グループの代替言語が含まれます。代替言語は、情報の一部が 1 つの言語でしか利用できないが、それとは異なる関連言語で買い物をする顧客にも表示する必要があるという場合に便利です。例えば、情報をサポートされているすべての言語に翻訳する作業がまだ完了していない場合、あるいは同じ言語の 2 つの類似した方言がサポートされていて、情報が同一の場合などです。

WebSphere Commerce Server の言語資産の構造の詳細については、WebSphere Commerce オンライン・ヘルプの『言語データ・モデル』を参照してください。

WebSphere Commerce での言語資産の作成

次の方法のうち 1 つを使用して、ストアがサポートする言語を定義できます。

- WebSphere Commerce アクセラレーター中のストア・ツールを使用する。
- XML ファイルを使用する。 XML ファイルは、ローダー・パッケージでロードすることも、管理コンソールの発行ツールでロードすることもできます。
- SQL の挿入 (INSERT) を使用してデータベースを直接編集する。
- SQL の編集および更新を使用する。

注: このツールは、事前に読み込み済みの XML ファイルを、ストア・アーカイブ形式で処理します。

ストアがサポートする言語をストア・ツールを使用して定義する方法の詳細については、 WebSphere Commerce オンライン・ヘルプを参照してください。ストアがサポートする言語を XML ファイルに定義する方法の詳細については、 145 ページの『XML ファイルによるストア・データ資産の作成』を参照してください。

第 23 章 通貨資産

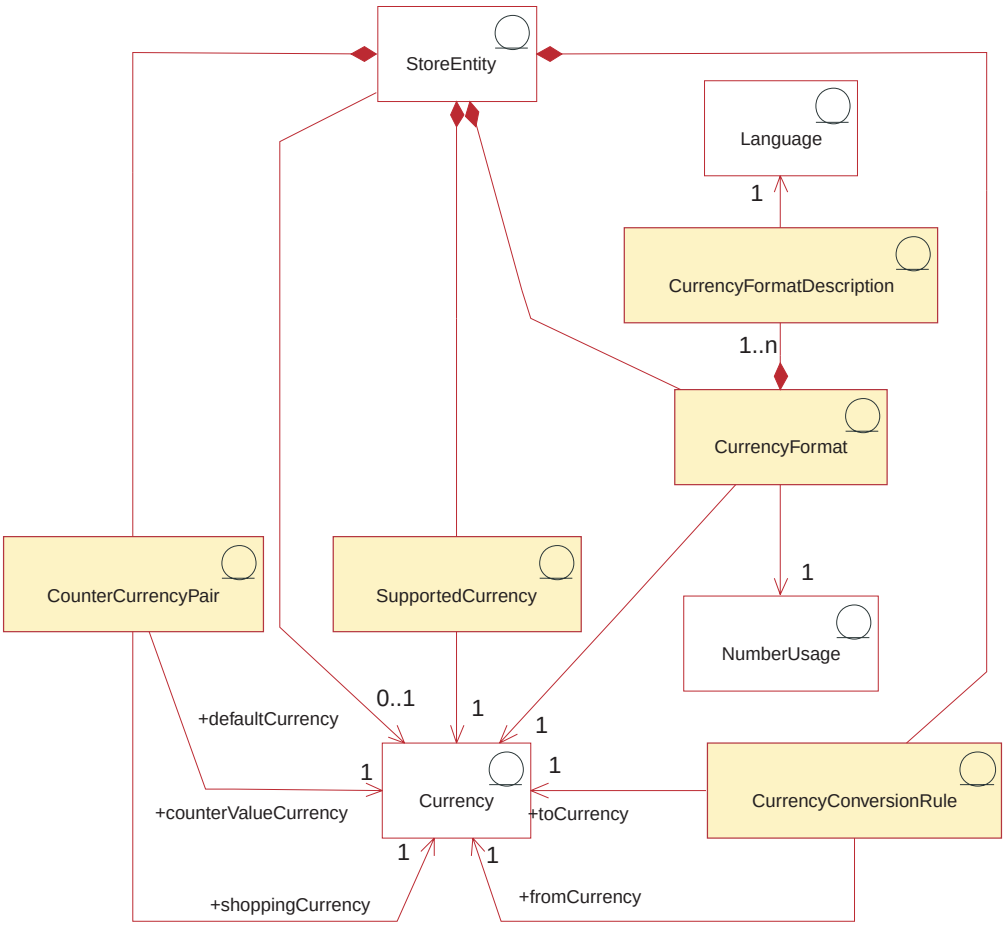
サイトでは、1 種類の通貨で価格を表示することもできますが、ユーロ用に用意された指示に従って複数の通貨を使用することも可能です (251 ページの『カウンター通貨』 を参照)。複数のストアを擁するサイトでは、ストアによって異なる通貨を使用したり、ストア・グループに通貨を割り当てたりすることができます。作成しようとしているサイトの特性によっては、使用したい通貨とその表示法を指定することができます。

WebSphere Commerce では、顧客がショッピング通貨を選択できるようにすることができます。ショッピング通貨は、顧客が特定のストアで商品に支払う通貨です。ストア・ページ上のすべての金額はこの通貨で表示されます。顧客がショッピング通貨を変更すると、ショッピング・カートに追加したアイテムの価格とオーダー合計価格は、自動的に新しいショッピング通貨に換算され、再計算されて表示されます。

顧客は、ユーロを含む多くの通貨でショッピングできます。ユーロは 1999 年 1 月 1 日に正規の通貨になり、現在は金融市場で使用されています。ユーロ通貨とすべての参加国の通貨の換算率 (為替レート) は固定されています。

WebSphere Commerce の通貨資産について

以下の図は、WebSphere Commerce Server における通貨構造を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、111 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、487 ページの『付録 A. UML の凡例』を参照してください。

上の図では、通貨が情報モデルの中心にあります。各ストア、またはストアのグループには、デフォルト通貨があります。

通貨形式

ストア・エンティティーには、多数の*通貨形式設定* のルールを設定できます。ストアに特定通貨の形式設定ルールがない場合は、そのストア・グループの形式設定ルールが使用されます。通貨形式は、CURFORMAT テーブルでセットアップされます。

他のストアによって使用できる通貨形式資産については、151 ページの『第 14 章 ストア間の関係』で説明されています。

数値の使用法

形式設定された各通貨ルールには、*数値の使用法* が 1 つ関連付けられます。数量や金額などの数値は、それに関連する使用法に応じて、さまざまに丸めたり形式設

定したりできます。ストアは、表示する数値の使用方法に応じて、それぞれの数値に異なる丸めと形式設定のルールを指定することができます。たとえば、あるストアは単価については単価の使用法を指定して小数第 4 位で丸め、他の通貨の額についてはデフォルトの使用法を指定して小数第 2 位で丸めるかもしれません。数値の使用法は、NUMBRUSG テーブルの中に保存されます。

通貨形式の説明

通貨形式のルールには、多数の*通貨形式記述* を設定できます。通貨形式記述は、ある数量を (表示目的で) 特定言語の特定の数量単位で形式設定する方法を説明します。各説明は、LANGUAGE テーブルで言語と関連付けられます。言語資産の詳細については、245 ページの『第 22 章 言語資産』を参照してください。グローバリゼーションのサポートの詳細については、337 ページの『第 34 章 グローバリゼーション』を参照してください。通貨形式記述は、CURFMTDESC テーブル保存されます。

サポートされる通貨

ストア・エンティティーには、多数の*サポートされる通貨* を設定できます。サポートされる通貨とは、支払いを受け取る通貨です。

他のストアによって使用可能な、サポートされている通貨資産については、151 ページの『第 14 章 ストア間の関係』で説明されています。

通貨変換ルール

すべての通貨には、他の通貨との間で変換を制御するルールがあります。各*通貨変換ルール* は、価格 (特定通貨のデータベースに保管されている) を変換し、サポートされているショッピング通貨による顧客への請求額を算出するために使用できます。

他のストアによって使用可能な、サポートされている通貨変換ルール資産については、151 ページの『第 14 章 ストア間の関係』で説明されています。

カウンター通貨

カウンター通貨 は、サポートされる通貨に伴って表示される通貨金額です。これは購入では使用できませんが、情報表示の目的で使用されます。顧客は、ユーロでショッピングすることにした場合、欧州通貨統合の通貨と他の通貨で金額をストアに表示することができます。ショッピング通貨の金額は、そのショッピング通貨に対応するすべてのカウンター値の通貨に変換されます。カウンター通貨は、オランダ・ギルダーとユーロのように、サポートされる通貨と対にされます。カウンター通貨の対は、CURCVLIST に保管されます。

他のストアによって使用できる通貨のカウンター値資産については、151 ページの『第 14 章 ストア間の関係』で説明されています。

WebSphere Commerce Server の通貨資産の構造の詳細については、WebSphere Commerce オンライン・ヘルプの『通貨データ・モデル』を参照してください。

WebSphere Commerce での通貨資産の作成

WebSphere Commerce の管理コンソールを使用すると、サポートされる通貨をストアに追加したり、ストアのデフォルト通貨を選択したりできます。管理コンソールを使用して編集できる資産の詳細については、 WebSphere Commerce オンライン・ヘルプのトピック『ストア・データベース資産の変更』を参照してください。

注: 管理コンソールは、事前に読み込み済みの XML ファイルを、ストア・アーカイブ形式で処理します。

また、サポートされる通貨とデフォルト通貨を XML ファイルを使用してストアに追加することもできます。そのファイルは、ローダー・パッケージを使用してデータベースにロードできます。この方法では、通貨換算やカウンター値の通貨など、別の種類の通貨資産の作成も可能です。

通貨の処理に関する情報は、WebSphere Commerce オンライン・ヘルプを参照してください。新しい通貨資産を XML ファイルの形式で作成することに関する情報は、 『XML ファイルを使用した通貨資産の作成』を参照してください。

XML ファイルを使用した通貨資産の作成

ストアの通貨資産は、XML ファイルの形式で作成します。その XML ファイルは、ローダー・パッケージを使用して、データベースにロードできます。ローダー・パッケージの詳細については、 363 ページの『第 10 部 ストアの発行』を参照してください。

資産を作成する前に、 429 ページの『第 38 章 WebSphere Commerce データベース資産グループのロード』で扱われている内容に精通する必要があります。

XML ファイルを使用してストアの通貨資産を作成するには、以下のようにします。

1. サンプル・ストアの通貨資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。

   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

   - *WC_installdir*/samplestores

     注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

   個々のサンプル・ストアには 1 つの `currency.xml` ファイルが組み込まれており、このファイルには通貨情報が組み込まれています。ストア・アーカイブの `currency.xml` ファイルを表示するには、 ZIP プログラムを使用してこれを解凍します。 `currency.xml` ファイルはデータ・ディレクトリーにあります。
2. 489 ページの『付録 B. データの作成』に記載されている情報を確認します。
3. サンプル・ストア・アーカイブの `currency.xml` ファイルの 1 つをコピーするか、新しいファイルを作成することにより、`currency.xml` ファイルを作成します。詳細については、*`WC_installdir`*`/schema/xml` ディレクトリー中の `wcs.dtd` ファイル、またはストア・アーカイブ中に組み込まれている DTD を参照してください。

4. ストアでサポートされる通貨を定義します。
   a. 次の例を参考にして、 XML ファイルの CURLIST テーブルでストアがサポートする通貨を定義します。

      ```
      <curlist currstr="USD" storeent_id="@storeent_id_1" />
      ```

      ここで、
      - currstr は、サポートされる通貨を表す 3 文字の ISO 4217 通貨コードです。このコードは、 SETCURR テーブルの SETCCURR 列に出現するものでなければなりません。ストアはサポートされているすべての通貨での支払いを受諾できなければなりません。
      - storeent_id は、ストア・エンティティーです。

   b. ストアがサポートする各通貨ごとに繰り返します。

ストアのデフォルト通貨は、STOREENT テーブルで定義されます。詳細については、 145 ページの『XML ファイルによるストア・データ資産の作成』を参照してください。

5. (オプション) ストアの価格がどのような通貨で表示されるかは、価格の設定方法によって異なります。ストアで使用するあらゆる通貨に対して価格を定義したり、デフォルト通貨だけについて価格を定義したりできます。価格の設定については、 201 ページの『WebSphere Commerce での価格設定資産の作成』を参照してください。

   価格の設定においてデフォルト通貨だけについて価格を定義した場合に、ストアにおいてサポートされている他の通貨で価格を表示したいなら、換算率をストアに追加する必要があります。その換算率を使用することにより、デフォルト通貨からサポートされる通貨に変換します。
   a. 換算前の通貨 (たとえば米国ドル USD) と、 1 つ以上の換算後の通貨 (たとえば日本円 JPY) とを決定します。各通貨の ISO 通貨コードについては、ISO 4217 の国際通貨についての情報を参照してください。
   b. 次の例を参考にして、CURCONVERT テーブルに換算情報を追加します。

      ```
      <curconvert
      storeent_id="@storeent_id_1"
      fromcurr="USD"
      tocurr="JPY"
      factor="105.10"
      multiplyordivide="M"
      bidirectional="Y"
      updatable="Y"
      curconvert_id="@curconvert_id_1" />
      ```

      ここで、
      - storeent_id は、ストア・エンティティーです。
      - fromcurr は、換算前の通貨です。通常、FROMCURR 通貨の金額は、価格、割引、配送料など、販売対象商品に関連した金額を決定するために使用されるルールやその他の情報の一部です。
      - tocurr は、換算後の通貨です。通常、TOCURR は顧客が支払う通貨です。多くの場合、この通貨の金額は、単価、配送料、税額など、オーダー・アイテムの一部です。
      - factor は、換算係数です。

- multiplyordivide は、FROMCURR から TOCURR への換算に関して、次のように指定します。
  - M = FACTOR を乗算する。
  - D = FACTOR で除算する。

  双方向ルールの場合、逆の操作を使用して TOCURR から FROMCURR に換算されます。
- bidirectional は、ルールが双方向か単一方向かを指定します。
  - Y = 双方向
  - N = 単一方向
- updatable は、通貨換算ルールを管理するユーザー・インターフェースで使用することを意図したフラグです。有効な値は、以下のとおりです。
  - N = 換算率は変更不能 - 決して変更できません。
  - Y = 換算率は変更可能です。
- curconvert_id は、生成される固有キーです。

c. 価格を表示するすべての通貨について、ステップ a と b を繰り返します。

価格設定情報の中で、サポートされているすべての通貨について価格を定義した場合であっても、ストアでサポートされる通貨のための通貨換算率を定義できます。

6. (オプション) ショッピング通貨とカウンター通貨の両方に表示価格を含めたいなら (たとえばオランダのギルダーとユーロの両方による表示価格の場合)、CURCVLIST テーブルに情報を追加する必要があります。

   a. 次の例を参考にして、CURCVLIST テーブルに情報を追加します。

   ```
   <curcvlist
   storeent_id="@storeent_id_1"
   currstr="NLG"
   countervaluecurr="EUR"
   displayseq="1" />
   ```

   ここで、
   - storeent_id は、ストア・エンティティーです。
   - currstr は、通貨を表す three 文字の ISO 4217 通貨コードです。このコードは、 SETCURR テーブルの SETCCURR 列に出現するものでなければなりません。通常、FROMCURR 通貨の金額は、価格、割引、配送料など、販売対象商品に関連した金額を決定するために使用されるルールやその他の情報の一部です。
   - countervaluecurr は、カウンター値通貨を表す three 文字の ISO 4217 通貨コードです。このコードは、 SETCURR テーブルの SETCCURR 列に出現するものでなければなりません。
   - displayseq は、カウンター値通貨の表示順序を指定する数値です。カウンター値通貨は、 CURCVLIST テーブルの DISPLAYSEQ 列で指定されるカウンター値表示順序に従い、昇順で表示されます。

@ と **&** の使用法の詳細については、 489 ページの『付録 B. データの作成』を参照してください。

通貨に関するその他のタスク

一般通貨について、また通貨に関する以下のようなその他のタスク、

- 現在のところ WebSphere Commerce でサポートされていない新しい通貨を追加する
- 既存の通貨形式を変更する

の詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

第 24 章 計測単位資産

商品の販売、在庫の追跡は、さまざまな数量単位 (キログラム、インチ、リットルなど) で行うことができます。商品は、これらの単位の最小数量や、それに特定数量を掛けた数でオーダーすることができます。

コントローラー・コマンドは、UOM (計測単位) を使用して数量単位を指定します。 UOM パラメーターが指定されていない場合、顧客が指定した数量に、CATENTSHIP データベース・テーブル内のカタログ・エントリーの名目数量が掛けられます。結果は要求された数量として認識されます。

要求数量は、カタログ・エントリーの、一番近い倍数に切り上げられます。たとえば、倍数が 2 kg で要求数量が 4.1 kg の場合、切り上げの結果は 6 kg になります。切り上げ済み数量は在庫の検査時に使用されます。在庫には、独自の数量単位があります。在庫の数量単位とカタログ・エントリーの数量単位が異なる場合、 2 つの単位間での変換が必要です。

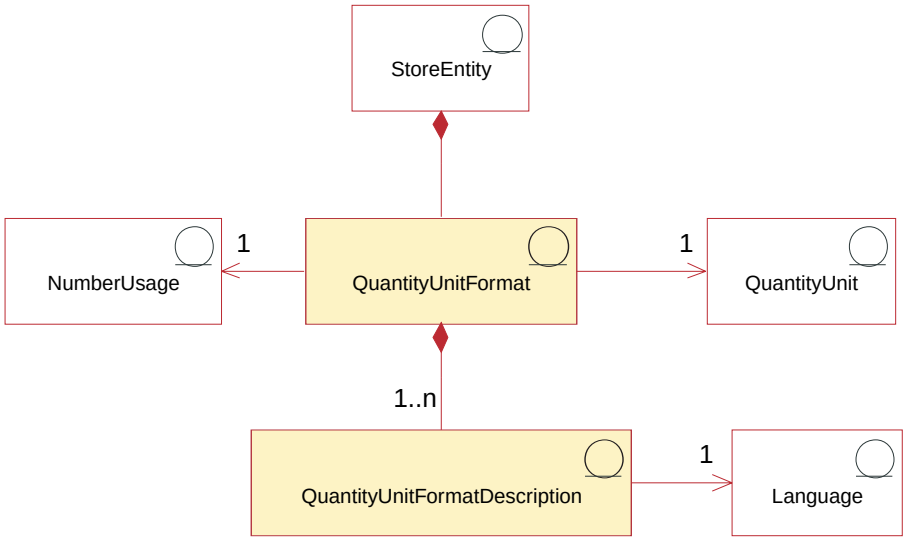
販売可能在庫数量による納期回答 (ATP) 在庫が使用可能な場合 (STORE の ALLOCATIONGOODFOR 列を参照)、在庫の数量単位は BASEITEM テーブルの QUANTITYMEASURE 列で定義されます。それ以外の場合は、INVENTORY テーブルの QUANTITYMEASURE 列で定義されます。

丸めの数量をカタログ・エントリーの名目上の数量で割ったものは、正規化された数量として認識されます。正規化された数量は、実行するコマンドに応じて、オーダー・アイテムまたは興味のあるアイテムに保管されます。たとえば、切り上げ済み数量が 6 kg で名目数量が 2 kg の場合、正規化数量は 3 になります。

カタログ・エントリーに対するオファーを検索する際、要求された数量はどのオファーが最適価格を提示するかに作用し、その結果どのオファーを使用するかが決まります。たとえば、切り上げ済み数量が 6 kg で 2 つのオファーがあるとします。一方は、名目数量 2 kg、最小数量 10 kg に対して価格を 4 ドルとするもの、もう一方は、名目数量 2 kg、最小数量 2 kg に対して価格を 4.5 ドルとするものです。この場合、使用できるのは、後者のオファーだけです。

WebSphere Commerce の計測単位について

次の図は、WebSphere Commerce Server における計測単位の構造を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、111 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、487 ページの『付録 A. UML の凡例』を参照してください。

数量単位と数量単位形式

数量単位 は、ストアで使用される計測単位です。たとえばキログラム、ポンド、メートル、インチ、リットルなどです。数量単位形式は、この数量単位がストアで形式設定される方法、たとえば数量単位の表示時に使用される小数部の桁数などです。

各*数量単位形式* はただ 1 つのストア・エンティティーの一部ですが、各ストア・エンティティーは複数の数量単位形式を持つことがあります。

1 つの数量単位と数値の使用法に対して 1 つの数量単位形式が可能であり、ストアがサポートする言語の数に応じて、1 つ以上の数量単位形式の記述が可能です。

Business あるストア中で定義されている数量単位を、他のストアで使用できます。あるストアが別のストアで定義されている数量単位を使用できるようにするには、それらのストア間にタイプ com.ibm.commerce.measurement.format のストア関係を作成する必要があります。詳細については、151 ページの『第 14 章 ストア間の関係』を参照してください。

数量単位形式の説明

数量単位形式記述 は、ある数量を (表示目的で) 特定言語の特定の数量単位で形式設定する方法を記述します。

数値の使用法

数値の使用法 では、アプリケーションにおいて数値を使用する方法を定義します。たとえば、WebSphere Commerce コードで数値の使用法を使用することによって、数値 (通貨または数量) のフォーマットまたは四捨五入方法を選択できます。これら

のコード (NUMBRUSG テーブルで定義される) によって、 CURFORMAT、CURFMTDESC、QTYFORMAT、および QTYFMTDESC にある、そのタイプの数値の使用法に対して指定された規則に従って、数値をフォーマットすることができます。これによりストアは、さまざまな状態の要件を満たすように、さまざまな方法で数値をフォーマットすることができます。

WebSphere Commerce Server の計測単位資産の構造の詳細については、WebSphere Commerce オンライン・ヘルプの『数量単位データ・モデル』を参照してください。

WebSphere Commerce での計測単位の作成

計測単位は、インスタンスが作成されるときに、 WebSphere Commerce Server データベースに事前に読み込まれます。詳細については、 125 ページの『第 11 章 サイト資産』を参照してください。

また、ストアで使用する新しい計測単位を WebSphere Commerce に定義したり、ストアで使用しないことにした計測単位を削除したりすることもできます。

ストアで使用する新しい計測単位を定義するには、次のデータベース・テーブルに情報を追加します。

- QTYUNIT
- QTUNITDSC
- QTYFORMAT
- QTYFMTDESC
- QTYUNITMAP
- QTYCONVERT

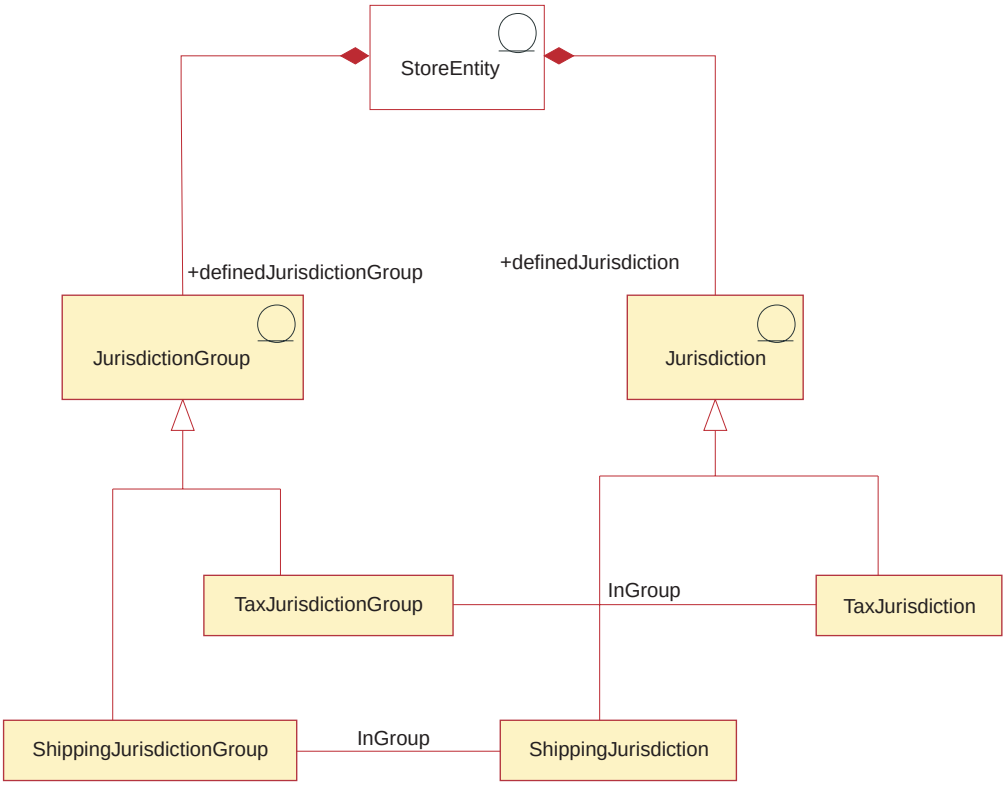
第 25 章 管轄区域資産

管轄区域 とは、商品を販売する国、都道府県、または郵便番号の範囲を表す地理的領域のことです。複数の管轄区域をグループにまとめて、*管轄区域グループ* にすることもできます。

管轄区域グループは、オーダーについての配送料と課税額の計算で使用されます。つまり、管轄区域グループは、配送料と税額計算のために使用するルールを限定するために使用できます。それら限定された計算ルールは、その計算ルールに対応する管轄区域グループ内のいずれかの管轄区域に含まれる住所にアイテムを出荷する場合にのみ、アイテムに適用されます。それで、配送料と税額の計算方法は、オーダーに含まれるさまざまなアイテムの配送先住所に応じて異なる場合があります。

WebSphere Commerce の管轄区域資産について

以下の図は、WebSphere Commerce Server に管轄区域および管轄区域グループが組み込まれている方法を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です情報モデルの詳細については、111 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、487 ページの『付録 A. UML の凡例』を参照してください。

WebSphere Commerce の場合、管轄区域または管轄区域グループはストアの一部であり、作成対象となるストアまたはストア・グループ専用になります。例えば、ストアに 3 つの管轄区域を作成してからそのストアを削除すると、管轄区域も削除されます。これらの管轄区域は、他の既存のストアや、今後作成するかもしれないストアで使用することはできません。

しかし、あるストア・グループに関して管轄区域を作成した場合、そのグループ内のストアが削除されても、それらの管轄区域は削除されません。そのストア・グループ内で新規に作成されるストアでは、それらの管轄区域が使用可能です。

WebSphere Commerce は、配送管轄区域と課税管轄区域という、2 つのタイプの管轄区域をサポートしています。複数の配送管轄区域をグループにまとめることにより、配送料の計算ルールを限定するための配送管轄区域グループにすることもできます。同じように、複数の課税管轄区域をグループにまとめることにより、税額の計算ルールを限定するための課税管轄区域グループにすることができます。

WebSphere Commerce Server の管轄区域資産の構造の詳細については、WebSphere Commerce オンライン・ヘルプの『管轄区域データ・モデル』を参照してください。

WebSphere Commerce での管轄区域資産の作成

課税額と配送料を適用するためには、ストアに管轄区域資産を作成する必要があります。管轄区域の作成の詳細については、284 ページの『WebSphere Commerce での税資産の作成』、または 265 ページの『WebSphere Commerce での配送資産の作成』を参照してください。

ストアに管轄区域を作成したら、 WebSphere Commerce アクセラレーター上のストア・ツール中の「税」および「配送」ノートブックを使用して、管轄区域を編集したり、新しい管轄区域を作成することができます。

注: 作成されたすべての管轄区域に関する管轄区域グループが、自動的に作成されます。ストアの管轄区域は作成できますが、ストア・グループに対する管轄区域は作成できません。

第 26 章 配送資産

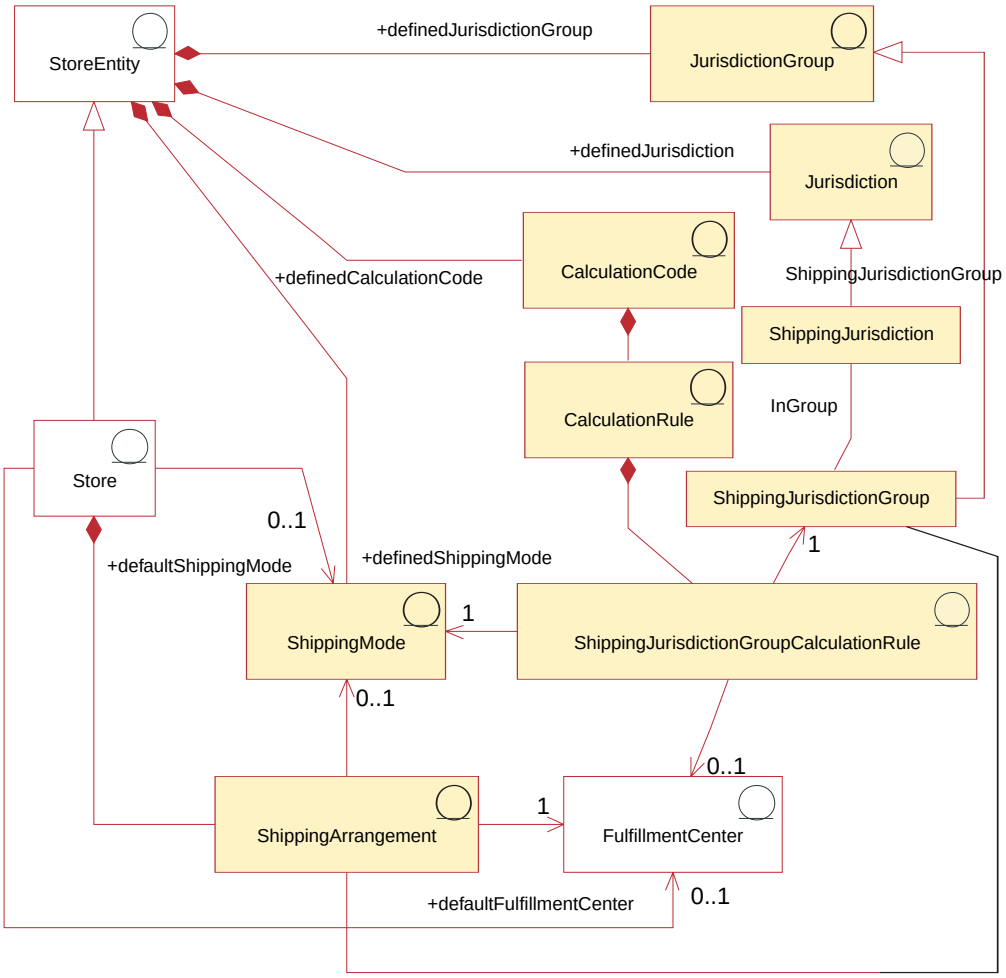
配送とは、ストアが顧客へ商品を物理的に配達する方法のことです。多くの場合、商品は配送センター (ストアの商品の保管に責任のある別個の代理店) から配送されます。

配送サービスの提供と課金を行うには、WebSphere Commerce を使用して作成するストアに、以下のものを組み込まなければなりません。

- 少なくとも 1 つの配送モード
- 少なくとも 1 つの配送量計算コード
- 管轄区域および管轄区域グループ

WebSphere Commerce の配送資産について

以下の図は、WebSphere Commerce Server 内での配送の構造を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、111 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、487 ページの『付録 A. UML の凡例』を参照してください。

配送モード

配送モード とは、商品を配送する方法のことです。詳しく説明すると、配送モードとは、運送会社 (配送センターから顧客への配送サービスを提供する会社) とその運送会社が提供する配送サービスの組み合わせのことです。たとえば、ABC 運輸の翌日配送サービスや、 ABC 運輸の速達は、配送モードの一例です。

配送モードは、ストア・エンティティーの一部です。ストア・エンティティーを削除すると、そのストア・エンティティー内で定義されている配送モードも削除されます。ストアのデフォルト配送モードは必須ではありませんが、設定することをお勧めします。

配送調整

配送調整 は、ストアと配送センターの間の調整で、配送センターは指定された配送モードを使用して特定のストアに商品を配送するように指定します。配送調整には、特定の制限を課すことができます。それには、配送調整の有効期間や配送管轄区域が含まれます。

配送モードに配送調整が関連付けられている場合、それはその配送モードだけに適用されます。それ以外の場合、配送調整は可能なすべての配送モードに適用されます。配送調整はストアの一部なので、ストアを削除すると共に削除されます。

計算コード

計算コード は配送料の計算に使用されます。つまり、配送料の計算コードはオーダー・アイテムの配送料が計算される方法を示します。オーダー・アイテムの配送料を計算するためには、カタログ・エントリーまたはカタログ・エントリー・グループのいずれかに対して、配送料計算コードを割り当てる必要があります。

計算コードは、ストア・エンティティーの一部です。 1 つの計算コードは 1 つのストア・エンティティーだけと関連付けることができますが、 1 つのストア・エンティティーに複数の計算コードがある場合があります。ストア・エンティティーを削除すると、そのストア・エンティティーに関連した計算コードも削除されます。

計算コードの使用の詳細については、「*WebSphere Commerce 計算フレームワーク・ガイド*」を参照してください。

計算ルール

各計算コードには*計算ルール* のセットがあります。あるオーダー・アイテムの配送料は、配送モード、配送センター、および配送管轄区域に応じて異なる場合があります。 ShippingJurisdictionGroupCalculationRules は、各オーダー・アイテムで使用する計算ルールを決定するために、配送計算ルールと、管轄区域、配送センター、および配送モードとを関連付ける関係オブジェクトです。

計算ルール、または ShippingJurisdictionGroupCalculationRules によって参照されるその他のオブジェクトのいずれかを削除すると、 ShippingJurisdictionGroupCalculation ルールも削除されます。

計算コードの使用の詳細については、「*WebSphere Commerce 計算フレームワーク・ガイド*」を参照してください。

管轄区域および管轄区域グループ

管轄区域 とは、商品を販売する国、都道府県、または郵便番号の範囲を表す地理的領域のことです。管轄区域をグループ化すると、*管轄区域グループ* になります。

WebSphere Commerce は、配送管轄区域と課税管轄区域という、2 つのタイプの管轄区域をサポートしています。どちらの管轄区域もそれぞれ対応するグループの一部になります。たとえば、配送管轄区域は配送管轄区域グループの一部になり、課税管轄区域は課税管轄区域グループの一部になります。

管轄区域グループは、計算ルールに関連付けられます。計算ルールは、計算の一部に管轄区域グループを使用して、配送料の金額を決定します。

管轄区域と管轄区域グループは、ストア・エンティティーの一部です。ストア・エンティティーを削除すると、そのストア・エンティティーに関連した管轄区域と管轄区域グループも削除されます。

1 つの配送先住所が、複数の配送管轄区域に関係する場合もあります。たとえば、東京内の配送先住所は、「東京、日本」、「日本」、および「世界」の各配送管轄区域に適用されます。 1 つの配送先住所が複数の配送管轄区域に適用される場合、複数の配送計算ルールが適用できることになります。そのような場合、対応する ShippingJurisdictionGroupCalculationRules の優先順位を使用することによって、使用されるルールが決定されます。

WebSphere Commerce Server の配送資産の構造の詳細については、WebSphere Commerce オンライン・ヘルプで『配送データ・モデル』を参照してください。

WebSphere Commerce での配送資産の作成

WebSphere Commerce アクセラレーターの配送ツールを使用すると、特定の配送資産 (配送モードや管轄区域など) の作成と編集ができます (すべての配送資産の作成と編集ができるわけではありません)。以下のリストは、配送ツールによって編集可能なデータベース・テーブルについて詳細に述べています。

- CALCODE
- CALCODEDSC
- CALRULE
- SHPJCRULE
- CRULESCALE
- CALSCALE

- CALSCALEDS
- CALRANGE
- CALRLOOKUP
- SHIPMODE
- SHPMODEDSC
- SHPARRANGE
- SHPARJURGP
- JURST
- JURSTGROUP
- JURSTGPREL
- CATENCALCD
- CATGPCALCD

また、配送資産を XML ファイルの形式で作成することもできます。ローダー・パッケージを使用してこれをデータベースにロードできます。したがって、配送資産を作成する選択肢には以下の 2 つがあります。

- WebSphere Commerce に付属のサンプル・ストアの 1 つの、既存の配送資産を新規作成または編集する。
- 新しい配送資産を XML ファイルの形式で作成する。

WebSphere Commerce アクセラレーターを使用した配送資産の作成および編集については、WebSphere Commerce オンライン・ヘルプを参照してください。新しい配送資産を XML ファイルの形式で作成することについては、 『XML ファイルを使用した配送資産の作成』を参照してください。

XML ファイルを使用した配送資産の作成

ローダー・パッケージを使用してデータベースにロードできる XML ファイルの形式で配送資産を作成します。ローダー・パッケージの詳細については、 363 ページの『第 10 部 ストアの発行』を参照してください。グローバル・ストアを作成する場合は、ストアでサポートされているロケールごとに別々の XML ファイルを作成することもできます。説明情報はすべてロケール固有のファイルに指定されるので、これを翻訳するのは容易です。グローバル・ストアの作成の詳細については、337 ページの『第 34 章 グローバリゼーション』を参照してください。

サンプル・ストア (以下のタスクの多くの例がサンプル・ストアから取られています) では、翻訳の不要な情報はすべて 1 つの `shipping.xml` ファイルに指定されており、翻訳の必要な情報は、そのストアがサポートするロケールごとに別の `shipping.xml` ファイルに指定されています。ロケール固有のファイルにはすべての説明情報が含まれているので、これを翻訳するのは容易です。

XML ファイルを使用してストアの配送資産を作成するには、以下のようにします。

1. 「*WebSphere Commerce 計算フレームワーク・ガイド*」に記載されている情報を確認します。 WebSphere Commerce 計算フレームワークは、顧客が購入のために選択した商品やサービスに関連した金額 (送料など) を計算します。
2. 429 ページの『第 38 章 WebSphere Commerce データベース資産グループのロード』に記載されている情報を確認します。

3. サンプル・ストアの配送資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。個々のサンプル・ストアには複数の shipping.xml ファイルが組み込まれており、これらのファイルには配送情報が含まれています。ストア・アーカイブ・ファイルは以下のディレクトリーにあります。
   - *WC_installdir*/samplestores

     注: 「*WebSphere Commerce サンプル・ストア・ガイド*」には、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

   ストア・アーカイブの shipping.xml ファイルを表示するには、 ZIP プログラムを使用してこれを解凍します。 shipping.xml ファイルはデータ・ディレクトリーにあります。言語ごとの shipping.xml は、データ・ディレクトリーのロケールごとのサブディレクトリーにあります。
4. 489 ページの『付録 B. データの作成』に記載されている情報を確認します。
5. サンプル・ストア・アーカイブの shipping.xml ファイルの 1 つをコピーするか、新しいファイルを作成することにより、shipping.xml ファイルを作成します。詳細については、wcs.dtd ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。
   - *WC_installdir*/schema/xml
6. 商品やサービスの配送先の管轄区域と管轄区域グループを定義します。管轄区域はすべて管轄区域グループに属していなければなりません。
   a. 以下の例を参考にして、XML ファイルの JURSTGROUP テーブルに管轄区域グループを定義します。

      ```
      <jurstgroup
      jurstgroup_id="@jurstgroup_id_1"
      description="Jurisdiction Group1 for Shipping"
      subclass="1"
      storeent_id="@storeent_id_1"
      code="World"/>
      ```

      ここで、
      - jurstgroup_id は、生成される固有キーです。
      - description は、管轄区域グループを管理するユーザー・インターフェースでの表示に適した、管轄区域グループの簡略説明です。
      - subclass は、以下の管轄区域グループのサブクラスです。
        - 1 = ShippingJurisdictionGroup
        - 2 = TaxJurisdictionGroup
      - storeent_id は、この管轄区域グループに関連したストア・エンティティーです。
      - code は、そのストア・エンティティーおよびサブクラスとともに、この管轄区域グループを固有に識別するコードです。

   b. 以下の例を参考にして、XML ファイルの JURST テーブルに管轄区域を定義します。

      ```
      < jurst
      ```

```
jurst_id="@jurst_id_1"
storeent_id="@storeent_id_1"
code="World"
subclass="1"/>
```

ここで、

- jurst_id は、生成される固有キーです。
- storeent_id は、この管轄区域グループに関連したストア・エンティティーです。
- code は、そのストア・エンティティーおよびサブクラスとともに、この管轄区域グループを固有に識別するコードです。
- subclass は、以下の管轄区域のサブクラスです。
  - 1 = ShippingJurisdiction
  - 2 = TaxJurisdiction

c. 以下の例を参考にして、JURSTGRPREL テーブルに情報を追加し、ステップ b で作成した管轄区域とステップ a で定義した管轄区域グループを関連付けます。

```
<jurstgprel
jurst_id="@jurst_id_1"
jurstgroup_id="@jurstgroup_id_1"
subclass="1"/>
```

ここで、

- jurst_id は管轄区域です。
- jurstgroup_id は管轄区域グループです。
- subclass は、管轄区域のサブクラスと管轄区域グループのサブクラスです。これらは一致していなければなりません。
  - 1 = ShippingJurisdiction[Group]
  - 2 = TaxJurisdiction[Group]

d. ストアでサポートされているすべての管轄区域と管轄区域グループについて、ステップ a ～ c を繰り返します。

7. ストアで使用される配送モードを定義します。

a. 次の例を参考にして、XML ファイルの SHIPMODE テーブルに配送モードを定義します。

```
<shipmode
shipmode_id="@shipmode_id_1"
field1
storeent_id="@storeent_id_1"
code="Ground 1 week"
carrier="XYZ Carrier"/>
```

ここで、

- shipmode_id は、生成される固有キーです。
- field1 は、カスタマイズの可能なフィールドです。

- storeent_id は、この配送モードに関連したストア・エンティティーです。
- code は、ストア・エンティティーごとに固有の、マーチャントが割り当てたコードです。
- carrier は、運送会社の名前または ID です。

b. 次の例を参考にして、配送モードに関する情報を SHPMODEDSC テーブルに追加します。多文化ストアを作成する場合は、ロケール固有の XML ファイルにこの情報を組み込む必要があります。

```
< shpmodedsc
description="International mail"
field1="USD$5.00 per order plus USD$1.00 for each item"
field2="5 business days"
shipmode_id="@shipmode_id_1"
language_id="&en_US;"/>
```

ここで、

- description は、顧客が選択するために表示するのに適した ShippingMode の簡略説明です。
- field1 および field2 は、カスタマイズの可能なフィールドです。
- shipmode_id は、生成される固有キーです。
- language_id は、使用される言語です。

c. ストアの中のすべての配送モードについて、ステップ a と b を繰り返します。

8. ストアで使用される計算コードを定義します。

a. 以下の例を参考にして、XML ファイルの CALCODE テーブルに計算コードを定義します。

```
< calcode
calcode_id="@calcode_id_1"
code="shipping Code 1- per/order"
calusage_id="-2"
storeent_id="@storeent_id_1"
groupby="0″
published="1"
sequence="+0.00E+000"
calmethod_id="-23"
calmethod_id_app="-24"
calmethod_id_qfy="-22"
flags="0" />
```

ここで、

- calcode_id は、生成される固有キーです。
- code は、この CalculationCode を固有に識別する文字ストリングであり、特定の CalculationUsage および StoreEntity が与えられています。

- calusage_id は、この CalculationCode がどのような種類の計算に使用されるかを示します。たとえば、CalculationCode は次の通貨金額の 1 つを計算するために使用することができます。
  - 割引額 (-1)
  - 配送料 (-2)
  - 消費税 (-3)
  - 配送税 (-4)
  - クーポン (-5)
- storeent_id は、この計算コードに関連したストア・エンティティーです。
- groupby は、計算時に OrderItems をグループ化する方法を CalculationCodeCombineMethod に指示するビット・フラグです。 0 = グループ化しない。適用可能なすべての OrderItems を 1 つのグループに入れる。詳細については、 WebSphere Commerce オンライン・ヘルプの『CALCODE テーブル: 詳細』を参照してください。
- published は、計算コードが発行されるかどうかを指定します。
  - 0 = 発行されない (一時的に使用不可)
  - 1 = 発行される
  - 2 = 削除のマーク (発行されない)
- sequence は、CalculationCodes が計算され、低位から高位の順で適用されるよう定義します。 2 つの計算コードのシーケンス番号が同じなら、calcode_id の小さい計算コードから順に計算されます。
- calmethod_id は、この CalculationCode の通貨金額を計算する方法を定義する CalculationCodeCalculateMethod です。 calmethod_id="-23" は、配送用の CalculationCodeCalculateMethod です。 WebSphere Commerce に付属している配送計算メソッドはこの 1 つだけです。
- calmethod_id_app は、関連する OrderItems の計算済みの金額を保管する CalculationCodeApplyMethod です。 calmethod_id_app="-24" は、配送用の CalculationCodeApplyMethod です。 WebSphere Commerce に付属している配送適用メソッドはこの 1 つだけです。
- calmethod_id_qfy は、この CalculationCode と関連した OrderItems を定義する CalculationCodeQualifyMethod です。 calmethod_id_qfy="-22" は、配送用の CalculationCodeQualifyMethod です。 WebSphere Commerce に付属している配送限定メソッドはこの 1 つだけです。
- flags は、この CalculationCode の CalculationCodeQualifyMethod が呼び出されるかどうかを指定します。
  - 0 = 制限なし。メソッドは呼び出されません。
  - 1 = 制限あり。メソッドは呼び出されます。

b. 以下の例を参考にして、 XML ファイルの CALCODEDSC テーブルに計算コードの説明情報を追加します。グローバル・ストアを作成する場合は、ロケール固有の XML ファイルにこの情報を組み込む必要があります。

```
<calcodedsc
 calcode_id="@calcode_id_3"
 description="5.00USD per order"
```

```
 language_id="&en_US"
 longdescription= "This shipping calculation code charges
5.00USD per order."
 />
```

ここで、

- calcode_id は、この情報が適用される計算コードです。
- description は、計算コードの簡略説明です。
- language_id は、この情報が適用される言語です。
- longdescription は、計算コードの詳細記述です。

c. ストアの中で使用される計算コードごとに、ステップ a と b を繰り返します。

9. ストアの計算ルールを定義します。

a. 以下の例を参考にして、XML ファイルの CALRULE テーブルで計算ルールをセットアップします。

```
<calrule
calrule_id="@calrule_id_1"
calcode_id="@calcode_id_1"
startdate="1900-01-01 00:00:00.000000"
enddate="2100-01-01 00:00:00.000000"
sequence="+1.00000000000000E+000"
combination="2"
calmethod_id="-27"
calmethod_id_qfy="-26"
flags="1"
identifier="1" />
```

ここで、

- calrule_id は、生成される固有の ID です。
- calcode_id は、この計算ルールを含む計算コードです。
- startdate は、この計算ルールが有効になる時刻です。
- enddate は、この計算ルールの停止が有効になる時刻です。
- sequence は、この計算ルールが処理される順序です。同じ計算コードの計算ルールは低位値から高位値の順序で処理されます。
- combination は、デフォルトの CalculationRuleCombineMethod インプリメンテーションによって実行される、特殊な処理を示す以下のビット・フラグを示します。詳細については、WebSphere Commerce オンライン・ヘルプの『CALRULE テーブル』を参照してください。
- calmethod_id は、一連の OrderItems の通貨結果を計算する CalculationRuleCalculateMethod です。
- calmethod_id_qfy は、 CalculationRuleCalculateMethod に送信する OrderItems のセットを決定する CalculationRuleQualifyMethod です。
- flags は、この計算ルールを他の計算ルールと結合する方法を決定するために CalculationRuleCombineMethod によって使用されます。詳細については、『CALRULE テーブル』を参照してください。

- identifier は、この計算ルールとその計算コードを組み合わせて識別します。

詳細については、WebSphere Commerce オンライン・ヘルプの『CALRULE テーブル』を参照してください。

b. ストアの中で使用される計算ルールごとに、ステップ a を繰り返します。個々の計算コードに複数の計算ルールがある場合もあるので注意してください。たとえば、calcode_id="@calcode_id_1" を、複数の calrule_ids と関連付けることができます。

10. ストアの計算スケールを定義します。

計算スケールは、計算に適用される範囲のセットです。たとえば、配送料金の場合、それぞれが料金に対応する重量範囲のセットがある場合があります。つまり、重量が 0〜5 kg の商品の配送料は $10.00、重量が 5〜10 kg の商品の配送料は $15.00、などです。これらの範囲がスケールを構成します。

a. 以下の例を参考にして、XML ファイルの CALSCALE テーブルで計算スケールをセットアップします。

```
<calscale
calscale_id="@calscale_id_1"
code="Scale Code 1 per order USD"
storeent_id="@storeent_id_1"
calusage_id="-2"
setccurr="USD"
calmethod_id="-28"/>
```

ここで、

- calscale_id は、生成される固有の ID です。
- code は、この計算スケールを固有に識別する文字ストリングであり、特定の計算方法およびストア・エンティティーが与えられています。
- storeent_id は、この計算スケールを含むストア・エンティティーです。
- calusage_id は、この CalculationScale がどのような種類の計算に使用されるかを示します。たとえば、CalculationScale は次の通貨金額の 1 つを計算するために使用することができます。
  - 割引額 (-1)
  - 配送料 (-2)
  - 消費税 (-3)
  - 配送税 (-4)
  - クーポン (-5)
- setccurr が指定されると、これはこの計算スケールの計算範囲オブジェクトの範囲開始値の通貨を示します。 CalculationScaleLookupMethod はこの通貨で「ルックアップ番号」を戻すことになります。
- calmethod_id は、一連のオーダー・アイテムが指定された CalculationScaleLookupMethod です。通貨金額を計算するために計算スケ

ールで使用できるルックアップ値、基本通貨値、結果乗数、および正確な重量のセットを決定します。使用する CalculationScaleLookupMethod を決定するには、以下のようにします。

- WebSphere Commerce オンライン・ヘルプの『CALMETHOD テーブル』を参照します。 SUBCLASS 列の説明を参照してください。『CALMETHOD テーブル: 詳細』のリンクをクリックします。このテーブルには、使用できる計算メソッドのタイプがリストされています。MonetaryCalculationScaleLookupMethod メソッドは 9 です。
- ブートストラップ・ファイル `wcs.bootstrap_xx_XX.xml` を開きます (`xx_XX` はロケールのコード)。ブートストラップ・ファイルは以下のディレクトリーに置かれています。
  - *WC_installdir*/schema
- 使用できる計算メソッドをリストしているセクション (CALMETHOD) を見つけます。
- calusage_ID が税 (消費税の場合は -3、配送税の場合は -4) の計算メソッドを見つけます。
- サブクラスが 7 の計算メソッドを見つけます。これはいくつか存在します。必要に応じて 1 つ選択してください。

詳細については、WebSphere Commerce オンライン・ヘルプの『CALSCALE テーブル』を参照してください。

b. ストアで使用される計算スケールごとに、ステップ a を繰り返します。たとえば、配送に関し、FashionFlow は、オーダー基準単価のスケールとアイテム基準単価のスケールを作成します。

11. 計算スケールの計算範囲を定義します。

a. 以下の例を参考にして、XML ファイルの CALRANGE テーブルで計算範囲をセットアップします。

```
<calrange
calrange_id="@calrange_id_1"
calscale_id="@calscale_id_1"
calmethod_id="-33"
rangestart="0.00000"
cumulative="0"/>
```

ここで、

- `calrange_id` は、生成される固有の ID です。
- `calscale_id` は、この計算範囲を含む計算スケールです。
- `calmethod_id` は、 CalculationRangeLookupResult からの通貨金額を決定する CalculationRangeMethod です。たとえば、FixedAmountCalculationRangeCmd、 PerUnitAmountCalculationRangeCmd、または PercentageCalculationRangeCmd。 CalculationRangeMethod を決定するには、以下のようにします。
  - WebSphere Commerce オンライン・ヘルプの『CALMETHOD テーブル』を参照します。 SUBCLASS 列の説明を参照してください。『CALMETHOD テーブル: 詳細』のリンクをクリックします。このテ

ーブルには、使用できる計算メソッドのタイプがリストされています。CalculationRangeMethod は 10 です。

– ブートストラップ・ファイル wcs.bootstrap_xx_XX.xml を開きます (xx_XX はロケールのコード)。ブートストラップ・ファイルは以下のディレクトリーに置かれています。

  - *WC_installdir*/schema

– 使用できる計算メソッドをリストしているセクション (CALMETHOD) を見つけます。

– calusage_ID が税 (消費税の場合は -3、配送税の場合は -4) の計算メソッドを見つけます。

– サブクラスが 9 の計算メソッドを見つけます。これはいくつか存在します。必要に応じて 1 つ選択してください。

• cumulative は、以下の有効値です。

– 0 = 最高の RANGESTART 値に一致する CalculationRange だけが使用される。

– 1 = 一致する CalculationRanges はすべて使用される。計算された通貨金額が合計されて、最終結果が得られます。

詳細については、WebSphere Commerce オンライン・ヘルプの『CALRANGE テーブル』を参照してください。

b. ストアの中で使用される計算スケールに関連した計算範囲ごとに、ステップ a を繰り返します。

12. 計算スケールの計算ルックアップ値を定義します。計算ルックアップ値は、計算スケールに関連した値です。たとえば、計算スケールに、以下のような配送の重量範囲と関連価格が組み込まれているとします。

• 0 ～ 5 kg の料金は 10 ドル

• 5 ～ 10 kg の料金は 15 ドル

この場合、ルックアップ値は 10 ドルと 15 ドルです。

a. 以下の例を参考にして、 XML ファイルの CALRLOOKUP テーブルで計算ルックアップ値をセットアップします。多文化ストアを作成する場合は、ロケール固有の XML ファイル、つまりストアでサポートされているロケール当たり 1 つのファイルにこの情報を組み込む必要があります。たとえば、ストアから米国と日本の顧客に配送する場合は、 1 つの XML ファイルに US ドルのルックアップ値を追加し、もう 1 つの XML ファイルに日本円のルックアップ値を追加する必要があります。

```
<calrlookup
calrlookup_id="@calrlookup_id_1"
setccurr="USD"
calrange_id="@calrange_id_1"
value="5.00"/>
```

ここで、

• calrlookup_id は、生成される固有の ID です。

• calrange_id は、計算範囲ルックアップ結果を含む計算範囲です。

- value は、計算範囲ルックアップ結果の値です。この値は、計算範囲の計算範囲メソッドによって使用され、通貨結果が決定されます。

詳細については、WebSphere Commerce オンライン・ヘルプの『CALRLOOKUP テーブル』を参照してください。

b. ストアで使用される計算スケールに関連したルックアップ値ごとに、ステップ a を繰り返します。

13. 計算ルールと計算スケールを関連付けます。

a. 以下の例を参考にして、XML ファイルの CRULESCALE テーブルで計算スケールと計算ルールを関連付けます。

```
< crulescale
calrule_id="@calrule_id_1"
calscale_id="@calscale_id_1" />
```

ここで、

- calrule_id は計算ルールです。
- calscale_id は計算スケールです。

b. 関連付ける計算スケールとルールごとに、ステップ a を繰り返します。

@ と **&** の使用法の詳細については、 489 ページの『付録 B. データの作成』を参照してください。

配送フルフィルメント資産の作成

ストアで配送資産が正しく機能するためには、ストアで使用される配送管轄区域グループと計算ルール、および配送センターと配送モードを関連付けなければなりません。

配送資産を配送センターに関連付けるには、その前にフルフィルメント資産を作成しなければなりません。フルフィルメント資産の作成の詳細については、 228 ページの『WebSphere Commerce での配送資産の作成』を参照してください。

フルフィルメント資産を作成し終えたら、 SHPJCRULE テーブルと SHPARRANGE テーブルに情報を追加して、配送資産を関連付けます。以下のようにします。

1. 「*WebSphere Commerce 計算フレームワーク・ガイド*」に記載されている情報を確認します。 WebSphere Commerce 計算フレームワークは、顧客が購入のために選択した商品やサービスに関連した金額 (送料など) を計算します。
2. 429 ページの『第 38 章 WebSphere Commerce データベース資産グループのロード』に記載されている情報を確認します。
3. サンプル・ストアの配送フルフィルメント資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。個々のサンプル・ストアには 1 つの shipfulfill.xml ファイルが組み込まれており、このファイルには配送フルフィルメント情報が組み込まれています。ストア・アーカイブの shipfulfill.xml ファイルを表示するには、 ZIP プログラムを使用してこれを解凍します。 shipfulfill.xml ファイルは、データ・ディレクトリーにあります。

ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

- *WC_installdir*/samplestores

注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

4. 489 ページの『付録 B. データの作成』に記載されている情報を確認します。
5. サンプル・ストア・アーカイブの shipfulfill.xml ファイルの 1 つをコピーするか、または新しいファイルを作成することにより、shipfulfill.xml ファイルを作成します。詳細については、wcs.dtd ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。
   - *WC_installdir*/schema/xml
6. SHPJCRULE テーブルに情報を追加して、計算ルールと配送管轄区域グループを関連付けます。次の例を参考にしてください。多文化ストアを作成する場合は、ストアでサポートされているロケールごとに 1 つずつ XML ファイルを作成する必要もあります。

```
<shpjcrule
calrule_id="@calrule_id_1"
ffmcenter_id="@ffmcenter_id_1"
jurstgroup_id="@jurstgroup_id_1"
precedence="0"
shipmode_id="@shipmode_id_1"
shpjcrule_id="@shpjcrule_id_1"
```

   ここで、

   - calrule_id は、使用される計算ルールです。
   - ffmcenter_id は、配送センターです。これが NULL の場合、この関連はすべての配送センターに適用されます。
   - jurstgroup_id は配送管轄区域グループです。これが NULL の場合、この関連はすべての配送管轄区域グループに適用されます。
   - precedence は、配送先住所が、同一の配送センターと配送モードのために指定された複数の配送管轄区域グループに属する場合に、SHPJCRULE.PRECEDENCE 値が最大の計算ルールにのみ限定されることを示します。
   - shipmode_id は配送モードです。
   - shpjcrule_id は、生成される固有の ID です。
7. ストアの管轄区域グループ、配送センター、およびルールの関連ごとに、ステップ 3 を繰り返します。
8. SHPARRANGE テーブルに情報を追加して、配送モードと配送センターをストアに関連付けます。次の例を参考にしてください。

```
<shparrange
 shparrange_id="@shparrange_id_2"
 store_id="@storeent_id_1"
 ffmcenter_id="@ffmcenter_id_1"
 shipmode_id= "@shipmode_id_2"
 startdate="1970-06-22 23:00:00.000000"
 enddate= "2008-06-22 23:00:00.000000"
 precedence= "0"
 flags="0"
/>
```

ここで、

- shparrange_id は、生成される固有の ID です。
- store_id は、ストアです。
- ffmcenter_id は、配送センターです。
- shipmode_id は配送モードです。 NULL は、配送モードに関係なくこの配送調整を使用できることを示します。
- startdate は、この配送調整の開始が有効になる時刻です。
- enddate は、この配送調整の停止が有効になる時刻です。
- precedence は、特定の時点に複数の配送調整が (同じストアおよび配送モードに対して) 有効であった場合には、 PRECEDENCE が一番高い配送調整が使用されることを示します。
- flags は、以下のビット・フラグを含みます。
  - 1 = 制限あり - この配送調整は、この配送調整と (テーブル SHPARJURGP を介して) 関連付けられた配送管轄区域グループのいずれかと一致する住所を持つオーダー・アイテムにのみ適用されます。

9. ストアで使用されるすべての配送モードについて、ステップ 5 を繰り返します。

@ と **&** の使用法の詳細については、 489 ページの『付録 B. データの作成』を参照してください。

ストア - カタログ - 配送資産の作成

配送モードとストアを関連付けるには、ストアに組み込まれている契約ごとに、ストアの計算コードとカタログ・エントリーを関連付けなければなりません。

ストア - カタログ - 配送資産を作成するには、その前にストア資産とカタログ資産を作成しなければなりません。ストア資産の作成の詳細については、 145 ページの『XML ファイルによるストア・データ資産の作成』を参照してください。カタログ資産の作成の詳細については、 187 ページの『ストア・カタログ資産の表示』を参照してください。

ストア - カタログ - 配送資産を作成するには、以下のようにします。

1. 「*WebSphere Commerce 計算フレームワーク・ガイド*」に記載されている情報を確認します。 WebSphere Commerce 計算フレームワークは、顧客が購入のために選択した商品やサービスに関連した金額 (送料など) を計算します。
2. 429 ページの『第 38 章 WebSphere Commerce データベース資産グループのロード』に記載されている情報を確認します。
3. サンプル・ストアの配送フルフィルメント資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。

   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

   - *WC_installdir*/samplestores

注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

個々のサンプル・ストアには 1 つの store-catalog-shipping.xml ファイルが組み込まれており、このファイルには配送フルフィルメント情報が組み込まれています。ストア・アーカイブの store-catalog-shipping.xml ファイルを表示するには、 ZIP プログラムを使用してこれを解凍します。
store-catalog-shipping.xml ファイルは、データ・ディレクトリーにあります。

4. 489 ページの『付録 B. データの作成』に記載されている情報を確認します。
5. サンプル・ストア・アーカイブの store-catalog-shipping.xml ファイルの 1 つをコピーするか、新しいファイルを作成することにより、
   store-catalog-shipping.xml ファイルを作成します。詳細については、wcs.dtd ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。
   - *WC_installdir*/schema/xml
6. CATENCALCD テーブルに情報を追加して、ストア - カタログ - 配送の関係を作成します。次の例を参考にしてください。

```
<catencalcd
 calcode_id="@calcode_id_1"
 catencalcd_id="@catencalcd_id_1"
 store_id="@storeent_id_1"
/>
```

ここで、

- calcode_id は計算コードです。
- catencalcd_id は、生成される固有の ID です。
- store_id は、ストアです。

@ と **&** の使用法の詳細については、 489 ページの『付録 B. データの作成』を参照してください。

デフォルト配送モードの作成

ストアのデフォルト配送モードを設定するには、情報を STOREDEF テーブルに追加しなければなりません。情報を STOREDEF テーブルに追加するには、以下のようにします。

1. 429 ページの『第 38 章 WebSphere Commerce データベース資産グループのロード』に記載されている情報を確認します。
2. サンプル・ストアのデフォルト資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。
   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。
   - *WC_installdir*/samplestores

   注: WebSphere Commerce オンライン・ヘルプには、サンプル・ストアに含まれる各データ資産についての情報が記載されています。

個々のサンプル・ストアには 1 つの store-defaults.xml ファイルが含まれており、このファイルにはデフォルトの配送情報が組み込まれています。ストア・アーカイブの store-defaults.xml ファイルを表示するには、 ZIP プログラムを使用してこれを解凍します。 store-defaults.xml ファイルは、データ・ディレクトリーにあります。

3. 489 ページの『付録 B. データの作成』に記載されている情報を確認します。
4. サンプル・ストア・アーカイブの store-defaults.xml ファイルの 1 つをコピーするか、新しいファイルを作成することにより、store-defaults.xml ファイルを作成します。詳細については、wcs.dtd ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。
   - *WC_installdir*/schema/xml
5. 次の例を参考にして、XML ファイルの STOREDEF テーブルに情報を追加して、ストアのデフォルト配送モードを指定します。

```
<storedef
   store_id="@storeent_id_1"
   shipmode_id="@shipmode_id_1"
/>
```

ここで、

- store_id は、ストアです。
- shipmode_id は、ストアのデフォルト配送モードです。

@ と **&** の使用法の詳細については、 489 ページの『付録 B. データの作成』を参照してください。

第 27 章 税資産

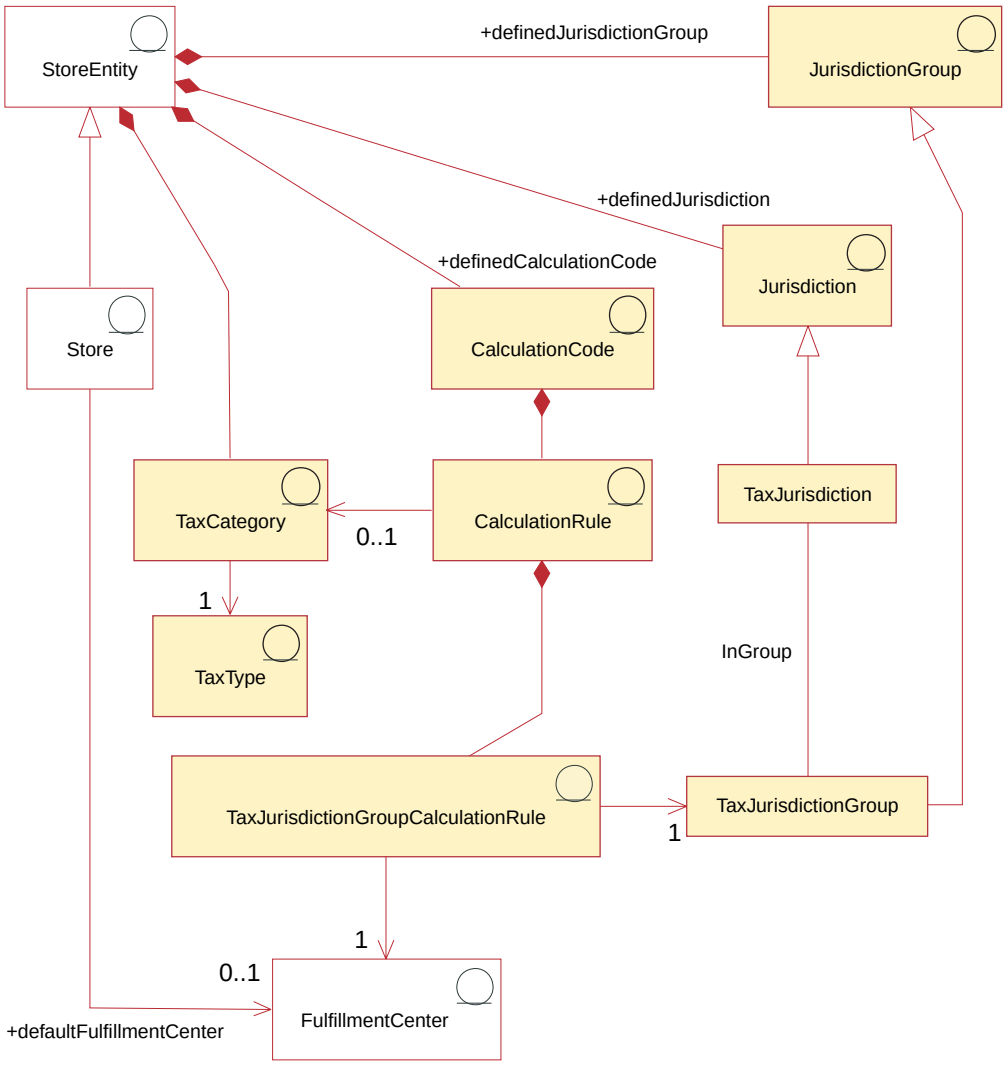
ストアに備えられている商品やサービスの税を課金したり徴収したりするには、WebSphere Commerce を使用して作成したストアに以下のものを組み込まなければなりません。

- 課税カテゴリー
- 計算コード
- 管轄区域および管轄区域グループ

課税カテゴリー、計算コード、および管轄区域と管轄区域グループの組み合わせにより、ストアの課税額が作成されます。

WebSphere Commerce の税資産について

以下の図は、WebSphere Commerce Server の課税構造を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、111 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、487 ページの『付録 A. UML の凡例』を参照してください。

課税カテゴリー

課税カテゴリー は、ストアが徴収する必要がある各種の税 (国税、都道府県税、市町村税など) に対応します。

1 つの課税カテゴリーは、1 つのストア・エンティティーの一部になります。一方、1 つのストア・エンティティーに複数の課税カテゴリーがある場合があります。ストア・エンティティーを削除すると、そのストア・エンティティーに関連した課税カテゴリーも削除されます。

税タイプ

ストアは普通、2 タイプの税を徴収します。それは、売上税または消費税と、配送税です。課税カテゴリーごとに 1 つの*税タイプ* があります。個々の課税カテゴリーの税タイプは 1 つだけですが (たとえば、税カテゴリーが国税の場合は税タイプは消費税)、数種類の課税カテゴリーが 1 つの税タイプに属する場合があります (たとえば、消費税の税タイプは、国税、都道府県税、市町村税のカテゴリーにあてはまります)。

計算コード

計算コード は課税額の計算に使用されます。つまり、税額計算コードはオーダー・アイテムの税が計算される方法を示します。オーダー・アイテムの税額を計算するためには、カタログ・エントリーまたはカタログ・エントリー・グループのいずれかに対して、消費税および配送税計算コードを割り当てる必要があります。特定のカタログ・エントリーまたはカタログ・エントリーのグループに対して、各税タイプの税額計算コードを 1 つだけ適用できます。通常は、売上税または消費税は正価に課され、配送税は配送料に課されます。

計算コードは、ストア・エンティティーの一部です。 1 つの計算コードは 1 つのストア・エンティティーだけと関連付けることができますが、 1 つのストア・エンティティーに複数の計算コードがある場合があります。ストア・エンティティーを削除すると、そのストア・エンティティーに関連した計算コードも削除されます。

計算コードを使用するための詳細については、「*IBM WebSphere Commerce 計算フレームワーク・ガイド*」を参照してください。

計算ルール

各計算コードには、少なくとも 1 つの*計算ルール* があります。これは、各課税カテゴリーの計算方法を定義し、計算実行の条件を指定するものです。個々の税額計算ルールは、1 つの課税カテゴリー、1 つの管轄区域グループ、および 1 つの配送センターに関連付けられます。それらの組み合わせにより、計算ルールが使用され

る条件が定義されます。たとえば、特定の課税カテゴリーにおける金額の計算においては、オーダーで指定される配送先住所や配送センターに応じて異なるルールが選択されることがあります。

各計算ルールは、ちょうど 1 つの計算コードに属します。

1 つの税額計算コードが複数の計算ルールを持つ場合があります。ストアに関連付けられている課税カテゴリー、課税管轄区域グループ、および配送センターの組み合わせごとに 1 つの計算ルールです。消費税および配送税の各計算ルールは、複数の TaxJurisdictionGroupCalculationRules (TaxRule) に関連付けることができます。たとえば下の表の中で、計算ルール 10001 は、管轄区域グループ 1234 と 1235 の両方に適用されます。

TAXJCRULE_ID	CALRULE_ID	FFMCENTER_ID	JURSTGROUP_ID	PRECEDENCE
10001	10001	NULL	1234	0
10002	10001	NULL	1235	0

各 TaxRule により、それぞれの計算ルールが適用される条件が定義されます。たとえば、ストアの出荷先となる管轄区域グループごとに計算ルールを定義できます。次に示す例の場合、計算ルール 10001 は管轄区域グループ 1234 と 1235 の両方に適用されます。

以下の例の税額計算コードでは、課税管轄区域がアルバータの場合は、都道府県税の消費税カテゴリーの計算ルール A が使用され、課税管轄区域がブリティッシュコロンビアの場合はルール C が使用されます。

課税管轄区域	国税の消費税	都道府県税の消費税
カナダのアルバータ	計算ルール B (Y%)	計算ルール A (X%)
カナダのブリティッシュコロンビア	計算ルール B (Y%)	計算ルール C (Z%)

配送先住所が複数の課税管轄区域グループと一致する場合、関連付けられた TAXJCRULE.PRECEDENCE 列の値が最も大きい計算ルールが使用されます。

TaxJurisdictionGroupCalculationRules (TaxRule) と計算ルールとの関連付けにより、どのような場合に計算ルールが適用されるかが決定されます。消費税または配送税の計算ルールは、 TaxRule で指定される条件のいずれか 1 つが満たされている場合に適用されます。以下の例の場合、計算ルール 10001 は、管轄区域グループ 1001 に配送する場合、または配送センター 1001 から配送する場合、または管轄区域グループ 1001 に配送する場合に適用されます。

CALRULE_ID	FFMCENTER_ID	JURSTGROUP_ID
10001	NULL	1001
10001	1001	1001

各 TaxJurisdictionGroupCalculationRule は、多くても 1 つの管轄区域グループに関連付けられます。計算ルールは、それ自体が直接管轄区域グループに関連付けられるわけではありません。

計算ルールを使用するための詳細については、「*IBM WebSphere Commerce 計算フレームワーク・ガイド*」を参照してください。

管轄区域および管轄区域グループ

管轄区域 とは、商品を販売する国、都道府県、または郵便番号の範囲を表す地理的領域のことです。管轄区域をグループ化すると、*管轄区域グループ* になります。

WebSphere Commerce は、配送管轄区域と課税管轄区域という、2 つのタイプの管轄区域をサポートしています。どちらの管轄区域もそれぞれ対応するグループの一部になります。たとえば、配送管轄区域は配送管轄区域グループの一部になり、課税管轄区域は課税管轄区域グループの一部になります。

管轄区域と管轄区域グループにより、課税額の計算に使用される計算ルールが決められます。

管轄区域と管轄区域グループは、ストア・エンティティーの一部です。個々の管轄区域や管轄区域グループは、1 つのストア・エンティティーの一部になります。しかしながら、1 つのストア・エンティティーに複数の管轄区域や管轄区域グループがある場合があります。ストア・エンティティーを削除すると、そのストア・エンティティーに関連した管轄区域と管轄区域グループも削除されます。

WebSphere Commerce Server の税資産の構造に関する詳細については、WebSphere Commerce オンライン・ヘルプで『税額データ・モデル』を参照してください。

WebSphere Commerce での税資産の作成

WebSphere Commerce アクセラレーターの税ツールを使用すると、すべての税資産ではなく特定の税資産 (課税カテゴリーや管轄区域など) を、作成して編集できます。

以下のリストは、税ツールによって編集可能なデータベース・テーブルについて詳細に述べています。

- CALCODE
- CALCODEDSC
- CALRULE
- TAXJCRULE
- CRULESCALE
- CALSCALE
- CALSCALEDS
- CALRANGE
- CALRLOOKUP
- TAXCGRY
- TAXCGRYDS
- JURST

- JURSTGROUP
- JURSTGPREL
- CATENCALCD
- CATGPCALCD

また、税資産を XML ファイルの形式で作成することもできます。ローダー・パッケージを使用してこのファイルをデータベースにロードできます。したがって、配送資産を作成する選択肢には以下の 2 つがあります。

- WebSphere Commerce に付属のサンプル・ストアの 1 つの、既存の税資産を新規作成または編集する。
- 新しい税資産を XML ファイルの形式で作成する。

既存のストア・アーカイブの税資産を編集することに関する情報や、税に関する一般的な情報については、 WebSphere Commerce オンライン・ヘルプを参照してください。新しい税資産を XML ファイルの形式で作成することに関する情報は、『XML ファイルを使用した税資産の作成』を参照してください。

XML ファイルを使用した税資産の作成

税資産を XML ファイルの形式で作成します。 XML ファイルは、ローダー・パッケージを使用して、データベースにロード可能です。ローダー・パッケージの詳細については、 363 ページの『第 10 部 ストアの発行』を参照してください。グローバル・ストアを作成する場合は、ストアでサポートされているロケールごとに別々の XML ファイルを作成することもできます。説明情報はすべてロケール固有のファイルに指定されるので、これを翻訳するのは容易です。

サンプル・ストア (以下のタスクの多くの例がサンプル・ストアから取られています) では、翻訳の不要な情報はすべて 1 つの `tax.xml` ファイルに指定されており、翻訳の必要な情報は、そのストアがサポートするロケールごとに別の `tax.xml` ファイルに指定されています。ロケール固有のファイルには、すべての説明情報が含まれています。

XML ファイルを使用してストアの税資産を作成するには、以下のようにします。

1. 429 ページの『第 38 章 WebSphere Commerce データベース資産グループのロード』に記載されている情報を確認します。「*IBMWebSphere Commerce 計算フレームワーク・ガイド*」に記載されている情報を確認します。 WebSphere Commerce 計算フレームワークは、顧客が購入のために選択した商品やサービスに関連した金額 (税金など) を計算します。
2. サンプル・ストアの税資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。

   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

   - *WC_installdir*/samplestores
3. 489 ページの『付録 B. データの作成』に記載されている情報を確認します。
4. サンプル・ストア・アーカイブの `tax.xml` ファイルの 1 つをコピーするか、または新しいファイルを作成することにより、`tax.xml` ファイルを作成します。詳細については、`tax.xml` に対応する DTD ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。

- *WC_installdir*/schema

5. 商品やサービスの配送先の管轄区域と管轄区域グループを定義します。課税管轄区域を、それが適用される課税カテゴリーの計算ルールに従って、課税管轄区域グループに割り当ててください。
   a. 以下の例を参考にして、XML ファイルの JURSTGROUP テーブルに管轄区域グループを定義します。

      ```
      <jurstgroup
      jurstgroup_id="@jurstgroup_id_2"
      description="Tax Jurstiction Group 1"
      subclass="2"
      storeent_id="@storeent_id_1"
      code="World"/>
      ```

      ここで、
      - jurstgroup_id は、生成される固有キーです。
      - description は、管轄区域グループを管理するユーザー・インターフェースでの表示に適した、管轄区域グループの簡略説明です。
      - subclass は、以下の管轄区域グループのサブクラスです。
        - 1 = ShippingJurisdictionGroup
        - 2 = TaxJurisdictionGroup
      - storeent_id は、この管轄区域グループに関連したストア・エンティティーです。
      - code は、そのストア・エンティティーおよびサブクラスとともに、この管轄区域グループを固有に識別するコードです。

   b. 以下の例を参考にして、XML ファイルの JURST テーブルに管轄区域を定義します。

      ```
      <jurst
      jurst_id="@jurst_id_2"
      storeent_id="@storeent_id_1"
      code="World"
      subclass="2"/>
      ```

      ここで、
      - jurst_id は、生成される固有キーです。
      - storeent_id は、この管轄区域グループに関連したストア・エンティティーです。
      - code は、そのストア・エンティティーおよびサブクラスとともに、この管轄区域グループを固有に識別するコードです。
      - subclass は、以下の管轄区域のサブクラスです。
        - 1 = ShippingJurisdiction
        - 2 = TaxJurisdiction

   c. 以下の例を参考にして、JURSTGRPREL テーブルに情報を追加し、ステップ b で作成した管轄区域とステップ a で定義した管轄区域グループを関連付けます。

```
<jurstgprel
jurst_id="@jurst_id_2"
jurstgroup_id="@jurstgroup_id_1"
subclass="2"/>
```

ここで、

- jurst_id は管轄区域です。
- jurstgroup_id は管轄区域グループです。
- subclass は、管轄区域のサブクラスと管轄区域グループのサブクラスです。これらは一致していなければなりません。
  - 1 = ShippingJurisdiction[Group]
  - 2 = TaxJurisdiction[Group]

d. ストアでサポートされているすべての管轄区域と管轄区域グループについて、ステップ a ～ c を繰り返します。

6. ストアで使用される課税カテゴリーを定義します。

a. 以下の例を参考にして、XML ファイルの TAXCGRY テーブルに課税カテゴリーを定義します。

```
<taxcgry
 taxcgry_id="@taxcgry_id_1"
 taxtype_id="-3"
 storeent_id="@storeent_id_1"
 name="Sales Tax"
 displayseq="0"
 displayusage="0"/>
```

ここで、

- taxcgry_id は、生成される固有キーです。
- taxtype_id="-3" は、この課税カテゴリーの税タイプです。 WebSphere Commerce では、以下の 2 つの税タイプをサポートしています。
  - 消費税または売上税 (-3)
  - 配送税 (-4)
- storeent_id は、この課税カテゴリーに関連したストア・エンティティーです。
- name は、課税カテゴリーの名前です。名前とストア・エンティティーとで、この課税カテゴリーを固有に識別します。
- displayseq は、オーダーで表示される場合の税額の順序を、低額のものから順に指定します。
- displayusage は、この課税カテゴリーを PriceDataBean との関連で以下のように指定します。
  - 0 = 計算されない
  - 1 = 計算される

  PriceDataBean を使用して、商品価格とともに表示する必要がある税金額を得ることができます。

b. ストアの中で使用される課税カテゴリーごとに、ステップ a を繰り返します。

c. 以下の例を参考にして、 XML ファイルの TAXCGRYDS テーブルに課税カテゴリーの説明情報を追加します。多文化ストアを作成する場合は、ロケール固有の XML ファイルにこの情報を組み込む必要があります。

```
<taxcgryds
 taxcgry_id="@taxcgry_id_1"
 description="Sales Tax"
 language_id="&en_US"/>
```

ここで、

- taxcgry_id は、課税カテゴリーです。
- description は、顧客に対して表示するのに適した、課税カテゴリーの簡略説明です。
- language_id は、この情報の表示に使用される言語です。

d. ストアの中で使用される課税カテゴリーごとに、ステップ c を繰り返します。

7. ストアで使用される計算コードを定義します。

a. 以下の例を参考にして、XML ファイルの CALCODE テーブルに計算コードを定義します。

```
<calcode
 calcode_id="@calcode_id_3"
 code="Tax Code 1"
 calusage_id="-3"
 storeent_id="@storeent_id_1"
 groupby="0"
 published="1"
 sequence="0"
 calmethod_id="-43"
 calmethod_id_app="-44"
 calmethod_id_qfy="-42"
 displaylevel="0"
 flags="0"
 precedence="0"
 />
```

ここで、

- calcode_id は、生成される固有キーです。
- code は、この計算コードを固有に識別する文字ストリングであり、特定の計算方法およびストア・エンティティーが与えられています。
- calusage_id は、この計算コードがどのような種類の計算に使用されるかを示します。たとえば、計算コードは次の通貨金額の 1 つを計算するために使用することができます。
  - 割引額 (-1)
  - 配送料 (-2)
  - 消費税 (-3)
  - 配送税 (-4)
  - クーポン (-5)
- storeent_id は、この計算コードに関連したストア・エンティティーです。
- groupby は、計算時にオーダー・アイテムをグループ化する方法を計算コード結合メソッドに指示するビット・フラグです。 0 を指定した場合、グループ化されません (適用可能なすべてのオーダー・アイテムが 1 つのグ

ループに属することになります)。詳細については、 WebSphere Commerce オンライン・ヘルプの『*CALCODE テーブル: 詳細*』を参照してください。

- published は、計算コードが発行されるかどうかを指定します。
  - 0 = 発行されない (一時的に使用不可)
  - 1 = 発行される
  - 2 = 削除のマーク (発行されない)
- sequence は、計算コードが計算される順序です。計算コード、低位から高位の順で計算されて適用されます。 2 つの計算コードのシーケンス番号が同じなら、 calcode_id の小さい計算コードから順に計算されます。
- calmethod_id は、この計算コードの税金額を計算する方法を定義する計算コード計算メソッドです。使用する計算コード計算メソッドを決定するには、以下のようにします。
  - WebSphere Commerce オンライン・ヘルプの『CALMETHOD テーブル』を参照します。 SUBCLASS 列の説明を参照してください。『CALMETHOD テーブル: 詳細』のリンクをクリックします。このテーブルには、使用できる CALMETHOD のタイプがリストされています。計算コード計算メソッド・タイプは 3 です。
  - ブートストラップ・ファイル wcs.bootstrap_xx_XX.xml を開きます (xx_XX はロケールのコード)。ブートストラップ・ファイルは以下のディレクトリーに置かれています。
    - *WC_installdir*/schema/xml
  - 使用できる計算メソッドをリストしているセクション (CALMETHOD) を見つけます。
  - calusage_ID が税 (消費税の場合は -3、配送税の場合は -4) の計算メソッドを見つけます。
  - サブクラスが 3 の計算メソッドを見つけます。この計算メソッドは -43 です。
- calmethod_id_app は、関連する OrderItems の計算済みの金額を保管する CalculationCodeApplyMethod です。 calmethod_id に記述されているメソッドを使用して、使用される計算コード適用メソッドを判別します。
  - calmethod_id_app="-44" は、消費税の場合の CalculationCodeApplyMethod です。
- calmethod_id_qfy は、この計算コードと関連したオーダー・アイテムを定義する CalculationCodeQualifyMethod です。 calmethod_id に記述されているメソッドを使用して、使用される計算コード限定メソッドを判別します。
  - calmethod_id_qfy="-42" は、消費税の場合の CalculationCodeQualifyMethod です。
- display level は、この計算コードが計算する金額を、以下の値で表示するかどうかを決定します。
  - 0 = オーダー・アイテム
  - 1 = オーダー
  - 2 = 商品

- 3 = アイテム
- 4 = 契約

• flags は、この計算コードの CalculationCodeQualifyMethod が呼び出されるかどうかを指定します。
- 0 = 制限なし。メソッドは呼び出されません。
- 1 = 制限あり。メソッドは呼び出されます。

b. 以下の例を参考にして、XML ファイルの CALCODEDSC テーブルに計算コードの説明情報を追加します。多文化ストアを作成する場合は、ロケール固有の XML ファイルにこの情報を組み込む必要があります。

```
<calcodedsc
 calcode_id="@calcode_id_3"
 description="Vitamins
 language_id="&en_US"
 longdescription= "In Ontario vitamins are taxed federally, but
not provincially."
 />
```

ここで、

• calcode_id は、この情報が適用される計算コードです。
• description は、計算コードの簡略説明です。
• language_id は、この情報が適用される言語です。
• longdescription は、計算コードの詳細記述です。

c. ストアの中で使用される計算コードごとに、ステップ a と b を繰り返します。

8. ストアの計算ルールを定義します。

a. 以下の例を参考にして、XML ファイルの CALRULE テーブルで計算ルールをセットアップします。

```
<calrule
 calrule_id="@calrule_id_10"
 calcode_id="@calcode_id_3"
 startdate="1900-01-01 00:00:00.000000"
 taxcgry_id="@taxcgry_id_1"
 enddate="2100-01-01 00:00:00.000000"
 flags="1"
 identifier="1"
 combination="2"
 calmethod_id="-47"
 calmethod_id_qfy="-46"
 />
```

ここで、

• calrule_id は、生成される固有の ID です。
• calcode_id は、この計算ルールを含む計算コードです。
• startdate は、この計算ルールが有効になる時刻です。
• taxcgry_id は、この計算ルールが有効な課税カテゴリーです。
• enddate は、この計算ルールの停止が有効になる時刻です。
• combination は、この計算ルールを他の計算ルールと結合する方法を決定するために CalculationRuleCombineMethod によって使用されます。詳細については、『CALRULE テーブル』を参照してください。
• identifier は、この計算ルールとその計算コードを組み合わせて識別します。

- flags は、デフォルトの CalculationRuleCombineMethod インプリメンテーションによって実行される、特殊な処理を示す以下のビット・フラグを示します。詳細については、WebSphere Commerce オンライン・ヘルプの『CALRULE テーブル』を参照してください。
- calmethod_id は、一連のオーダー・アイテムの通貨結果を計算する CalculationRuleCalculateMethod です。使用する計算ルール計算メソッドを決定するには、以下のようにします。
  - WebSphere Commerce オンライン・ヘルプの『CALMETHOD テーブル』を参照します。 SUBCLASS 列の説明を参照してください。『CALMETHOD テーブル: 詳細』のリンクをクリックします。このテーブルには、使用できる CALMETHOD のタイプがリストされています。計算ルール計算メソッドは 7 です。
  - ブートストラップ・ファイル wcs.bootstrap_xx_XX.xml を開きます (xx_XX はロケールのコード)。ブートストラップ・ファイルは以下のディレクトリーに置かれています。
    - *WC_installdir*/schema/xml
  - 使用できる計算メソッドをリストしているセクション (CALMETHOD) を見つけます。
  - calusage_ID が税 (消費税の場合は -3、配送税の場合は -4) の計算メソッドを見つけます。
  - サブクラスが 7 の計算メソッドを見つけます。この計算メソッドは -47 です。
- calmethod_id_qfy は、 CalculationRuleCalculateMethod に送信する OrderItems のセットを決定する CalculationRuleQualifyMethod です。calmethod_id に記述されているメソッドを使用して、使用される計算ルール限定メソッドを判別します。

b. ストアの中で使用される計算ルールごとに、ステップ a を繰り返します。個々の計算コードについて、適用可能な各課税カテゴリーごとに 1 つずつ、複数の計算ルールがある場合もあるので注意してください。たとえば、calcode_id="@calcode_id_1" を、複数の calrule_ids と関連付けることができます。

9. ストアの計算スケールを定義します。

   計算スケールは、計算に適用される範囲のセットです。これらの範囲がスケールを構成します。

   a. 以下の例を参考にして、XML ファイルの CALSCALE テーブルで計算スケールをセットアップします。

```
<calscale
 calscale_id="@calscale_id_19"
 code="Sales Tax 1"
 storeent_id="@storeent_id_1"
 calusage_id="-3"
 setccurr="USD"
 calmethod_id="-53"
 />
```

   ここで、

   - calscale_id は、生成される固有の ID です。

- code は、この計算スケールを固有に識別する文字ストリングであり、特定の計算方法およびストア・エンティティーが与えられています。
- storeent_id は、この計算スケールを含むストア・エンティティーです。
- calusage_id は、この CalculationScale がどのような種類の計算に使用されるかを示します。たとえば、CalculationScale は次の通貨金額の 1 つを計算するために使用することができます。
  - 割引額 (-1)
  - 配送料 (-2)
  - 消費税 (-3)
  - 配送税 (-4)
  - クーポン (-5)
- setccurr が指定されると、これはこの計算スケールの計算範囲オブジェクトの範囲開始値の通貨を示します。 CalculationScaleLookupMethod はこの通貨で「ルックアップ番号」を戻すことになります。この場合、それは指定されていません。 CalculationScaleLookupMethod は、オーダーの通貨によりルックアップ番号を戻します。スケール範囲の開始値が 0 でない場合以外は、通貨を指定する必要はありません。
- calmethod_id は、一連のオーダー・アイテムが指定された CalculationScaleLookupMethod です。通貨金額を計算するために計算スケールで使用できるルックアップ番号、基本通貨値、結果乗数、および正確な重量のセットを決定します。使用する CalculationScaleLookupMethod を決定するには、以下のようにします。
  - WebSphere Commerce オンライン・ヘルプの『CALMETHOD テーブル』を参照します。 SUBCLASS 列の説明を参照してください。『CALMETHOD テーブル: 詳細』のリンクをクリックします。このテーブルには、使用できる CALMETHOD のタイプがリストされています。 MonetaryCalculationScaleLookupMethod メソッドは 9 です。
  - ブートストラップ・ファイル wcs.bootstrap_xx_XX.xml を開きます (xx_XX はロケールのコード)。ブートストラップ・ファイルは以下のディレクトリーに置かれています。
    - *WC_installdir*/schema/xml
  - 使用できる計算メソッドをリストしているセクション (CALMETHOD) を見つけます。
  - calusage_ID が税 (消費税の場合は -3、配送税の場合は -4) の計算メソッドを見つけます。
  - サブクラスが 9 の計算メソッドを見つけます。サブクラス 9 の計算メソッドは複数あります。必要に合ったメソッドを選出します。

  詳細については、WebSphere Commerce オンライン・ヘルプの『CALSCALE テーブル』を参照してください。

b. ストアで使用される計算スケールごとに、ステップ a を繰り返します。

c. 以下の例を参考にして、 XML ファイルの CALSCALDS テーブルに計算スケールの説明情報を追加します。多文化ストアを作成する場合は、ロケール固有の XML ファイルにこの情報を組み込む必要があります。

```
<calscaleds
 calscale_id="@calscale_id_19"
 description="Sales Tax 5% "
 language_id="&en_US"
 />
```

ここで、

- calscale_id は、この記述の適用対象となる計算スケールです。
- description は、計算の実行方法を顧客に対して表示するための、計算スケールの簡略説明です。たとえば、「1 kg 当たり 0.1 ドル、最低料金 5 ドル」や、「5 個以上お買い上げの場合 10 % オフ」などです。
- language_id は、この情報の表示に使用される言語です。

d. ストアで使用される計算スケールごとに、ステップ c を繰り返します。

10. 計算スケールの計算範囲を定義します。

a. 以下の例を参考にして、XML ファイルの CALRANGE テーブルで計算範囲をセットアップします。

```
<calrange
 calrange_id="@calrange_id_37"
 calscale_id="@calscale_id_19"
 calmethod_id="-59"
 rangestart="0.00000"
 cumulative="0"
 />
```

ここで、

- calrange_id は、生成される固有の ID です。
- calscale_id は、この計算範囲を含む計算スケールです。
- calmethod_id は、 CalculationRangeLookupResult からの通貨金額を決定する CalculationRangeMethod です。たとえば、FixedAmountCalculationRangeCmd、 PerUnitAmountCalculationRangeCmd、または PercentageCalculationRangeCmd。 CalculationRangeMethod を決定するには、以下のようにします。
  - WebSphere Commerce オンライン・ヘルプの『CALMETHOD テーブル』を参照します。 SUBCLASS 列の説明を参照してください。『CALMETHOD テーブル: 詳細』のリンクをクリックします。このテーブルには、使用できる CALMETHOD のタイプがリストされています。 CalculationRangeMethod は 10 です。
  - ブートストラップ・ファイル wcs.bootstrap_xx_XX.xml を開きます (xx_XX はロケールのコード)。ブートストラップ・ファイルは以下のディレクトリーに置かれています。
    - *WC_installdir*/schema/xml
  - 使用できる計算メソッドをリストしているセクション (CALMETHOD) を見つけます。
  - calusage_ID が税 (消費税の場合は -3、配送税の場合は -4) の計算メソッドを見つけます。
  - サブクラスが 10 の計算メソッドを見つけます。サブクラス 10 にはいくつかの計算メソッドがあります。必要に応じて 1 つ選択してください。

- rangestart は、ルックアップ番号がこの RANGESTART 以上の場合、または RANGESTART が NULL の場合に、この行がルックアップ番号にマッチングする、というように使用されます。
- cumulative は、以下のとおりです。
  - 0 = 最高の RANGESTART 値に一致する CalculationRange だけが使用される。
  - 1 = 一致する CalculationRanges はすべて使用される。計算された通貨金額が合計されて、最終結果が得られます。

詳細については、WebSphere Commerce オンライン・ヘルプの『CALRANGE テーブル』を参照してください。

b. ストアの中で使用される計算スケールに関連した計算範囲ごとに、ステップ a を繰り返します。上記の例の場合、すべての金額の税率は同じなので、範囲は 1 つだけです。

11. 計算スケールの計算ルックアップ値を定義します。計算ルックアップ値は、計算スケールに関連した値です。たとえば、計算スケールに、レストランで消費される肉に対するオンタリオ州の消費税に関して、以下のような範囲と対応する税率が含まれているとします。
    - $0.00 ～ $3.99 の場合、税率 0.00%
    - $4.00 以上の場合、税率 8.00%

    この場合、ルックアップ値は 0.00 と 8.00 です。

    a. 以下の例を参考にして、 XML ファイルの CALRLOOKUP テーブルで計算ルックアップをセットアップします。

```
<calrlookup
 calrlookup_id="@calrlookup_id_37"
 calrange_id="@calrange_id_37"
 value="5.00"
 />
```

ここで、

- calrlookup_id は、生成される固有の ID です。
- calrange_id は、計算範囲ルックアップ結果を含む計算範囲です。
- value は、計算範囲ルックアップ結果の値です。この値は、計算範囲の計算範囲メソッドによって使用され、通貨結果が決定されます。この例の場合、税率は 5.00% です。

詳細については、WebSphere Commerce オンライン・ヘルプの『CALRLOOKUP テーブル』を参照してください。

b. ストアの中で使用される計算スケールに関連したルックアップ値ごとに、ステップ a と b を繰り返します。この例の場合、CALRLOOKUP.SETCCURR が NULL なので CALRLOOKUP 値は 1 つだけであり、すべての金額について税率が同じなので CALRANGE は 1 つだけです。

12. 計算ルールと計算スケールを関連付けます。

    a. 以下の例を参考にして、XML ファイルの CRULESCALE テーブルで計算スケールと計算ルールを関連付けます。

```
<crulescale
  calrule_id="@calrule_id_10"
  calscale_id="@calscale_id_19"
 />
```

ここで、

- calrule_id は計算ルールです。
- calscale_id は計算スケールです。

b. 関連付ける計算スケールとルールごとに、ステップ a を繰り返します。上記の例の場合、各計算ルールの計算スケールは 1 つだけです。

注: 税率が購入金額に応じて異なる場合、範囲開始値が 0 以外のスケールを作成することが必要になります。その場合、サポートされている通貨のうち換算率を設定していないものについて (CURCONVERT テーブルを参照)、 CALSCALE.SETCCURR を該当する通貨に設定することにより、サポートされている通貨ごとに計算スケールを作成し、さらにそれらすべてをその特定の課税カテゴリーの計算ルールに関連付ける必要があります。たとえば、オンタリオ州の消費税は、$4.00 以下の食事の場合には課せられません。米国ドルでの肉の販売をサポートするストアの場合は、米国ドルからカナダ・ドルへの換算を設定するか、または該当する範囲開始値 (おそらく $6.00 USD) を使用して別個の税額計算ルールを作成し、オーダーの通貨に応じて、該当する計算スケールだけが使用されることになります。

@ と **&** の使用法の詳細については、 489 ページの『付録 B. データの作成』を参照してください。

税フルフィルメント資産の作成

ストアで税資産が正しく機能するためには、ストアの中の課税管轄区域グループとストアで使用される配送センターを関連付けてから、その両方に計算ルールを関連付けなければなりません。

税資産を配送センターに関連付けるには、その前にフルフィルメント資産を作成しなければなりません。フルフィルメント資産の作成の詳細については、 228 ページの『WebSphere Commerce での配送資産の作成』を参照してください。

フルフィルメント資産を作成し終えたら、TAXJCRULE テーブルに情報を追加して、税資産を関連付けます。以下のようにします。

1. 「*IBMWebSphere Commerce 計算フレームワーク・ガイド*」に記載されている情報を確認します。 WebSphere Commerce 計算フレームワークは、顧客が購入のために選択した商品やサービスに関連した金額 (税金など) を計算します。
2. サンプル・ストアの税フルフィルメント資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。

   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

   - *WC_installdir*/samplestores

   各サンプル・ストアには 1 つの taxfulfill.xml ファイルが組み込まれています。これには税に関する情報が入っています。ストア・アーカイブの taxfulfill.xml ファイルを表示するには、ZIP プログラムを使用してこれを解凍します。 taxfulfill.xml ファイルは、データ・ディレクトリーにあります。

3. 489 ページの『付録 B. データの作成』に記載されている情報を確認します。
4. サンプル・ストア・アーカイブの taxfulfill.xml ファイルの 1 つをコピーするか、新しいファイルを作成することにより、taxfulfill.xml ファイルを作成します。詳細については、taxfulfill.xml に対応する DTD ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。
   - *WC_installdir*/xml/sar
5. 以下の例を参考にして、XML ファイルの TAXJCRULE テーブルに情報を追加します。

```
<taxjcrule
taxjcrule_id="@taxjcrule_id_1"
calrule_id="@calrule_id_10"
ffmcenter_id="@ffmcenter_id_1"
jurstgroup_id="@jurstgroup_id_2"
precedence="0"
 />
```

   ここで、
   - taxjcrule_id は、生成される固有の ID です。
   - calrule_id は、使用される計算ルールです。
   - ffmcenter_id は、配送センターです。これが NULL の場合、この関連はすべての配送センターに適用されます。
   - jurstgroup_id は課税管轄区域グループです。これが NULL の場合、この関連はすべての課税管轄区域グループに適用されます。
   - precedence は、配送先住所が、同一の配送センターと配送モードのために指定された複数の課税管轄区域グループに属する場合に、TAXJCRULE.PRECEDENCE 値が最大の計算ルールにのみ限定されることを示します。
6. ストアの管轄区域グループ、配送センター、およびルールの関連ごとに、ステップ 3 を繰り返します。

@ と **&** の使用法の詳細については、 489 ページの『付録 B. データの作成』を参照してください。

ストア - カタログ - 税資産の作成

ストアの税を商品やサービスと関連付けるには、ストアに組み込まれている契約ごとに、ストアの計算コードとカタログ・エントリーを関連付けなければなりません。

ストア - カタログ - 税資産を作成するには、その前にストア資産とカタログ資産を作成しなければなりません。ストア資産の作成の詳細については、 145 ページの『XML ファイルによるストア・データ資産の作成』を参照してください。カタログ資産の作成の詳細については、 187 ページの『ストア・カタログ資産の表示』を参照してください。

ストア - カタログ - 税資産を作成するには、以下のようにします。

1. 「*IBMWebSphere Commerce 計算フレームワーク・ガイド*」に記載されている情報を確認します。 WebSphere Commerce 計算フレームワークは、顧客が購入のために選択した商品やサービスに関連した金額 (送料など) を計算します。
2. サンプル・ストアのストア - カタログ - 税資産を作成するために使用される XML ファイルを確認します。サンプル・ストアのすべてのファイルは、対応するストア・アーカイブ・ファイルの中にあります。

   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

   - *WC_installdir*/samplestores
3. 489 ページの『付録 B. データの作成』に記載されている情報を確認します。
4. サンプル・ストア・アーカイブの `store-catalog-tax.xml` ファイルの 1 つをコピーするか、新しいファイルを作成することにより、`store-catalog-tax.xml` ファイルを作成します。詳細については、`store-catalog-tax.xml` に対応する DTD ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。

   - *WC_installdir*/xml/sar
5. CATENCALCD テーブルに情報を追加して、ストア - カタログ - 税の関係を作成します。次の例を参考にしてください。

```
<catencalcd
   calcode_id="@calcode_id_3"
   catencalcd_id="@catencalcd_id_3"
   store_id="@storeent_id_1"
/>
```

   ここで、

   - `calcode_id` は計算コードです。
   - `catencalcd_id` は、生成される固有の ID です。
   - `store_id` は、ストアです。

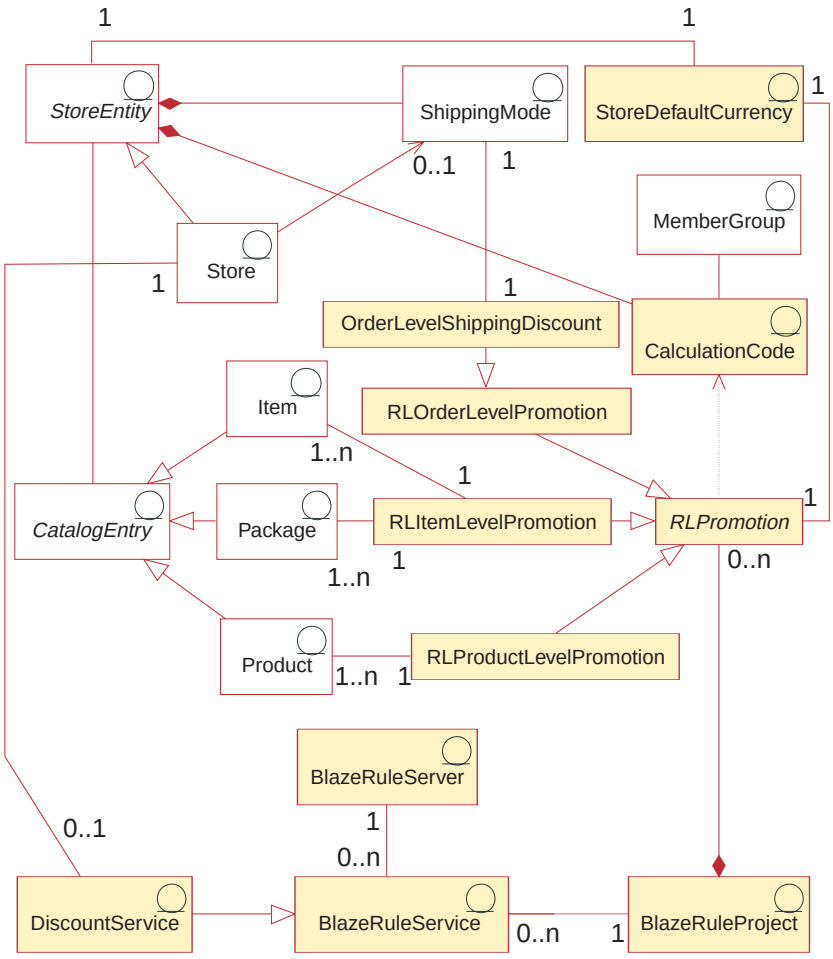
@ と **&** の使用法の詳細については、 489 ページの『付録 B. データの作成』を参照してください。

第 28 章 割引資産

割引は、顧客に対して売り上げを伸ばしたり商品の販売を促進したりするきっかけになる可能性があります。 WebSphere Commerce には、ルール・ベースの割引とスキーマ・ベースの割引という、 2 つの使用可能な割引のインプリメンテーションがあります。どちらのインプリメンテーションも、パーセンテージ割引 (10% 割引など) または固定額の割引 (15 ドルの値引きなど) を使用できます。割引は、特定の商品または買い物全体のどちらに対しても適用することができます。たとえば、高齢者に対して 20% の割引を実施したり、多数の赤い野球キャップが在庫にあれば、期間限定で赤いキャップを 25% 引きで販売することができます。ルール・ベースの割引は、これらの割引タイプ以上のもので、配送割引や購入時に進呈される無料の景品をオファーします。

WebSphere Commerce のルール・ベースの割引について

以下の図は、WebSphere Commerce Server のルール・ベースの割引構造を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、111 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、487 ページの『付録 A. UML の凡例』を参照してください。

ストアのデフォルト通貨

これは、STOREENT テーブルで定義されるストアのデフォルト通貨のことです。ルール・ベースの割引はこの通貨で定義されますが、ストアでサポートされているどの通貨でも使用してオンデマンドで評価できます。通貨の使用法の詳細については、249 ページの『第 23 章 通貨資産』を参照してください。

計算コード

割引 は、計算フレームワーク中に割引*計算コード* で表されます。割引計算コードは、対応する計算ルールによってオーダー・アイテムの割引が計算される方法を示します。

計算コードは、ストア・エンティティーに属するものです。 1 つのストア・エンティティー内に複数の計算コードを定義できます。ストア・エンティティーを削除すると、そのストア・エンティティーに定義されている計算コードも削除されます。

各割引計算コードには、割引の有効な期間を定義する開始日と終了日があります。また、割引計算コード有効なメンバー・グループを定義する 1 つ以上のメンバー・グループに関連付けることもできます。

RLPromotion

これはルール・ベース割引の親オブジェクトです。 RLPromotion はオブジェクト・タイプ名ですが、ルール・ベース割引に対応していると理解する必要があります。個々のルール・ベース割引には、名前、さまざまな状況で表示される複数の説明、優先順位、ターゲット・セグメント (事前定義済みの顧客プロファイル)、および日時を制御する実行スケジュールがあります。

優先順位の属性は詳細に説明する必要があります。優先順位の属性は、複数の割引を並行して適用できる場合に、競合を解決するのに役立ちます。適用できる割引は、それぞれの優先順位の値によって定義された順序で降順に適用されます。つまり、優先順位の値が最も高い割引が最初に適用されます。

以下にリストされているすべての子オブジェクトによって、ルール・ベースの割引タイプがさらに細かく分類され、必要に応じて割引タイプに固有の値が作成されます。これらの各オブジェクトにも、ドメイン XML ファイルを操作するための該当するロジックが含まれています。このファイルは割引を定義します。

RLProduct レベルの販売促進

この種のオブジェクトは、商品レベルのルール・ベース割引を表します。このクラスは RLPromotion クラスから派生します。このクラスには追加の属性 SKU が必要です。この追加属性は対象商品を識別します。

RLItem レベルの販売促進

この種のオブジェクトは、アイテム・レベルのルール・ベース割引を表します。このクラスは RLPromotion クラスから派生します。このクラスには追加の属性 `catEntryID` が必要です。この追加属性は対象商品を識別します。これらのアイテム・レベルの販売促進を使用して、パッケージをルール・ベースの割引の対象にすることもできます。なぜなら、独自の `catEntryID` と価格により個別にオーダーできるからです。バンドルとダイナミック・パッケージをルール・ベースの割引の対象にすることはできません。

RLOrder レベルの販売促進

この種のオブジェクトは、オーダー・レベルのルール・ベース割引を表します。これは RLPromotion から派生します。このクラスには追加の属性 `inCombineWithProductLevelDiscount` が必要です。この追加属性は、商品レベルの販売促進と同時にオーダー・レベルの割引を適用できるかどうかを判別します。

オーダー・レベルの配送割引: このクラスは RLOrder レベルの販売促進クラスから派生します。このクラスには、使用する配送方法と割引レートを定義する追加の属性が必要です。

Blaze ルール・プロジェクト

Blaze ルール・プロジェクトには、ドメイン XML ファイルから生成される、現在定義済みのストアの割引がすべて含まれます。このルール・プロジェクトは、ファイル・システム中にあり、割引サービスの移植に使用されます。

Blaze ルール・サービス

これは、 WebSphere Commerce で Blaze ルール・サーバーとの通信に使用されるインターフェースです。

割引サービス

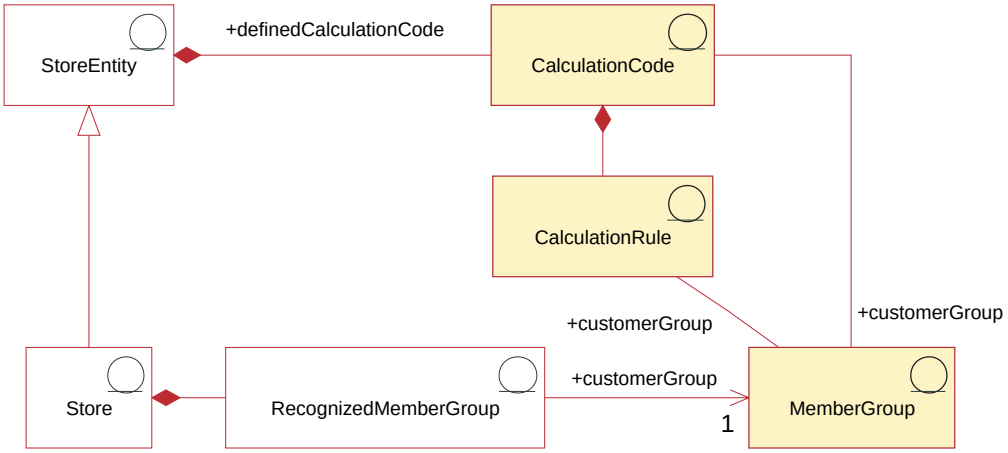
このサービスは、Blaze ルール・サービスから拡張し、評価や計算が行われるルール・プロジェクトのインスタンスが含まれます。このサービスは、WbeSphere Commerce 計算フレームワークに割引コンテキストを渡します。割引コンテキストとは、適用できる割引に関する情報を含むオブジェクトのことです。

Blaze ルール・サーバー

これは、オーダー処理コマンドからの要求が到着するたびにルール・プロジェクトを評価する Blaze ソフトウェアです。

WebSphere Commerce のスキーマ・ベースの割引について

以下の図は、WebSphere Commerce Server のスキーマ・ベースの割引構造を示しています。



計算コード

割引 は割引*計算コード* で表され、それを使用して計算されます。割引計算コードは、オーダー・アイテムの割引が計算される方法を示します。

計算コードは、ストア・エンティティーに属するものです。 1 つのストア・エンティティー内に複数の計算コードを定義できます。ストア・エンティティーを削除すると、そのストア・エンティティーに定義されている計算コードも削除されます。

各割引計算コードには、割引の有効な期間を定義する開始日と終了日があります。また、割引計算コード有効なメンバー・グループを定義する 1 つ以上のメンバー・グループに関連付けることもできます。

割引計算コードは、1 つ以上のカタログ・エントリーおよびカタログ・グループに、付加することができます。カタログ・グループに計算コードを付加すると、カタログ・グループのすべてのカタログ・エントリーに直接付加したのと同じ効果があります。しかし、カタログ・グループ A にカタログ・グループ B が含まれている場合、カタログ・グループ A の割引計算コードは、カタログ・グループ B 内の製品やアイテムには関連付けられません。

カタログ・エントリーやカタログ・グループに複数の割引が関連付けられる場合があります。 1 つのオーダーに複数の割引コードが適用可能な場合、割引計算は、計算コード順序属性の昇順に実行されます。

注: 割引の順序は、割引後の金額をさらに割り引くことをインプリメントするように定義してください。

以下の方法の 1 つにより、オーダー・アイテムは計算のためにグループ化されます。

- 取引条件別
- 商品別
- オファー別

- 配送先住所別

詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

計算コードを使用するための詳細については、「*IBM WebSphere Commerce 計算フレームワーク・ガイド*」を参照してください。

計算ルール

各計算コードには計算ルールのセットがあります。これは、計算が実行される条件を定義します。個々の割引計算ルールは 1 つ以上のメンバー・グループに関連付けられ、そのメンバー・グループに対して割引が有効になります。メンバー・グループに対して一度に複数の割引があてはまる場合があります。

注: 適格なメンバー・グループが計算コード・レベルで定義されている場合、それを計算ルールのレベルで再度定義する必要はありません。

計算ルールを使用するための詳細については、「*IBM WebSphere Commerce 計算フレームワーク・ガイド*」を参照してください。

WebSphere Commerce での割引資産の作成

WebSphere Commerce で作成したストアに割引を作成する主な方法は、 WebSphere Commerce アクセラレーターの「割引」ウィザードを使用することです。 WebSphere Commerce アクセラレーターを使用して割引を作成することの詳細については、 WebSphere Commerce オンライン・ヘルプを参照してください。

XML ファイルを使用して割引を作成し、その後でローダー・パッケージによってロードできます。しかし、この方法で作成された割引や、以前のバージョンからのマイグレーション時にインポートされた割引は、機能的には問題ないものの、 WebSphere Commerce アクセラレーターには正しく表示されない場合があります。

第 29 章 在庫資産

在庫には、配送センター内で物理的に存在可能なものがすべて含まれます。満たすべき在庫タイプの特定の定義 (たとえば、アイテム、商品、SKU、バンドル、パッケージなど) はありますが、これらはすべて在庫と見なされます。商品のフルフィルメントは、「商品」ウィザードと「商品」ノートブックで構成します。これには、在庫を追跡記録するオプション、バックオーダーを許可するオプション、バックオーダーを強制するオプション、別々にリリースするオプション、および商品を返品不可にするオプションが含まれています。 WebSphere Commerce アクセラレーターは、受け取ることのできる在庫の、 2 つの主要タイプを区別します。

- 予測在庫レコードと関連付けられた予測在庫
- 特別在庫、または予測として記録されていない在庫

予測在庫は取引先から受け取り、一般に、購入オーダーと一緒に支払われます。WebSphere Commerce アクセラレーターは、予定在庫レコードを使用して予定在庫をトラックするので、外部 ID (通常は外部システムからの購入オーダー番号) を記録できるようになります。これにより簡単に、到着したものと到着していないもの、およびオーダーした在庫を把握していくことができます。予測在庫詳細とは、予測在庫レコードの商品に関する詳細のことです。たとえば、商品を見込んでいる配送センター、予測受け取り日付、予測数量、コメントなどです。

在庫がいったん受け取られると、それに対応する予測在庫レコードは削除できません。また、予測在庫の詳細情報も、それに含まれる在庫の一部でも受け取られると、変更や削除ができなくなります。

配送センターで使用可能な在庫に対してオーダーが発行された場合、オーダー・システムは在庫をそれらのオーダーに割り振ります。在庫をオーダーに割り当てると、この在庫はオーダー・システムで使用できなくなります。オーダーがキャンセルされると、在庫はまた使用できるようになります。購入可ではない在庫がオーダーされた場合、バックオーダーを作成できます。バックオーダーを実施するために使用できる予測在庫がある場合、予測在庫はバックオーダーに対して割り振られ、顧客には予測配送日付が知らされます。

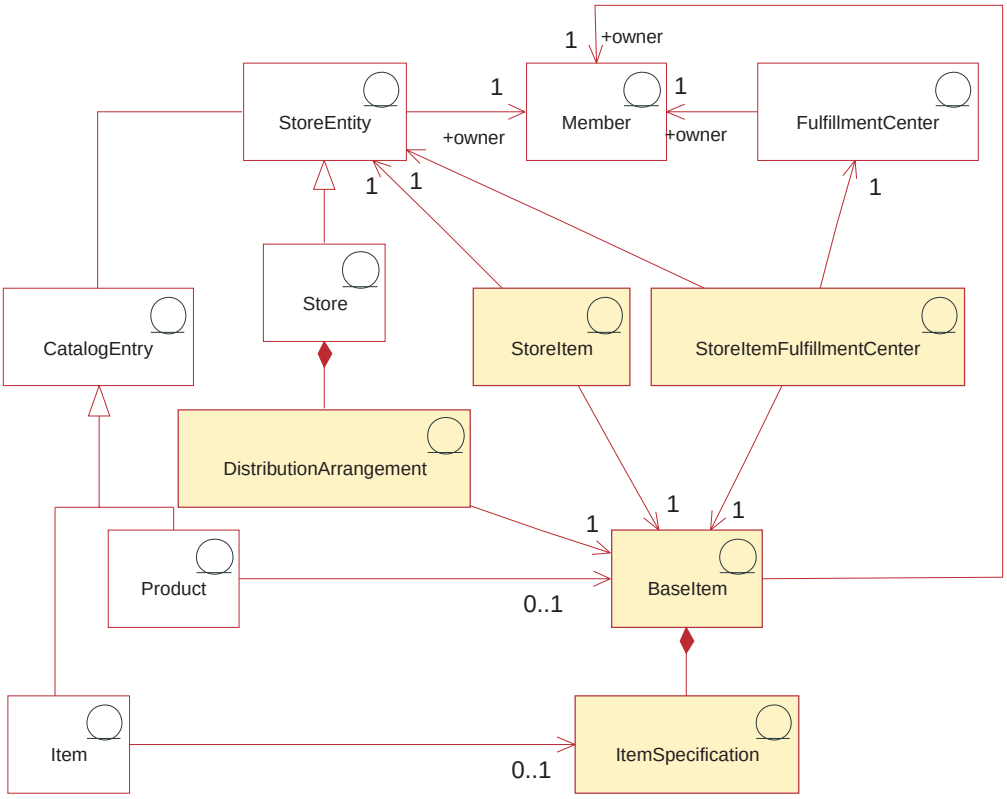
対応する予測在庫レコードがない在庫が配送センターに届くと、特別在庫受領書が作成されます。これは、予期せずに在庫が到着したために生じたのかもしれません。あるいは、予測在庫レコードを使用して在庫受け取りを記録しないようにマーチャントまたはセラーが選択した可能性もあります。

注: 在庫受け取りが予定在庫受け取りか特別在庫受け取りかに関係なく、商品が受け取られるためには、商品が WebSphere Commerce システムに存在しなければなりません。

WebSphere Commerce の在庫資産について

在庫資産を理解するには、在庫とストアの間の関係を理解することが必要です。これは、情報モデルを使用して説明できます。次に、ストアおよび他の資産と在庫との関係、および関連を説明します。以下の図は、販売可能在庫数量による納期回答 (ATP) 在庫の関係と非 ATP 在庫の間の関係と関連の全体像を示したものです。ストアは ATP または非 ATP 在庫方式のどちらでも使用できます。 STORE データベース・テーブル中の ALLOCATIONGOODFOR 列にゼロより大きい値が含まれている場合は、ストアで ATP を使用できると見なされます。個々の図とその関連については、後で説明します。 ATP の詳細については、オンライン・ヘルプを参照してください。

ATP 在庫



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、 111 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、 487 ページの『付録 A. UML の凡例』を参照してください。

基本アイテム

基本アイテム とは、名前と説明を共有する、商品の一般的なファミリーです。基本アイテムはもっぱらフルフィルメントのために使用されます。ストアごとに固有なものではありません。カタログの商品を表す各カタログ・エントリーには、フルフ

ィルメントを目的として、対応する基本アイテムがあります。基本アイテムは BASEITEM テーブルに定義されています。

アイテム仕様

アイテム仕様 とは、すべての属性の値が定義されている基本アイテムのことです。カタログのアイテムを表す各カタログ・エントリーには、フルフィルメントを目的として、対応するアイテム仕様があります。

カタログ・エントリー

商品やアイテムは*カタログ・エントリー* です。カタログ・エントリーは、ストア・エンティティーと関連付けられます。結果として、商品やアイテムなどのカタログ・エントリーがストアに表示されます。

配送手配

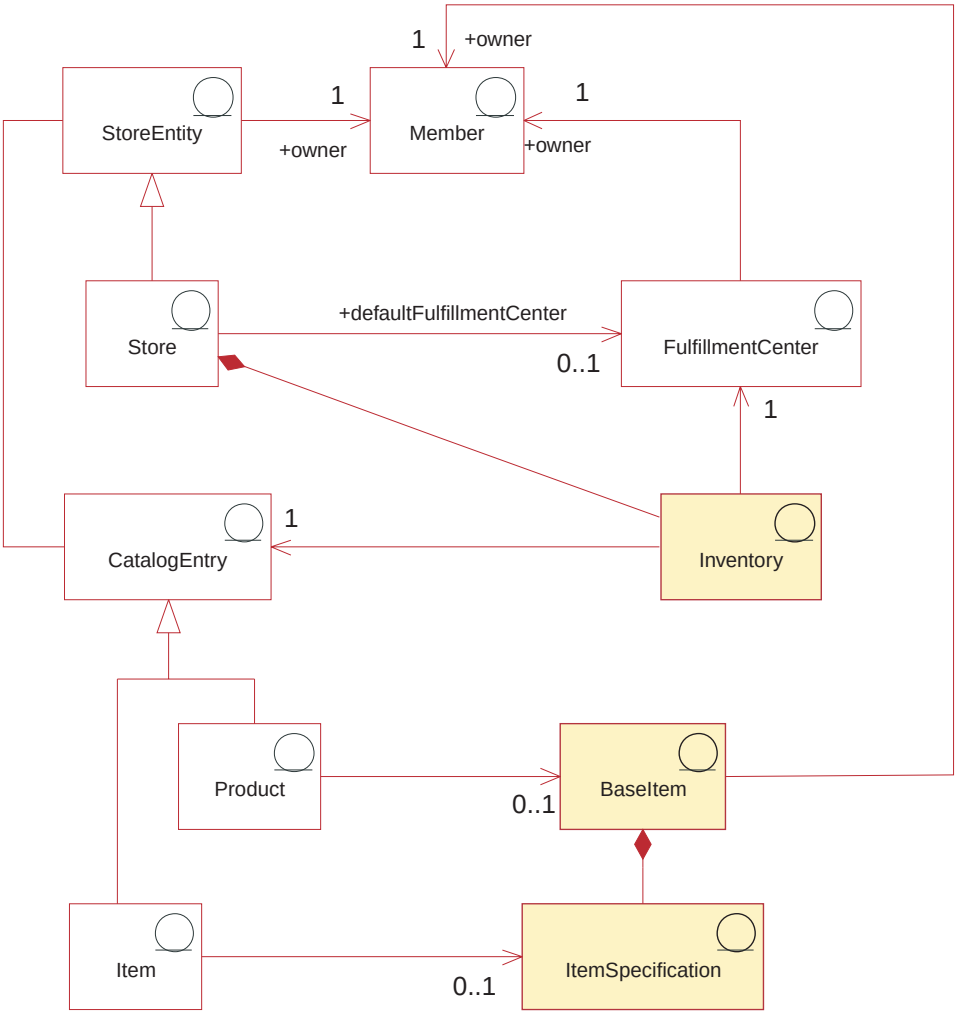
配送手配 は基本アイテムと関連付けられ、これによりストアは独自の在庫を販売できるようになります。配送手配は、DISTARRANG テーブルに保管されます。

ストア・アイテム

ストア・アイテム は、特定のストアやストア・グループが、特定の基本アイテムの指定アイテムの在庫を割り振る方法を制御する属性を表します。これには、バックオーダーや在庫調査を行えるようにするかどうかなどが含まれます。 STORITMFFC テーブルには、オーダー・アイテムがフルフィルメントのためにリリースされた時点から、これが顧客に配送されるまでにかかる見積秒数が定義されます。このテーブルにデータが読み込まれるのは、ストアがストア・アイテムの FFMCENTER デフォルト配送オフセットにオーバーライドを定義することを望む場合だけです。

他のストアによって使用できるストア・アイテム資産については、 151 ページの『第 14 章 ストア間の関係』で説明されています。

非 ATP 在庫



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、111 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、487 ページの『付録 A. UML の凡例』を参照してください。

非 ATP 在庫の図でも、基本アイテムが中心になっています。基本アイテムと商品、アイテム、およびカタログ・エントリーとの関係は、一般の在庫の図と同じです。この図でも、基本アイテムはメンバーによって所有されています。そのメンバーが基本アイテムを定義すると、これをストアで販売できるようになります。しかしながら、この図には、配送手配、ストア・アイテムの関連、ストア・アイテム配送センターがありません。

配送センター

在庫には 1 つの配送センター と 1 つのストアが関連付けられます。ストアは 1 つのデフォルト配送センターを指定できます。基本アイテムと同様に、 配送センターもメンバーが所有します。フルフィルメント資産の詳細については、225 ページの『第 19 章 フルフィルメント資産』を参照してください。

WebSphere Commerce Server の在庫資産の構造の詳細については、WebSphere Commerce オンライン・ヘルプで『在庫オブジェクト・モデル』と『データ・モデル』を参照してください。

WebSphere Commerce での在庫資産の作成

在庫は運用データなので、毎日、顧客がストアから商品を購入したりアイテムを返品したりするたびに変更されます。商品を販売したり、配送センターがサプライヤーから新規在庫を受け取ったりするたびに、在庫レベルは変動します。 WebSphere Commerce アクセラレーターによって、以下の関連タスクを実行することができます。

- 予測在庫の記録
- 取引先からの予測在庫と特別在庫の受け取り
- 在庫調整
- 返品記録の保守
- 返品理由の保守
- 顧客からの返品在庫の受け取り
- 返品在庫の処分の管理

WebSphere Commerce アクセラレーターを使用して在庫を管理することの詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

在庫調整コードの管理

在庫調整コードは、ユーザーが在庫に対する特定の調整を加える理由を指定できるようにするために、 WebSphere Commerce で提供されています。以下の表には、最初の時点で WebSphere Commerce で提供されている在庫調整コードがリストされています。 INVADJCODE および INVADJDESC データベース・テーブルに対して SELECT * データベース・クエリーを入力しても、このようなコードを表示できます。ストア開発者は、必要に応じてこれらのコードの追加、変更、または削除を行って、在庫環境の必要に合わせることができます。以下の表で説明されているコードは、使用できるコードのタイプの例と見なす必要があります。独自の調整コードを追加したり、プリロード済みのコードに変更を加えたりして、環境をさらにカスタマイズできます。

表 11. プリロード済みの在庫調整コード

調整コード	種類	説明
RTND	戻り	アイテム* が在庫に戻されたことを指定するには、このコードを使用します。たとえば、カラーかサイズが間違っていたため、顧客がオーダーからアイテムを戻すことが考えられます。このコードは、すべてのサンプル・ストアで使用するようにプリロードされています。

表 11. プリロード済みの在庫調整コード (続き)

調整コード	種類	説明
EXPD	期限切れ	在庫中のアイテムが期限切れになっていることを指定するには、このコードを使用します。たとえば、処方薬や保存期間の短い商品 (乳製品など) の有効期限が切れることが考えられます。このコードは、すべてのサンプル・ストアで使用するようにプリロードされています。
DMGD	損傷	アイテムが損傷している (へこみ、引っかき傷、切断、または欠陥など) ことを指定するには、このコードを使用します。このコードは、すべてのサンプル・ストアで使用するようにプリロードされています。
LOST	紛失	配送センターまたは保管場所でアイテムが紛失したり盗まれたりしたことを指定するには、このコードを使用します。このコードは、すべてのサンプル・ストアで使用するようにプリロードされています。
MSCT	数え間違い	以前に数えたアイテム数が間違えていることを指定するには、このコードを使用します。このコードは、すべてのサンプル・ストアで使用するようにプリロードされています。
PCNT	実査	アイテムの実査を指定するには、このコードを使用します。このコードは、消費者向けおよび B2B 向けのサンプル・ストア用にプリロードされており、ホストされるストア用ではありません。
SPLG	仕損品	アイテムがだめになったことを指定するには、このコードを使用します。このコードは、消費者向けおよび B2B 向けのサンプル・ストア用にプリロードされており、ホストされるストア用ではありません。
DISC	廃棄	在庫からアイテムが廃棄されたことやこれから廃棄されることを指定するには、このコードを使用します。このコードは、消費者向けおよび B2B 向けのサンプル・ストア用にプリロードされており、ホストされるストア用ではありません。
* この表では、「アイテム」という単語は一般的な意味で使用されています。在庫商品は、アイテム、商品、SKU、バンドル、およびパッケージの場合があります。		

この表で、「調整コード」列中の値は、 INVADJCODE データベース・テーブルの ADJUSTCODE 列にある値を表しています。「種類」列中の値は、 INVADJDESC データベース・テーブル中の DESCRIPTION 列にある値を表しています。

在庫調整コードの追加、変更、または削除を行うには、 Load コマンド (massloader コマンド) を使用してください。 Load コマンドの詳細については、393 ページの『Load コマンド』に記載されています。

サンプル・ストア内でデータが編成される方法の詳細については、 429 ページの『第 38 章 WebSphere Commerce データベース資産グループのロード』を参照してください。

Load コマンドを使用する前に、 wcs.dtd ファイルがご使用のパス (*WC_installdir*¥schema¥xml) に含まれていることを確認してください。 massloader.cmd ファイルもご使用のパス (*WC_installdir*¥bin) 中になければなりません。

在庫調整コードの追加

ストア用の在庫調整コードを追加するには、以下のようします。

1. 以下の例のように XML ファイルを (名前を自分で選択して) 作成し、ローダーが検出できる場所に配置します。 XML ファイル中で、invadjcode および invadjdesc エレメントの値を指定し、新しい在庫調整コードを INVADJ と INVADJDESC の 2 つのデータベース・テーブルに追加します。両方のテーブル中で、追加するコードの invadjcode_id が同じ値になっている必要があります。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE import SYSTEM "wcs.dtd">
<import>
<invadjcode invadjcode_id="404" adjustcode="BRKN" storeent_id="-1"
markfordelete="0" />
<invadjdesc invadjcode_id="404" description="BROKEN" language_id="-1" />
</import>
```

   ここで、

   - invadjcode_id は、割り当てる調整コード ID です。この ID は内部的に使用されるもので、 WebSphere Commerce ユーザー・インターフェースでユーザーに対して表示されません。
   - adjustcode は、ユーザー・インターフェースでの表示に適した、コードを固有に識別する 4 文字のコードです。
   - storeent_id は、ストア・エンティティーまたはストア・グループの ID です。この在庫調整コードをすべてのストアに追加するには、値「-1」を入力します。それ以外の場合は、特定のストア・エンティティーまたはストア・グループの ID を指定します。
   - description は、ユーザー・インターフェースでの表示に適した、在庫調整コードのテキスト記述です。
   - language_id は、ストアでショッピングする顧客に対して表示される情報のためのデフォルト言語です。言語サポートの詳細については、 337 ページの『第 34 章 グローバリゼーション』を参照してください。

   複数の在庫調整コードを追加する必要がある場合は、複数のコードを XML ファイル中に指定できます。

2. XML ファイルに対して Load コマンドを実行して、データを 2 つのターゲット・データベースにロードします。

   Windows

   ```
   massload.cmd -dbname dbname -dbuser dbuser -dbpwd dbpwd -infile
   xml_file_name -method sqlimport
   ```

   AIX Solaris Linux まず *wasuser* (WebSphere Application Server のユーザー ID) になります。

   ```
   su - wasuser
   ```

   次に、以下のコマンドを出します。

```
./massload.sh -dbname dbname -dbuser dbuser -dbpwd dbpwd -infile
xml_file_name -method sqlimport
```

400 QShell セッション (STRQSH) を開始します。次に、*WC_installdir*/bin ディレクトリーから以下のコマンドを実行します。

```
massload.sh -dbname dbname
-dbuser dbuser -dbpwd dbpwd -infile
xml_file_name -method sqlimport
```

3. データベース・クエリーを実行し、新しいコード値が両方のテーブルに追加されたことを表示して、在庫調整コードの追加を確認します。

在庫調整コードの変更

在庫調整コードの説明に変更を加えるには、以下のようにします。

1. 新しいコードを追加する場合と同様のステップに従います。ただし、XML ファイル中の特定の invadjcode_id 用の adjustcode および description エレメントの値を変更します。たとえば、

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE import SYSTEM "wcs.dtd">
<import>
<invadjcode invadjcode_id="404" adjustcode="DEFE" storeent_id="-1"
markfordelete="0" />
<invadjdesc invadjcode_id="404" description="DEFECTIVE" language_id="-1" />
</import>
```

この例では、前述の調整コードと種類 (BRKN、BROKEN) が、示されている新しい値に変更されます。必要に応じて、invadjcode_id 値も変更できます。invadjcode_id を検索するには、 massextract.cmd (Extract コマンド) を出します。 Extract コマンドの詳細については、 WebSphere Commerce オンライン・ヘルプを参照してください。 massextract.cmd は、データベースからデータを抽出し、それを XML ファイル中に挿入します。その後、このファイルをブラウズして、invadjcode_id 値を検索できます。代わりに、 INVADJCODE および INVADJDESC データベース・テーブルに対して SELECT * クエリーを実行することもできます。

2. 新しい調整コードを追加する場合と同じ massload.cmd を実行します。

Windows

```
massload.cmd -dbname dbname -dbuser dbuser -dbpwd dbpwd -infile
xml_file_name -method sqlimport
```

AIX Solaris Linux *wasuser* として、以下のコマンドを実行します。

```
./massload.sh -dbname dbname -dbuser dbuser -dbpwd dbpwd -infile
xml_file_name -method sqlimport
```

400 QShell セッション (STRQSH) を開始します。次に、*WC_installdir*/bin ディレクトリーから以下のコマンドを実行します。

```
massload.sh -dbname dbname
-dbuser dbuser -dbpwd dbpwd -infile
xml_file_name -method sqlimport
```

3. データベース・テーブル中で在庫調整コードに変更が加えられたことを確認します。

在庫調整コードの削除

WebSphere Commerce データベース・テーブルから在庫調整コードを削除するには、以下のようにします。

1. 削除したいコードを含む XML ファイルを作成します。例については、 311 ページの『在庫調整コードの追加』に示されているサンプル XML ファイルを参照してください。
2. 以下の削除方法を指定していることを確認して、massload.cmd を実行します。

   Windows

   ```
   massload.cmd -dbname dbname -dbuser dbuser -dbpwd dbpwd -infile
   xml_file_name -method delete
   ```

   AIX Solaris Linux *wasuser* として、以下のコマンドを実行します。

   ```
   ./massload.sh -dbname dbname -dbuser dbuser -dbpwd dbpwd -infile
   xml_file_name -method delete
   ```

   400 QShell セッション (STRQSH) を開始します。次に、*WC_installdir*/bin ディレクトリーから以下のコマンドを実行します。

   ```
   massload.sh -dbname dbname
   -dbuser dbuser -dbpwd dbpwd -infile
   xml_file_name -method delete
   ```

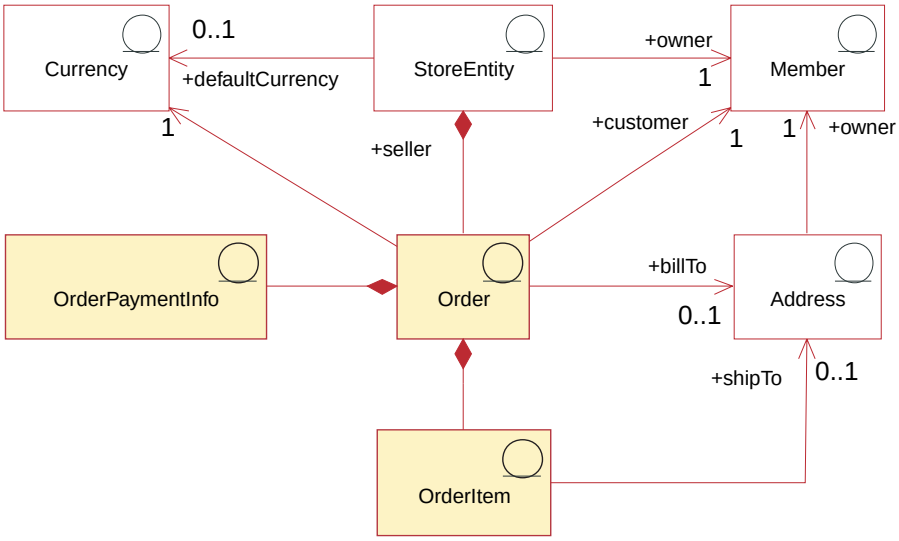
3. データベース・テーブルから在庫調整コードが削除されたことを確認します。

第 30 章 オーダー資産

WebSphere Commerce システムのオーダー資産は、ショッピング・カート、オーダー管理、およびオーダー・プロセッシング機能を備えています。オーダー・プロセッシング機能には、クイック・オーダーや即時購入、スケジュール・オーダー、複数保留オーダー、再オーダー、オーダーの分割、およびバックオーダーが含まれます。また、価格設定、税、支払い、およびフルフィルメントなどの関連サービスも、オーダー資産の一部です。

WebSphere Commerce のオーダー資産について

以下の図は、WebSphere Commerce Server のオーダー資産を示しています。図の後に、個々の資産の説明があります。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、111 ページの『ストア・データの情報モデル』を参照してください。

オーダーおよびオーダー・アイテム

WebSphere Commerce システムの場合、顧客やショッパーにとっては、オーダー は選択した商品のリストで (例えば、オーダーに 2 冊の本と CD を含めることができます)、そのリスト上にある各商品がオーダー・アイテム です (例えば、本や CD はそれぞれ、同じオーダーのオーダー・アイテムです)。顧客がストアでオーダーを発行する際には、ストアから送り状を送付する先となる、請求先住所を指定しなければなりません。各オーダーごとに 1 つの通貨 ID が関連付けられます。ストアの側から見ると、オーダーはオーダー・アイテムのリストです。これはストアのデータの一部です。

通貨

ストアでは、1 種類の*通貨* で価格を表示することもできますし、複数の通貨を使用することもできます。個々のストアで*デフォルト通貨* も定義しなければなりません。また、顧客が*ショッピング通貨* を選択できるようにすることもできます。ショッピング通貨がストアのデフォルト通貨と同じである場合、これは STOREENT テーブルですでにサポートされています。ショッピング通貨がストアのデフォルト通貨でない場合、CURLIST テーブルにその通貨を追加しなければなりません。顧客はショッピング通貨を使用して、ストアでオーダーを発行します。

支払い情報

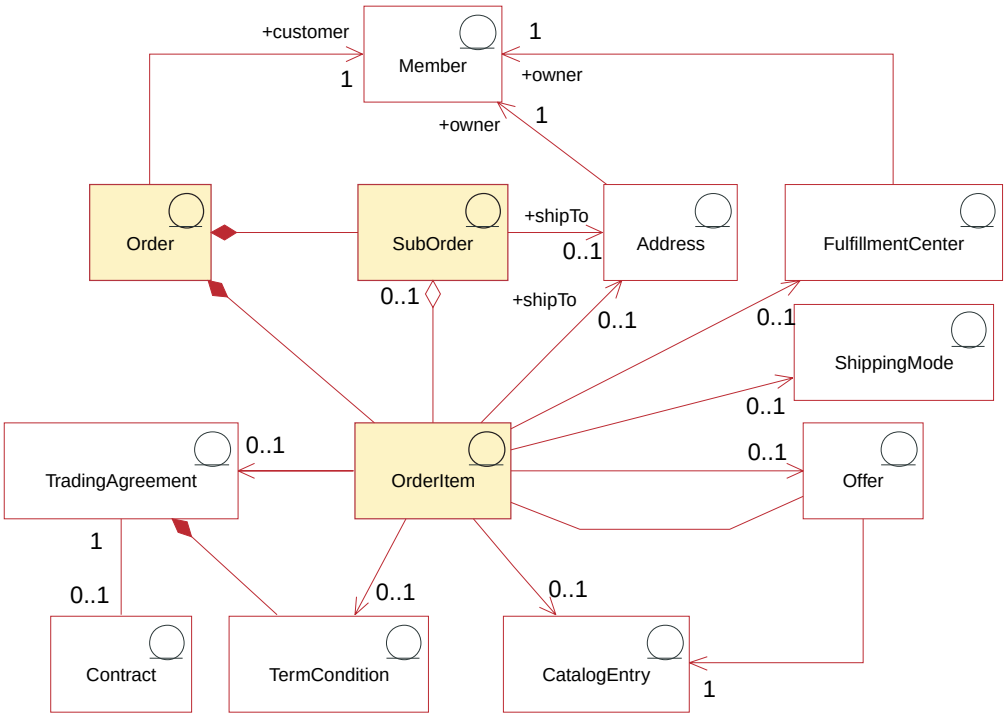
いったん顧客が希望のショッピング通貨を選択したら、支払いはすべてその通貨で処理されます。ストアの支払いのサポートとポリシーに応じて、顧客はオンライン決済 (この場合、顧客はインターネットを介してストアのサイトに支払い情報を提供する) かオフライン決済 (この場合、顧客はインターネット・チャネルを介さずに、電話や FAX などで支払い情報を提供する) のどちらかを使用して、支払いと購入を行うことができます。オンライン決済とオフライン決済のどちらの方式であるかに関係なく、顧客はオーダーを発行する際に、以下を含む*支払い情報* を指定しなければなりません。

- 支払い方式: 顧客がオーダーする際の支払い方式。ストアの WebSphere Commerce Payments で構成されている決済カセットに応じて、オフライン決済を受諾するか、顧客がオンライン・ウォレットを使用する必要のないオンライン決済に関する他の支払いプロトコルを使用するか、またはカスタム支払い方式を使用するようにストアをセットアップできます。
- カードに関する情報 (クレジット・カード支払いの場合): 顧客がオーダー時の支払いに使用するクレジット・カードの社名、番号、および有効期限日付。通常クレジット・カード情報は、ストアでオンライン決済がサポートされている場合に必要になります。
- 購入オーダー番号: ストアでオーダーする際に顧客が提示する購入オーダー番号。購入オーダー番号は、ストアと顧客との間の契約に記された条件によって規定されているように、その顧客がそのストアからオーダーする権限のある顧客であることを認証します。

オーダー・アイテム

オーダー・アイテム とは、オーダーに含まれる個々の商品やアイテムのことです。1 つのオーダーには 1 つ以上のオーダー・アイテムがなければなりません。各オーダー・アイテムは、顧客が購入に選択したものを表します。さらに、各オーダー・アイテムは、取引条件 (通常は契約)、配送モード、配送センター、および価格オファーを参照します。割引、配送料および税総額は、各オーダー・アイテムとともに保管されなければなりません。

以下の図は、WebSphere Commerce のオーダー・アイテム資産を示しています。図の後に、個々の資産の説明があります。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、 111 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、 487 ページの『付録 A. UML の凡例』を参照してください。

サブオーダー

オーダー・アイテムは、サブオーダー に分けられます。サブオーダーとは、特定の住所に配送されるオーダーの一部のことです。たとえば、顧客は、ショッピング・カート内の複数の商品に、それぞれ異なる配送先住所を指示することができます。個々の配送先住所と、それに関連付けられた商品が、サブオーダーを構成します。サブオーダー内のオーダー・アイテムの配送先住所は同じであり、それらのオーダー・アイテムの額の小計を表示するのに使用できます。

OrderItem オブジェクトの数量属性は、単位のない数値であり、 CatalogEntry オブジェクトに関連する CatalogEntryShippingInformation オブジェクトの名目数量属性を掛けて、その OrderItem によって表示される実際の量にすることができます。CatalogEntryShippingInformation オブジェクトは、数量の単位を指定します。

オーダーは通常、単一のストアに関連付けられますが、ストア、またはストア・グループのいずれかも関連付けることができる特別なオーダー・タイプが、オーダー・プロファイルです。オーダー・プロファイルは、オブジェクト・モデル内で、状況「Q」のオーダーとして表されます。オーダー・プロファイルは、顧客に関するデフォルトの情報 (支払い情報、配送先住所、配送モード、請求先住所など) を保持します。

その他のオーダー・アイテム資産

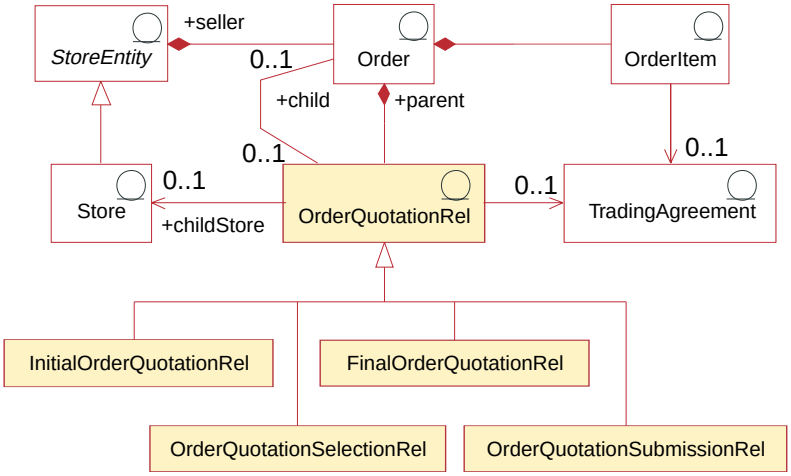
オーダー・アイテムを、以下のオブジェクトのうちの 1 つに関連付けることができます。あるいは、どれにも関連付けないことも可能です。

- そのオーダー・アイテムを含むオーダーを発行した顧客の配送先住所。ストアの配送センターで配送先住所を使用してオーダー・アイテムを適切に配送できるように、顧客はオーダー・プロセス中に配送先住所を指定しなければなりません。
- 顧客のオーダーに含める必要のあるオーダー・アイテムの配送や受け取りを行ったり、オーダー・アイテムの在庫を保管したりする配送センター。
- オーダー・アイテムの配送モード。これは、運送会社 (配送センターから顧客への配送サービスを提供する会社) とその運送会社が提供する配送サービスの組み合わせです。たとえば、ABC 運輸の翌日配送サービスや、 ABC 運輸の速達は、配送モードの一例です。
- オーダー・アイテムに関連付けられている価格のオファー。さまざまな価格表 (または「取引位置コンテナー」) にさまざまなオファーを組み込むと、ストアで、同じ商品または SKU について、顧客によって異なる価格を提供できます。たとえば、旅行代理店で、大人料金、高齢者料金、子供料金、および学生料金の 4 種類の価格表に飛行機のチケットのオファーを組み込むことができます。
- オーダー・アイテムのカタログ・エントリー。つまり、カタログからアイテムをオーダーすると個々のオーダー・アイテムになります。
- アイテムをオーダーする際の契約条件を定義した取引条件。これは通常は契約ですが、オーダーが処理用に送信されるまでの間は、Business 見積依頼 (RFQ) の場合もあり、これはネゴシエーションを表します。

WebSphere Commerce Server のオーダー資産の構造の詳細については、WebSphere Commerce オンライン・ヘルプで『オーダー・オブジェクト・モデル』と『データ・モデル』を参照してください。

オーダー見積もり関係

以下の図は、オーダー見積もり関係について説明しています。



WebSphere Commerce Server のオーダー資産の構造の詳細については、WebSphere Commerce オンライン・ヘルプで『オーダー・オブジェクト・モデル』と『データ・モデル』を参照してください。

オーダー見積もり関係 (OrderQuotationRel) とは、親のショッピング・カート・オーダーと子の見積もり関連オーダーとの間の関係のことです。バリュー・チェーン (デマンド) ビジネス・モデルでは、チャネルを使用して販売が行われる場合、チャネル域のショッパーはショッピング・カートを使用して複数のストアから商品やサービスの価格見積もりを入手し、その結果の見積もりから数量を選択し、見積もりを提供したストアにオーダーを送信できます。オーダー見積もり関係は、見積もりの要求元のストアと契約 (取引条件) を示します。

親のショッピング・カートには、見積もりのストアと契約の対ごとに、以下のタイプの子の見積もり関連オーダーがあります。それは、初期見積もり、現行見積もり選択、最終見積もり、および 1 つ以上の送信済みオーダーです。

InitialOrderQuotationRel の子のオーダーは、親のショッピング・カート・オーダー中のアイテムに関する受信済みの初期見積もりを表します。初期見積もりには、親のオーダーに明示的に示されていない代替商品や関連商品が組み込まれている場合もあります。

OrderQuotationSelectionRel の子のオーダーは、初期見積もりや最終見積もりから現在選択されているアイテムとその数量を表します。 OrderQuotationSelectionRel の子のオーダーをそのストアに送信して処理できます。

FinalOrderQuotationRel の子のオーダーは、選択オーダー中のアイテムに関する受信済みの最終見積もりを表します。最終見積もりは、その最終見積もりの要求が送信された時点で選択オーダー中にあるアイテムだけの価格と数量を提示します。

OrderQuotationSubmissionRel の子のオーダーは、送信済みオーダーを表します。選択オーダーが送信されると、その OrderQuotationSelectionRel オブジェクトは OrderQuotationSubmissionRel オブジェクトに変更されます。

WebSphere Commerce でのオーダー資産の作成

顧客はストアからオーダーを発行したり、ストアの顧客サービス担当者にこのタスクの完了を要求したり (WebSphere Commerce アクセラレーターを使用する) できます。消費者向け顧客に代わってオーダーを作成するには、 WebSphere Commerce オンライン・ヘルプのトピック『登録済み顧客のオーダーの作成』および『未登録顧客のオーダーの作成』を参照してください。 B2B 向け顧客に代わってオーダーを作成するには、ヘルプ・トピック『ビジネス・ユーザーのオーダーの作成』を参照してください。

第 31 章 取引先資産

取引先は、配送センターで受け取ったか、または配送センターで受け取ることが予期されている商品取引のソースを表します。ストア・モデルに応じて、取引先はバイヤー、プロダクト・マネージャー、セラー (マーチャント)、またはその他の与信済みの役割によって定義されます。 WebSphere Commerce アクセラレーターを使用して、すべての取引先のリストを表示したり、新しい取引先を作成したり、既存の取引先に変更を加えたり、取引先を削除したりできます。

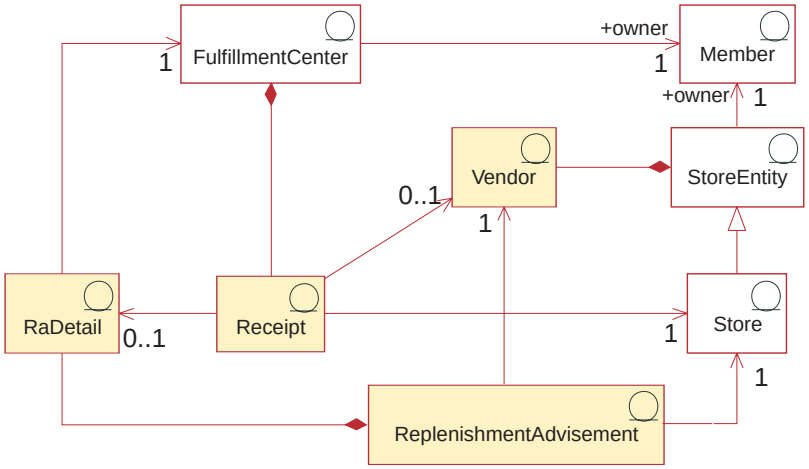
取引先レコードには、名前、住所、および連絡先情報などの、取引先に関する情報が含まれます。ストアで予定在庫レコードを作成するには、その前に取引先を作成する必要があります。

予定在庫レコードは、「予定在庫」ページ上で、取引先、外部 ID (普通は購入オーダー番号)、およびオーダー日付によって示されます。

取引先レコードの管理の詳細については、 WebSphere Commerce オンライン・ヘルプの『*取引先情報*』のトピックを参照してください。

WebSphere Commerce での取引先資産について

次に、ストアおよび他の資産と取引先との関係を説明します。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です。情報モデルの詳細については、 111 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、 487 ページの『付録 A. UML の凡例』を参照してください。

取引先は、ストアが商品取引を受け取るソースを表すので、ストアに対するサプライヤーと見なせます。

補充喚起 (replenishment advisement) と予定在庫レコードは同義です。

受け取りとは在庫受け取りのことです。通常、受け取りは予定在庫レコードからの結果です。

RaDetail は、在庫が予定されている日付、配送センター ID、およびオーダーした数量などの、予定在庫レコードのアイテムに関する詳細情報を表します。

在庫受け取りの詳細については、 305 ページの『第 29 章 在庫資産』を参照してください。

取引先資産の作成

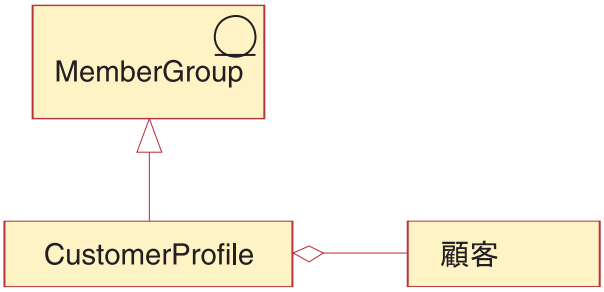
WebSphere Commerce アクセラレーターを使用して、取引先資産を作成できます。説明については、 WebSphere Commerce オンライン・ヘルプの『*取引先の作成*』を参照してください。

第 32 章 顧客プロファイル

顧客プロファイルは、マーケティング・メッセージの宛先をグループ化することにより、マーケティング作業の組織化に役立ちます。顧客プロファイルは、通常は WebSphere Commerce Accelerator を使用するマーチャントによって作成されます。

WebSphere Commerce での顧客プロファイルについて

以下の図は、WebSphere Commerce Server での顧客プロファイル資産を示しています。



この図、およびストア・データ・セクションの他のすべての図は、WebSphere Commerce Server 情報モデルの一部です情報モデルの詳細については、111 ページの『ストア・データの情報モデル』を参照してください。この図で使用されている規則の詳細については、487 ページの『付録 A. UML の凡例』を参照してください。

顧客プロファイルは、顧客またはアカウントの動的グループを定義する、登録情報、個人情報、住所情報、顧客文化情報、購入ヒストリー、およびその他の各種属性をまとめたものです。顧客プロファイルは、広告、販売促進、お勧め商品提示商法、割引、および E メール活動の宛先として使用できます。キャンペーンを作成する前に、顧客プロファイルを作成する必要があります。顧客は個人データおよび購入履歴に基づいてプロファイルに属していて、それらはどちらも変更される可能性があるので、プロファイルは動的なものとして扱われます。たとえば、顧客の登録状況にとづいてプロファイルを作成したと想定します。登録した顧客だけがメンバーとなれるプロファイルを作成すると、未登録の顧客は除外されます。その顧客が後に登録した場合、顧客はその宛先プロファイルのメンバーとなり、そのプロファイルが削除されるまでメンバーのままとなります。

顧客プロファイルは、静的基準もサポートします。特定のカスタマーまたはアカウントを明示的に含めるまたは除外することが可能で、その場合には定義済みの動的基準が指定変更されます。この方法により、たとえば本来は適合しない顧客をプロファイルに含めたり、本来は適合するアカウントをプロファイルから除外したりすることができます。つまり、静的基準と動的基準の両方が同じ顧客プロファイルに対して定義されている場合、まず動的基準が評価されてから静的基準が評価されることになります。

さらに、明示的に含めるまたは除外することのできる顧客やアカウントは、WebSphere Commerce Analyzer によるデータ・マイニングの結果に基づくものとすることもできます。 WebSphere Commerce は WebSphere Commerce Analyzer による高度な分析機能を内蔵しているので、マーチャントは WebSphere Commerce Analyzer が生成するセグメントに基づく明示的な顧客プロファイルを簡単に作成できます。

第 7 部 ストアへのアクセス制御の追加

第 33 章 ストアでのアクセス制御

WebSphere Commerce では、顧客であれ管理者であれ、特定のユーザーが実行できるタスクをアクセス制御によって決定できます。この章では、ストアにアクセス制御を追加する方法、たとえば、顧客に表示するページの制限やストア内で実行できるタスクの制限について取り上げます。

WebSphere Commerce 中のアクセス制御の詳細については、 39 ページの『第 4 章 WebSphere Commerce でのアクセス制御』を参照してください。アクセス制御モデルの詳細については、「*WebSphere Commerce セキュリティー・ガイド*」を参照してください。

WebSphere Commerce のアクセス制御について

ストアでのアクセス制御

WebSphere Commerce で作成されるストアはすべて、ストアを所有する組織がサブスクライブしたかまたは上位組織から継承した、デフォルトのアクセス制御ポリシーに従います。

デフォルトでは、ルート組織は管理ポリシー・グループにサブスクライブしません。 WebSphere Commerce に付属するいずれかのサンプル・ストアをベースにしてストアを作成する場合、そのストアを所有する組織によって所有およびサブスクライブされるストア特定のアクセス制御ポリシーおよびポリシー・グループを作成します。ストア特定のポリシー・グループに加え、所有する組織はストアの性質に応じて、管理、共通ショッピング、および B2C または B2B ポリシー・グループにもサブスクライブできます。サブスクライブするポリシー・グループの詳細については、「*WebSphere Commerce セキュリティー・ガイド*」を参照してください。

独自のストアを作成する場合には、サンプルをベースとしているかどうかに関係なく、新しいアクセス制御ポリシーを作成したり、既存のポリシーを変更することができます。これは、該当組織によって所有されるストアにのみ適用されます。たとえば、ストア・ページを表示するときの新しいビューを作成する場合、アクセス制御ポリシーをそれらのビューに割り当てる必要があります。

アクセス制御データは、高水準のアクセス制御ポリシー・ファイルで定義されます。これらのファイルでは、任意のポリシーで使用できるアクション、アクション・グループ、リソース、リソース・グループ、および関係を定義しています。さらに、特定の組織に固有なポリシーおよびポリシー・グループのサブスクリプションも定義しています。 WebSphere Commerce で提供されているサンプル・ストアには、このような高水準アクセス制御ポリシー・ファイルが含まれています。次の節では、サンプル・ストアがこのようなアクセス制御ポリシー・ファイルを使用して組織がサブスクライブしたアクセス制御ポリシー・グループ情報を定義する仕組みを説明します。

サンプル・ストアでのアクセス制御

すべてのストアが、高水準アクセス制御ポリシー・ファイルを持っています。これらのファイルは、それらのストア向けに特別に作成されたアクセス制御ポリシーおよびポリシー・グループを定義します。これらのアクセス制御データは、ストアを所有する組織に所有されます。

サンプル・ストアの高水準アクセス制御ポリシー・ファイルは、以下のとおりです。

- 消費者向け
  - FashionFlowAccessControl.xml
- Business B2B 向け
  - ToolTechAccessControl.xml
- Business デマンド・チェーン
  - CommercePlazaAccessControl.xml (チャネル・ハブ用)
  - ResellerStoreFrontAssetStoreAccessControl.xml
- Business サプライ・チェーン
  - SupplierHubAccessControl.xml
  - SupplierAssetStoreAccessControl.xml
- Business ホスティング
  - CommerceHostingHubAccessControl.xml
  - HostedStoreFrontAssetStoreAccessControl.xml
  - StoreDirectoryAccessControl.xml

これらのファイルは以下のディレクトリーにあります。

- *WC_installdir* /samples/stores/*businessmodel*

注: Express Express Store の高水準アクセス制御ファイルは、 Express Store ストア・アーカイブ (ExpressStore.sar) にあります。高水準アクセス制御ファイルは、`AccessControl.xml` と呼ばれています。

サンプル・ストアのアクセス制御ポリシー・ファイルについて: ストア・レベルでアクセス制御を追加する仕組みを理解するために、高水準サンプル・ストアのアクセス制御ポリシー・ファイルに精通してください。以下の例は、Business `ToolTechAccessControl.xml` ファイルから取られたものです。

アクションの定義: Business `ToolTechAccessControl.xml` ファイルの最初のセクションでは、ストアでの新しいアクションを定義します。これらは、ブートストラップ・アクセス制御ポリシーでは網羅されていません。このケースでは、アクションはストアで使用されるすべてのビューになります。 URL から直接に呼び出せるビューまたは (ビューに転送することで立ち上げることとは対照的に) 別のコマンドからのリダイレクトで立ち上げられるビューを使用してストアにページを表示するには、ビューをアクションとして定義しなければなりません。次の例を考えてください。

```
<!-- [Start of Action definitions] -->
<!-- [this is the dictionary of possible actions -->
<Action Name="GenericApplicationError"
   CommandName="GenericApplicationError">
 </Action>

 <Action Name="GenericSystemError"
   CommandName="GenericSystemError">
 </Action>

 <Action Name="OrderOptionsView"
   CommandName="OrderOptionsView">
 </Action>

  <!--[End of Action definitions] -->
```

ここで、

- Action Name は、XML ファイルでは、このアクションを指すときに使用されるラベルです。これらの例では、ラベルはビュー名と同じです。
- CommandName は、VIEWREG テーブルの VIEWNAME 列に保管されたビューの名前です。 CommandName は、ACACTION テーブルの Action 列に保管されます。

アクション・グループの定義: 2 番目のセクションでは、アクション・グループを定義します。このアクション・グループは、ファイルの最初のセクションで定義したアクションをグループ化したものです。 ToolTech の例では、新しいユーザー・ビューはすべて、グループ ToolTechAllUserViews (これは、すべてのユーザーがそれらのビューにアクセスできるようにするポリシーで使用される) にグループ化されるか、グループ ToolTechRegisteredCustomerViews (これは、登録ユーザーだけがそれらのビューにアクセスできるようにするポリシーで使用される) にグループ化されます。

注: また、WebSphere Commerce 内のどこかで定義されるアクションを、使用しているアクション・グループに追加することも可能です。これらのアクションが WebSphere Commerce のどこかで定義されている場合は、 328 ページの『アクションの定義』で説明した Action リストで再定義する必要があります。

```
<!-- [Start of Action Group definitions] -->
 <!-- Dictionary of grouped actions usable in policies -->
 <!-- cross-component view-related action groups -->
  <ActionGroup Name="ToolTechAllUsersViews"
    OwnerID="RootOrganization">

  <ActionGroupAction Name="UserRegistrationForm"/>
  <ActionGroupAction Name="UserRegistrationErrorView"/>
  <ActionGroupAction Name="GenericApplicationError"/>
  <ActionGroupAction Name="GenericSystemError"/>
  <ActionGroupAction Name="LogonForm"/>
    </ActionGroup>

  <!-- [End of Action Group definitions] -->
```

ここで、

- ActionGroup Name は、アクション・グループの名前です。アクション・グループ名は、ACACTGRP テーブル中に定義されています。

- OwnerID は、アクション・グループの所有者です。ルートの組織は、通常はアクション・グループの所有者です。その他の組織を使用する場合は、その組織の orgentity id を使用します。
- ActionGroupAction Name は、このグループに属するアクションの名前です。ActionGroupAction Name は、 328 ページの『アクションの定義』 の Action Name エレメントで定義した名前と一致していなければなりません。このアクションとアクション・グループの関係は、 ACACTACTGP テーブルに保管されます。

ポリシーの定義: 次のセクションでは、ストアで使用する新しいポリシーを定義します。

```
<!-- [Start of Policy definitions] -->

 <!-- AllUsers for ToolTech can execute ToolTechAllUsersViews -->


<Policy Name="AllUsersForToolTechExecuteToolTechAllUsersViews"
   OwnerID="&seller_b2b_mbr_id;"
   UserGroup="AllUsers"
   UserGroupOwner="RootOrganization"
   ActionGroupName="ToolTechAllUsersViews"
   ResourceGroupName="ViewCommandResourceGroup"
   PolicyType="groupableStandard">
    </Policy>
 <!-- RegisteredApprovedUsers for ToolTech can execute
ToolTechRegisteredApprovedUsersViews -->

  <Policy Name="RegisteredCustomersForOrgForTool
TechExecuteToolTechRegisteredCustomerViews"
   OwnerID="&seller_b2b_mbr_id;"
   UserGroup="RegisteredCustomersForOrg"
   UserGroupOwner="RootOrganization"
   ActionGroupName="ToolTechRegisteredCustomerViews"
   ResourceGroupName="ViewCommandResourceGroup"
   PolicyType="groupableTemplate">
    </Policy>

 <!-- [End of of Policy definitions] -->
```

ここで、

- Policy Name は、定義されるポリシーの名前です。ポリシーは ACPOLICY テーブルに定義されています。
- OwnerId は、ポリシーの所有者です。このケースでは、ポリシーの所有者は、ストアを所有する組織です。
- UserGroup は、ポリシーを適用する対象のユーザーのグループ (アクセス・グループ) です。
- UserGroupOwner は、アクセス・グループの所有者です。この例では、アクセス・グループの所有者は、ポリシーの所有者とは異なります。ポリシーの所有者と UserGroupOwner が同じであれば、このエレメントは省略できます。
- ActionGroupName は、ポリシーを適用する対象のアクションのグループです。
- ResourceGroupName は、ポリシーを適用する対象のリソースのグループです。この例では、リソース・グループ ViewCommandResourceGroup はブートストラップ・データ内に定義済みです。そのため、この高水準 xml ファイルで再定義する必要はありません。

- PolicyType は、ポリシーのタイプです。ポリシーは groupableStandard タイプまたは groupableTemplate タイプのいずれかになります。ポリシー・タイプの詳細については、「*WebSphere Commerce セキュリティー・ガイド*」を参照してください。

ポリシー・グループの定義: 最後のセクションでは、ストアに関する特定のポリシー・グループを定義します。

```
<PolicyGroup Name="ToolTechPolicyGroup" OwnerID="&seller_b2b_mbr_id;">
 <PolicyGroupPolicy Name="AllUsersForToolTechExecuteToolTechAllUsersViews"/>
 <PolicyGroupPolicy Name="RegisteredCustomersForOrgForToolTechExecuteTool
TechRegisteredCustomerViews"/>
 <PolicyGroupSubscription OrganizationID="&seller_b2b_orgentity_id;"/>
</PolicyGroup
```

ここで、

- PolicyGroup Name は、ポリシー・グループの名前です。この名前は ACPOLGRP テーブルに定義されています。
- OwnerID は、ポリシー・グループの所有者です。この例では、ポリシー・グループの所有者は、ストアを所有する組織です。
- PolicyGroupPolicy Name は、このグループに属するポリシーの名前です。このポリシーとポリシー・グループの関係は、ACPOLGPPOL テーブルに保管されます。
- PolicyGroupSubscription OrganizationID は、このポリシー・グループをサブスクライブする組織です。このサブスクリプションは、ACPLGPSUBS テーブルに保管されます。

アクセス制御に XML ファイルを使用することの詳細については、「*WebSphere Commerce セキュリティー・ガイド*」で『XML を使用したアクセス制御ポリシーのカスタマイズ』の章を参照してください。

ストアへのアクセス制御の追加

ストア開発の観点からすると、必要なアクセス制御のうち最も一般的なのは、ストアのために作成する新しいビューとコマンドのアクセス制御です。しかし、他のタイプのアクセス制御をストアに追加することができます。ビュー、コマンド、および他の機能用のアクセス制御の詳細については、「*WebSphere Commerce セキュリティー・ガイド*」を参照してください。本書で概説されている次のステップに進む前に、必ず「*WebSphere Commerce セキュリティー・ガイド*」に記載されている情報を確認してください。

新しいアクセス制御機能を、サンプル・ストアをベースにしたストアに追加する場合、既存の高水準アクセス制御ポリシー XML ファイルを編集します。アクセス制御を、サンプル・ストアをベースにしていないストアに追加する場合、新しい高水準アクセス制御ポリシー XML ファイルを作成する必要があります。両方のシナリオに関する詳細な手順については、332 ページの『ストアでのアクセス制御の作成または編集』 を参照してください。

ストアでのアクセス制御の作成または編集

アクセス制御資産はストアの他の資産とは異なり、高水準アクセス制御 XML ファイルを作成してから、それらを変換してロードします。

アクセス制御資産を作成したり編集したりするには、以下のようにします。

1. サンプル・ストアのストア資産を作成するために使用される高水準 XML ファイル: *samplestorename*AccessPolicies.xml を確認します。これらのファイルは以下のディレクトリーにあります。

   - *WC_installdir*/samples/stores/*businessmodel*

   高水準アクセス制御 XML ファイルの構文については、*samplestorename*AccessPolicies.xml によって参照される DTD ファイルを参照してください。 DTD ファイルは以下のディレクトリーにあります。

   - *WC_installdir*/xml/policies/dtd

   サンプルをベースにしてストアに変更を加えるには、これらのファイルを編集します。サンプルをベースにしていないストアにアクセス制御を追加する場合、サンプル・ストア・ファイルの 1 つをコピーして、ご使用のストアの新しい高水準アクセス制御ファイルを作成してください。ファイルをコピーした後は、名前変更して該当する編集を加えるだけで済みます。ストアにアクセス制御を追加することの詳細については、「*WebSphere Commerce セキュリティー・ガイド*」を参照してください。
2. 該当する *samplestorename*AccessPolicies.xml ファイルを以下のディレクトリーにコピーします。

   - *WC_installdir*/xml/policies/xml
3. 以下の図表に示されているプレースホルダーを、実際の値に置き換えます。値を検索するには、以下のステップを実行します。

   a. 以下のクエリーを使用して、識別名 (DN) の値を検索します。以下の図表に示されている DN 値をクエリーで使用します。

      - `select orgentity_id from orgentity where dn= <the DN of the organization>`

      例: CommercePlazaAccessControl.xml ファイルで、 &channel_mbr_id; と &channel_orgentity_id; を、上記のクエリーを使用して検索した値に置き換えます。 DN 値は ou=Reseller Hub Organization,o=Demand Chain Management Organization,o=Root Organization です。

表 12.

ファイル名	実際の値に置き換えるプレースホルダー	DN 値
Business CommercePlazaAccessControl.xml	&channel_mbr_id; &channel_orgentity_id;	ou=Reseller Hub Organization,o=Demand Chain Management Organization,o=Root Organization
	&proxy_orgentity_id;	ou=Distributor Proxy Organization,o=Demand Chain Management Organization,o=Root Organization
Business ResellerStoreFrontAssetStoreAccessControl.xml	&profile_mbr_id;	ou=Asset Store Organization,o=Demand Chain Management Organization,o=Root Organization
Business SupplierAssetStoreAccessControl.xml	&profile_orgentity_id &;profile_mbr_id;	ou=Asset Store Organization, o=Supply Chain Management Organization,o=Root Organization
Business SupplierHubAccessControl.xml	&channel_orgentity_id; &channel_orgentity_id;	ou=Supplier Hub Organization,o=Supply Chain Management Organization,o=Root Organization
FashionFlowAccessControl.xml	&seller_b2c_mbr_id; &seller_b2c_orgentity_id;	″ou=B2C,o=Seller Organization,o=Root Organization″
Business ToolTechAccessControl.xml	&seller_b2b_mbr_id; seller_b2b_orgentity_id;	ou=B2B,o=Seller Organization,o=Root Organization
CommerceHostingHublAccessControl.xml	&channel_mbr_id; &channel_orgentity_id;	ou=Hosting Hub Organization,o=Hosting Organization,o=Root Organization
HostedStoreFrontAssetStoreAccessControl.xml	&profile_mbr_id;	ou=Asset Store Organization,o=Hosting Organization,o=Root Organization
StoreDirectoryAccessControl.xml	&public_mbr_id; &public_orgentity_id;	ou=Store Directory Organization,o=Hosting Organization,o=Root Organization

4. 適切なアクセス制御情報をファイルに追加します。詳細については、 328 ページの『サンプル・ストアでのアクセス制御』および「*WebSphere Commerce セキュリティー・ガイド*」を参照してください。
5. acpload コマンドを実行して、 *samplestorename*AccessPolicies.xml ファイルを変換し、データベースにロードします。

   注: acpload コマンドを実行するためには、データベース・ユーザー ID に以下の許可が必要です。

   - *WC_installdir*/xml/policies、*WC_installdir*/logs のディレクトリー、サブディレクトリー、およびファイルに対する、読み取り/書き込み/実行の権限。
   - *WC_installdir*/bin ディレクトリーおよびそのファイルに対する、読み取り/書き込みの権限。

   a. コマンド・プロンプトで、ディレクトリーを次のように変更します。
      - *WC_installdir*/bin
   b. acpload コマンド・ファイルを実行します。
      - 2000 **構文**: acpload.cmd *databasename database user database user password Policies XML file* [schema name] **例**: acpload mall dbuser dbusrpwd ChannelHubAccessControl.xml
      - 400 AIX Solaris Linux **構文**: acpload.sh *databasename database user database user password Policies XML file* [schema name] **例**: acpload.sh mall dbuser dbusrpwd ChannelHubAccessControl.xml
   c. 次のログ・ファイルを調べ、アクセス制御データが正常にロードされたことを確認します。
      - *WC_installdir*/logs/acpload.log
      - 400 *WC_userdir*/instances/acpload.log

@ と **&** の使用法の詳細については、 489 ページの『付録 B. データの作成』を参照してください。

第 8 部 ストアのグローバル化

第 34 章 グローバリゼーション

グローバリゼーションのサポート

WebSphere Commerce は、以下の 10 の言語に翻訳されます。

- 米国英語
- フランス語
- ドイツ語
- イタリア語
- スペイン語
- ブラジル・ポルトガル語
- 中国語 (簡体字)
- 中国語 (繁体字)
- 韓国語
- 日本語

これにはソフトウェア、その資料、ユーザー・インターフェース、およびサンプルが組み込まれています。他の言語サポートも追加することができます。ページ上の商品説明、メッセージ、およびテキストなど、サイトのフィーチャーの多くを翻訳することができます。これはストア上のページ、および WebSphere Commerce アクセラレーターおよび管理コンソールなどの、 WebSphere Commerce ツールに基づいたブラウザー上のページに適用されます。

WebSphere Commerce には、サイト作成のためのいくつかのフィーチャーが組み込まれています。それらは、国際的または文化的に多種多様な顧客の基本的な必要に適合させるために、調整することができます。 Java テクノロジーおよび柔軟なデータベース・スキーマを使用することにより、ロケーションまたはお客様ごとの設定に基づいて、サイトの国/地域別環境の特性を変更することができます。それには以下のようなものがあります。

言語 ストアは複数の言語で表示することができます。たとえば、サイトを表示する言語をユーザーが選択できるようにしたり、ストアのロケーションに応じてデフォルトの言語が自動的に選択されるようにすることができます。

通貨 ストアは複数の通貨で価格を表示することができます。 WebSphere Commerce では、どの通貨を使用するかは、言語の選択には関連していません。

国/地域別環境形式
データは、各種のカスタマイズ可能な形式で表示することができます。国/地域別環境の異なる顧客に対しては、特定の情報を異なる仕方で表示する必要があるかもしれません。たとえば 10 進数を表記するのに、顧客の言語、国、または地域によっては、コンマを使用したり、ピリオドを使用したりします。

住所 住所は、さまざまな国際標準に合わせるために、さまざまな形式で入力、保管、および表示されます。

課税　課税規則を、さまざまな取扱範囲について定義することができます。これには、売り上げ税およびその他の事業税に関する規則も含まれます。

配送　配送規則および運送会社を、さまざまな取扱範囲について定義することができます。

支払いメソッド
支払いメソッドを、さまざまな取扱範囲について定義することができます。

価格　ある通貨で価格を設定し、変換係数によってこの通貨から他の通貨へ変換することもできますし、通貨ごとに価格を設定することもできます。

オンライン・カタログ内容
商品およびカテゴリー説明、属性、およびイメージは、ロケーションによって変更することができます。 WebSphere Commerce では、言語テーブル内のそれぞれのフォーマットを定義することによって、選択可能な表示フォーマット用のオンライン・カタログ内容をマーチャントが管理できます。また、多数のストアと共用することのできるマスター商品カタログを保守することもできます。

WebSphere Commerce データベースは、Unicode UTF-8 エンコードまたは
400 UCS-2 を使用することによって、国際的に認識可能なデータを作成および保守できるような柔軟性を持つよう設計されています。 Unicode UTF-8 エンコードは、Web ブラウザーに送られる時に、最も国際的なエンコード形式に変換することができます。多数の国のためにグローバル化されたサイトの実用的なサンプルについては、サンプル・ストアのいずれかを参照してください。

管理コンソールおよび WebSphere Commerce アクセラレーターは、グローバリゼーションの使用をサポートします。これらのツールは、WebSphere Commerce のサポートする 10 の言語のいずれかで表示されます。それらは一般機能またはページのルック・アンド・フィールに影響を与えることなく別の言語を追加できるようにするために、プログラミング・モデルを使用します。

ストアやその他の Web サイトを翻訳するように、ツールも他の言語に翻訳することができます。ツールを翻訳するには、次のディレクトリーにある適切なプロパティー・ファイルを変更してください。
WAS_userdir/installedApps/*cell_name*/*WC_instance_name*.ear/properties
/com/ibm/commerce/tools/

サンプル・ストア

サンプル・ストアおよびサイトは、ストアを作成するための土台を提供します。すべてのサンプル・ストアは、グローバル化されたサイトを作成および保守する方法を例示します。

サンプル・ストアでは、顧客はサイトを表示する言語を選択することができます。FashionFlow および Commerce Plaza では、リストから希望の表示フォーマットを選択します。リストはサイト全体を通じて、左のフレームのドロップダウン・リストに表示されます。顧客はサイトをナビゲートして、それを選択した言語で表示することができます。

Business ToolTech では、個人情報ページに進んで、優先言語を選択することによって、言語を切り替えます。

Business Commerce Hosting Hub、および Commerce Supplier Hub では、ログオン・ページに、言語を選択するために使用できるドロップダウン・リストがあります。

サンプル・ストアは、すべてのストアおよび言語プログラミング・モデルに対して、単一テンプレートを使用します。サポートされる各言語には、その言語用の翻訳されたテキストおよびメッセージが含まれている、独自のプロパティー・ファイルがあります。すべてのストア・アーカイブには以下のものが含まれます。publishNLS.properties、publishNLS_en_US.properties。これらは、発行ウィザードでのみ使用されます。

FashionFlow および FashionFlow に基づいてホストされるストアの場合は、以下が含まれます。

- infashiontext_*locale*.properties
- infashiontext_dynamic_*locale*.properties
- infashiontext_dynamic_labels_*locale*.properties (ページの変更 GUI にのみ使用される)
- AuctionSample_*locale*.properties

Business ToolTech の場合は次のとおりです。

- tooltechtext_*locale*.properties

Business Commerce Plaza の場合は次のとおりです。

- pcdmarket_*locale*.properties

Business Commerce サプライヤー・ハブの場合は次のとおりです。

- tooltechtext_*locale*.properties

Business Commerce ホスティング・ハブの場合は次のとおりです。

- b2cHostingChannel_*locale*.properties

Business ホスティング・サンプルにあるストア・ディレクトリーの場合は以下のとおりです。

- b2cHostingPublic_*locale*.properties

プロパティー・ファイル内では、ページの多数のエレメントが翻訳されています。

テキスト
テキスト・ページ・コンテンツ。

ラベル フォーム・フィールド・ラベル。

メッセージ
エラー、状況、および確認メッセージ。

代替テキスト

イメージ、Java アプレット、その他の埋め込みオブジェクト用。

文字セットの判別

ブラウザーでテキストが表示される文字セットは、プロパティー・ファイル内で ENCODESTATEMENT プロパティーを使用して定義されます。たとえば、infashiontext_en_US.properties ファイルには、次のステートメントが含まれます: ENCODESTATEMENT = text/html; charset=ISO_8859-1 エンコードは JSP テンプレートではなくプロパティー・ファイル内で指定されるので、それぞれの言語ごとに文字セットを変えることができます。生成された JSP ページの文字エンコードは、 JSP テンプレート内の次のステートメントを使用して設定されます。

<%response.setContentType(infashiontext.getString ("ENCODESTATEMENT")); %> 実行時、要求された各 JavaServer Page ページには、ファイル EnvironmentSetup.jsp が組み込まれます。このファイル内でコマンド・コンテキストが検索され、そこからロケールを使用して、infashiontext Properties Java オブジェクトを検索します。このオブジェクトは、適切なロケール固有のディレクトリー内の infashiontext_locale.properties ファイルから値を取得します。その後、必要に応じて、ResourceBundle オブジェクトの getString() メソッドを使用して、テンプレートがそれぞれのプロパティーにアクセスします。

注: *これは、ストア作成ウィザードのみを使用して作成される、ホストされるストアのバリュー・チェーン用です。* セラーがホストされるストアを作成する場合、これは*通常*のストア発行ではなく、ストアのデフォルト言語以外の追加のストア言語資産はホストされるストアに引き継がれません。このため、セラーがサポートされる言語をストアに追加する場合、その言語のストア資産を使用することはできません。サポートされる言語をホストされるストアに追加する場合、変換された資産 (プロパティー・ファイル) がストアに対して使用可能であるか、ストア・ページが正常に機能しないことを確認します。

翻訳されたイメージの表示

ロケールおよび言語が実行時に検索されて、イメージ・ファイルの検索場所である正確なフォルダーを判別します。テンプレートはファイル FashionFlow/*language_Locale*/images/go_button.gif を探すことがあります。*language_Locale* は、コマンド・コンテキストからの表示フォーマットによって置換されます。たとえば、表示されるページは、次のイメージを表示します。FashionFlow/en_US/images/go_button.gif または FashionFlow/jp_JA/images/go_button.gif

カタログ内容の表示

カタログには、複数の翻訳が含まれています (サポートされるそれぞれのロケールごとに 1 つずつ)。実行時に、コマンド・コンテキストが Data Bean を介して送られて、データベースから検索してページに表示するのはどの翻訳かを決定します。

リソース・バンドルとプロパティー・ファイル

リソース・バンドルおよびプロパティー・ファイルによって、ご使用の JavaServer Pages 用のロケール固有の Java オブジェクトのコレクションを保守することができます。フォーム・フィールド・ラベル、グラフィカル・ユーザー・インターフェース・メッセージ、またはドロップダウン・メニューの値など、ロケール固有のリソ

ースがページに必要な場合、ページはそれを、選択したロケールに適したリソース・バンドルまたはプロパティー・ファイルからロードして、顧客がそのページを自分の言語で表示できるようにすることができます。この方法では、リソース・バンドルまたはプロパティー・ファイル内のロケール固有の情報すべてを分離して、顧客のロケールとは大いに独立した JSP テンプレートを作成することができます。

リソース・バンドルおよびプロパティー・ファイルは同様の機能を実行しますが、それらが処理される仕方がいくらか異なっています。下記の表は、リソース・バンドルとプロパティー・ファイルのさらに重要な差異のいくつかを示しています。

表 13. ListResource バンドルとプロパティー・ファイル

プロパティー・ファイル	ListResource バンドル
テキスト・ファイル	コンパイル形式
性能が若干低下する	性能が若干良くなる
プロパティー・ファイルが変更された場合には、WebSphere Application Server を再始動して変更を表示する必要がある。	リソース・バンドルが変更された場合には、再コンパイルが必要であり、 WebSphere Application Server を再始動して変更を表示する必要がある。
言語およびロケールに依存。それぞれのロケールごとに、1 つのファイルが必要。非 ISO-8859-1 文字に対して native2ascii を実行する必要がある。	言語およびロケールに依存。それぞれのロケールごとに、1 つのファイルが必要。非 ISO-8859-1 文字に対して native2ascii を実行する必要がある。

多数の国のためにグローバル化されたサイトでのプロパティー・ファイルの使用例は、サンプル・ストアを参照してください。これらのトピックの詳細については、Sun Microsystems Java Web サイトをご覧ください。

データの保管および転送

それぞれの言語が異なる文字セットを使用するとしても、単一ストアは複数言語でページを表示することができます。これを行うには、広範囲な言語に適用できるユニバーサル・フォーマットで、 WebSphere Commerce データベースにデータを保管します。必ずしもすべての Web ブラウザーが同じ文字セットをサポートするとは限らないので、データが JavaServer Page によって要求された場合には、それが適切な文字セットに変換されます。

以下に、データベースからブラウザーにデータが転送される仕方を記述します。

1. テキスト・データが Unicode UTF-8 エンコードまたは 400 UCS-2 を使用して、WebSphere Commerce データベースに保管されます。
2. JDBC ドライバーがデータベースからデータをロードして、それを UTF-8 から Java の固有 16 ビット Unicode エンコードに変換します。
3. JSP ファイルは Java 16 ビット・エンコードを使用してデータを出力します。
4. WebSphere Application Server は、JSP ファイル出力を 16 ビット Unicode から宛先のエンコードに変換します。エンコード方式は、JSP ファイル内またはプロパティー・ファイル内のいずれかで指定することができます。例えば、次のようにして、日本語のページに対して Shift-JIS エンコードを指定します。
   - JSP ファイル <%@ page contentType=″text/html; charset=Shift-JIS″%>

- プロパティー・ファイル ENCODESTATEMENT = text/html; charset=Shift-JIS。 生成された JSP ファイルの文字エンコードは、JSP テンプレート内で次のステートメントを使用して設定されます。 <%response.setContentType(fashionflowtext.getString (″ENCODESTATEMENT″)); %>

  必ずしもすべてのブラウザーがすべてのエンコード・スキームを理解できるとは限らないので、 UTF-8 や Shift-JIS など、よく知られたエンコード・スキームを指定することをお勧めします。

5. 変換されたデータがブラウザーに戻されます。
6. ブラウザーは、ヘッダーで指定されたエンコードに基づいて、HTTP 応答を解釈します。

以下に、ブラウザーからデータベースにデータが転送される仕方を記述します。

1. データがブラウザーに入力されます。入力メソッドを使用して、マルチリンガル・データを入力することができます。
2. WebSphere Commerce がブラウザーから送られたデータを、setCharacterEncoding() メソッドを使用して、Java 16 ビット・エンコードに変換します。 LANGUAGE テーブル内のそれぞれの LANGUAGE_ID は、ENCODING 列を使用してエンコード値にマップされます。この値を使用して、ブラウザーか送られるデータを解釈します。
3. データがデータベースに送られ、そこで Java 16-bit から UTF-8 エンコードに変換されて、データベースに保管されます。

表示フォーマット

表示フォーマットを使用すると、単一ストアで、グローバル化されたマルチリンガルの顧客をベースに、販売を行うことができます。それぞれの表示フォーマットは、3 つの因子によって識別することができます。すなわち、言語、地域、および定義可能な変形です。これらの因子のいずれかが異なるグループに対して、異なるコンテンツを表示するように、サイトを設計できます。例えば、米国英語およびカナダ英語に対して、それぞれ別個のフォーマットを持つ言語およびロケールを使用することができます。これらの表示フォーマットは、テキストは同じですが、通貨と単位が異なるようにすることができます。さらに、カナダ・フランス語の表示フォーマットを追加することができます。この表示フォーマットは、通貨と単位はカナダ英語と同じように表示しますが、テキストはカナダ・フランス語で表示されるようにすることができます。 3 つの因子を使用して、同じ国/地域別環境内の特定の人々 (十代の若者、科学者、技術者など) のための、別個のフォーマットを用意して、それらのグループに適するようにサイトを調整できます。

顧客が、サイトを表示するフォーマットを選択することもできますし、サイトの作成者が、顧客のためのデフォルト値を設定することもできます。フォーマットに関する情報は、言語を変更したい場合に、URL パラメーターによって渡されます。たとえば、langId=-2 を渡すと、セッションは現在の言語をフランス語に設定します。langID はセッションに保管されます。顧客がページを要求すると、表示フォーマットが検索する Web 資産およびカタログ情報を判別します。

新規表示フォーマットの作成

次のようにして、新規表示フォーマットを作成します。

1. 次のコマンドを実行します : select * from language このコマンドは、現在使用可能な表示フォーマットごとに、使用中の言語 ID を戻します。次の使用可能な ID_VALUE を選択します。
2. 次のコマンドを実行します : insert into language (LANGUAGE_ID, ENCODING, LOCALENAME, LANGUAGE, COUNTRY) values (ID_Value, ENCODING_VALUE, 'x', 'y', 'z') ここで、

   ID_VALUE
   ステップ 2 で選択した値。

   LANGUAGE_ID (必須)
   表示フォーマットを一意的に識別する ID。

   ENCODING_VALUE (必須)
   この言語用のページを表示するために、ブラウザーが使用すべき文字エンコード値。これは、プロパティー・ファイルで使用されるのと同じエンコード値でなければなりません。 ENCODESTATEMENT = text/html; charset=[ENCODING_VALUE]。 Sun JDK によってサポートされるエンコード値のリストは、 Sun Java サイト www.java.sun.com で入手できます。

   LOCALENAME (必須)
   別個の言語およびフォーマットの習慣を持つ政治的、地理的、または文化的地域を表すために使用される Java ロケール。 localename は 2 文字の ISO 639 言語コードで、その後に下線をはさんで 2 文字の ISO 3166 の国別コードが続きます。

   LANGUAGE (オプション)
   言語名。

   COUNTRY (オプション)
   その表示フォーマットに該当する国または地域。

   VARIANT (オプション)
   変形列は、特定の国/地域別環境内のサブグループ (十代の若者、技術者、その他の種別) を記述できる余分の列です。

   たとえば、米国で話されるイタリア語の表示フォーマットを追加するには、次のステートメントを実行することができます : insert into language (LANGUAGE_ID, ENCODING, LOCALENAME, LANGUAGE, COUNTRY) values ('333', 'ISO8859-1', 'it_US', 'Italian', 'United States') 代替言語を指定することもできます。『*新規表示フォーマットの作成例*』を参照してください。
3. エントリーを LANGUAGEDS テーブルに追加してください。例として、『*新規表示フォーマットの作成例*』を参照してください。
4. エントリーを LANGPAIR テーブルに追加してください。例として、『*新規表示フォーマットの作成例*』を参照してください。
5. ストアに言語を追加します。ストアに言語を追加する方法について詳しくは、351 ページの『ストアへの言語の追加』を参照してください。

新規表示フォーマットの作成例

次の例は、FashionFlow サンプル・ストア・ページをタイ語で表示する、表示フォーマットの作成方法を示しています。

1. infashiontext_locale.properties ファイルをタイ語に翻訳します。
2. プロパティー・ファイル内のエンコード・ステートメントが、宛先ブラウザーのサポートする文字を参照することを確認します。タイ語の場合、エンコード・ステートメントは次のようになります : ENCODESTATEMENT = text/html; charset=MS874
3. ファイルを infashiontext_th_TH.properties として保管します。
4. DB2 コマンド・ウィンドウを開きます。
5. 次のコマンドを実行します : select * from language このコマンドは、現在使用可能な表示フォーマットごとに、使用中の言語 ID を戻します。次の使用可能な ID_VALUE を選択します。この例では、ID_VALUE はタイ語の場合 3 です。
6. 次のコマンドを実行します : insert into language (LANGUAGE_ID, ENCODING, LOCALENAME, LANGUAGE, COUNTRY, MIMECHARSET) values (ID_Value, ENCODING_VALUE, 'w', 'x', 'y','z') 次の値を使用します : insert into language (LANGUAGE_ID, ENCODING, LOCALENAME, LANGUAGE, COUNTRY) values ('3', 'MS874', 'th_TH', 'Thai', 'Thailand', 'MS874') ここで、

   ID_VALUE
   ステップ 2 で選択した値。

   LANGUAGE_ID (必須)
   表示フォーマットを一意的に識別する ID。

   ENCODING_VALUE (必須)
   この言語用のページを表示するために、ブラウザーが使用すべき文字エンコード値。 Sun JDK によってサポートされるエンコード値のリストは、 Sun Java サイト www.java.sun.com で入手できます。エンコード値は Sun JDK によってサポートされなければなりません。

   LOCALENAME (必須)
   別個の言語およびフォーマットの習慣を持つ政治的、地理的、または文化的地域を表すために使用される Java ロケール。 localename は 2 文字の ISO 639 言語コードで、その後に下線をはさんで 2 文字の ISO 3166 の国別コードが続きます。 ISO 言語コードについては、国際標準組織 Web サイト www.iso.ch を参照してください。

   LANGUAGE (オプション)
   言語名。

   COUNTRY (オプション)
   その表示フォーマットに該当する国または地域。

   MIMECHARSET (オプション)
   MIME メッセージングで使用する文字セット。

   VARIANT (オプション)
   変形列は、特定の国/地域別環境内のサブグループ (十代の若者、技術者、その他の種別) を記述できる余分の列です。
7. エントリーを LANGUAGEDS テーブルに追加します:
   languageds language_id=-1、description=your_language_name_in_English、

language_id_desc=-11。
例: ″languageds language_id=-1、description=French、language_id_desc=-11、languageds language_id=-11、description=Your_language_name_in_your_own_language、language_id_desc=-11、desc=-11。
例: ″languageds language_id=-11、description=Francais、language_id_desc=-11。

8. タイ語で要求されたデータが存在しない場合に使用される、タイ語の代替言語を作成します。これは、データベース内の必ずしもすべてのデータが新規言語に翻訳されるわけではない場合に役立ちます。代替言語を作成するには、次のコマンドを実行します : insert into langpair(LANGUAGE_ID, LANGUAGE_ID_ALT, SEQUENCE, STOREENT_ID) values (ID_Value, ID_Value_ALT, 'x', 'y')。次の値を使用します : insert into langpair(LANGUAGE_ID, LANGUAGE_ID_ALT, SEQUENCE , STOREENT_ID) values ('3','-1', '1' '12345')。LANGUAGE_ID は要求された言語です。 LANGUAGE_ID_ALT は代替言語です。 SEQUENCE は、要求された言語が STORELANG テーブルで指定されたとおりにサポートされるが、情報がその言語では入手できない場合に、代替言語が SEQUENCE の昇順に試行されます。ストアはその StoreGroup に指定された SEQUENCE を指定変更することができます。 STOREENT_ID は、この関係が属する StoreEntity です。ストアの代替言語関係には、その StoreGroup の代替言語関係が組み込まれています。上記の insert ステートメントは、1 番目の代替言語として英語を割り当て (language id = -1)、タイ語のデータが見つからない場合に id '12345' を持つストアで試行します。
9. プロパティー・ファイルをネイティブから ASCII に変換します : infashiontext_th_TH.properties を一時ディレクトリー (例、/tmp) にコピーします。以下のコマンドを実行します : *JDK_dir*/bin/native2ascii -encoding TIS620 /tmp/infashiontext_th_TH.properties /tmp/infashiontext_th_TH_new.properties。 この /tmp/infashiontext_th_TH_new.properties を、ディレクトリー /*WAS_userdir*/installedApps/*cell_name*/WC_instance.ear/Stores.war/WEB-INF/classes/*storeDir*/infashiontext_th_TH.properties にコピーします。ここで、*JDK_dir* は JDK へのパスです。

マルチリンガル・データ・エントリー

ブラウザーが Unicode をサポートする場合には、複数の言語を同時にブラウザーで表示することができます。 Netscape Navigator および Internet Explorer バージョン 4 以降のいずれも Unicode 表示をサポートします。ただし、ご使用のオペレーティング・システムが、ある言語でテキストを入力するために必要な文字を備えていないかもしれません。それで、入力メソッドを使用する必要があるかもしれません。入力メソッドとは、キーを押すと、直接タイプできないテキスト入力に変換されるというソフトウェア・コンポーネントです。入力メソッドは通常、日本語、中国語、韓国語、タイ語、ヒンディ語などの、標準キーボードにある文字以外の文字を持つ言語でテキストを入力するために使用されます。共通入力メソッド・ツールは Microsoft® Global Input Method Editor で、これは、Microsoft Web サイトから購入可です。 Global IME は、ショッピング・ページでデータを入力する顧客にとっても、 WebSphere Commerce アクセラレーターおよび管理コンソールでデータを入力する管理者にとっても、ふさわしいツールです。

入力メソッドを使用しないことにした場合は、多数のマシン・セットアップがあり、それぞれが異なったオペレーティング・システム言語を使用し、適正に構成さ

れたブラウザーがあれば、さまざまな言語でデータを入力することができます。ブラウザーは、それがインストールされているマシンに固有の言語を、自動的にサポートします。例えば、日本語およびドイツ語のデータを入力するには、 2 台のマシンを、1 つはドイツ語のオペレーティング・システムを使用し、もう 1 つは日本語のオペレーティング・システムを使用するようにセットアップして、それぞれがそのオペレーティング・システムからのデータを表示することのできるブラウザーを装備することができます。このケースの詳細については、ご使用のオペレーティング・システムまたは Web ブラウザーの資料を調べてください。

Unicode: WebSphere Commerce テキスト・データは、Unicode 文字セットを使用してエンコードされます。 Unicode はヨーロッパ、中東、およびアジアの言語を含む、主要な言語で使用する文字を表示できます。 WebSphere Commerce では、Unicode UTF-8 標準を使用して、同じデータベース・インスタンス内に複数の言語でデータを保管します。顧客は、WebSphere Commerce によって駆動されるサイトを表示するために、Unicode を使用できるブラウザーを必要とはしませんが、管理者は、同じマシン上で複数の言語でサイトを表示したい場合に、それを必要とするかもしれません。英語以外の言語でサイトを表示する場合には、Unicode が使用できるブラウザーが必要です。 Unicode の詳細については、Unicode の Web サイトをご覧ください。

グローバル・ストアの作成

以下のようにして、グローバル・ストアを作成します。

1. ストアを作成します。
2. テンプレートを管理します。
3. ストアに言語を追加します。
4. グローバル化されたカタログを作成します。
5. グローバル化された資産を管理します。
6. プロパティー・ファイルを翻訳します。

ストアの作成

サンプル・ストア・アーカイブのいずれかを発行して、ストアの結果を編集することによってストアを作成するか、ストアフロント、ビジネス・ロジック、またはデータ資産を別々に作成することができます。

- ストアフロント: ストアの外部を構成する部分、つまり顧客に対して表示される部分は、ストアフロントと呼ばれます。ストアフロント資産は、HTML ページ、JSP ファイル、スタイルシート、イメージ、グラフィックス、およびその他のマルチメディア・ファイル・タイプなどの Web 資産で構成されています。詳細については、 83 ページの『第 4 部 ストアフロントの開発』を参照してください。
- ビジネス・ロジック: コマンド、カスタマイズされたコードを含む顧客要求を処理するストアの部分は、ビジネス・ロジックと呼ばれます。ビジネス・ロジックとカスタマイズされたコードの作成の詳細については、「*WebSphere Commerce プログラミング・ガイドとチュートリアル*」を参照してください。
- ストア・データ: ストアを構成するデータ資産。ストアが正しく動作するためには、顧客のすべてのアクティビティーをサポートするデータがストアに配置されていなければなりません。たとえば、顧客が商品を購入するためには、販売商品

のカタログ、オーダー処理のためのプロセス、要求を実行する在庫機能、および配送プロセスが必要です。さらに、支払を処理し、集金するための手段も必要です。ストア・データの作成に関係するさまざまな概念や作業については、 123 ページの『第 6 部 ストア・データの開発』で説明します。

サンプル・ストア・アーカイブの発行について詳しくは、 WebSphere Commerce オンライン・ヘルプのトピック『ストア・アーカイブの発行』を参照してください。

グローバル化されたサイト用のテンプレートの管理

グローバル化されたサイトの静的ページおよび動的テンプレートを管理するには、ファイルをディレクトリー構造に保管することが必要です。そうするなら、ファイルおよびそれが属するロケールを、すばやく簡単に識別することができます。

ファイル・ディレクトリー・パスは、WebSphere Commerce インスタンス、ストア・プロファイル内に含まれる Business ストア・パス、および登録済みファイル・パスに基づいて構成されます。グローバル化されたサイトを作成する際には、複数のストアを作成して、それぞれがそのサイトのサポートされる配送先範囲を表すようにし、またそれぞれがサポートされる言語のリストを備えるようにします。テンプレート・ファイルはサイトのルック・アンド・フィールに影響を与えるので、ロケール固有のディレクトリー下に保管されます。したがって、リソース・バンドルの選択と同様の方法でそれらを選択することができます (ロケール値を使用する)。システムが特定の言語形式に使用するテンプレートを選択する場合、ファイルの検索元のディレクトリーを判別するために使用する言語形式を判別するのに、ロケールが使用されます。

グローバル化された環境におけるテンプレートの保管モデルが 3 つあります。

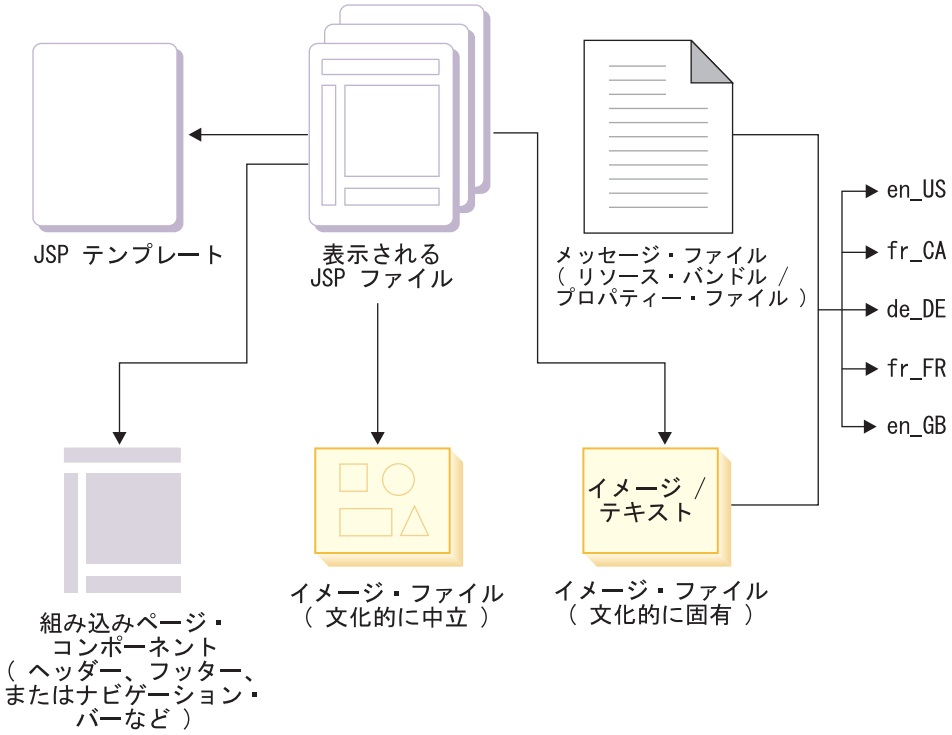
表 14. グローバル化された環境におけるテンプレートの保管

	すべてのストアおよび言語に対して 1 つのテンプレート	**言語ごとに 1 つのテンプレート**	**ストアごとに 1 つのテンプレート**
カスタマイズ	ほとんどのストアに対して、各ストアおよび各ストア言語形式間で、十分のレベルのカスタマイズを提供する。	各ストアおよび各ストア言語形式間で、最大レベルのカスタマイズを可能にする。	各ストア間で、いくらかのレベルのカスタマイズを可能にする。
ページのルック・アンド・フィール	ページが同じように見える。	ページが非常に異なることがある。	ページの一般レイアウトが同じ。

表 14. グローバル化された環境におけるテンプレートの保管 (続き)

保守	1 つのテンプレートだけ変更すればよいので、サイト全体のページ設計の変更が簡単。ほとんどのグローバル化されたサイトの場合、このモデルは保守容易性および拡張容易性の最適なレベルを提供する。	各テンプレートの複数のコピーを管理する必要がある。すべてのストアまたはすべての言語形式に影響を与える変更は、すべてのテンプレートに対して行う必要がある。	JSP ファイルのルックに対するサイト全体の変更は、複数のテンプレートで行う必要がある。
使用する場合	各ストアおよび各言語ごとのルック・アンド・フィールが非常に似ている場合。	ページのルック・アンド・フィールおよびコンテンツが、言語によって非常に異なる場合。この場合には、異なる言語間で共用できる部分はあまりないので、それぞれの言語ごとに別個のページを作成するほうが簡単。	ストアによってルック・アンド・フィールはかなり異なるが、ストアのルック・アンド・フィールは、言語によってそれほど異ならない場合。
使用しない場合	ストアおよび言語によって、サイトのルックが非常に異なるようにする場合には、使用しない。	ストア間および言語形式間でページが非常に似ている場合には、使用しない。	ストアのルック・アンド・フィールが非常に似ている場合には、使用しない。
プロパティー・ファイル	必須。サポートされる各言語には、ページの生成時に組み込まれる、独自のプロパティー・ファイルもある。	不要。ストアおよびロケールの各組み合わせには、独自の JavaServer Page テンプレートがある。	必須。各言語形式間でテンプレートを共用できるようにするため。
ショッピング・フロー	言語およびストア間で、ショッピング・フローは同じ。	ショッピング・フローは、言語によってかなり変えることができる。	言語およびストア間で、ショッピング・フローは同じ。

すべてのストアおよび言語に対して 1 つのテンプレート・プログラミング・モデル



すべてのストアおよび言語に対して 1 つのテンプレート・プログラミング・モデルでは、各ページは、基本ページ・レイアウトおよび国/地域別環境に依存しないデータおよびイメージを含む、単一の JavaServer Page テンプレートから構成されます。このテンプレートは、顧客が選択した表示フォーマットに基づいて、国/地域別環境に依存するコンポーネントと実行時に結合されます。ページの設計に対する変更は、サポートされる国/地域別環境の数に関係なく、一度行うだけですみます。言語または国/地域別環境の追加または除去は簡単です。なぜなら、国/地域別環境に依存するものは、ページの他のフィーチャーとは分離されているからです。

次の表は、ファイルが編成される仕方を示しています。 *webapp* は、サイトまたはアプリケーションのルート・ディレクトリーのことです。そのディレクトリー下に、共通ディレクトリー、およびそれぞれの表示フォーマットごとか、またはサポートされる言語とロケールの組み合わせごとのディレクトリーを置くことができます。実際にどのような構造にするかは、ご自分で決定することができます。実行時に、テンプレートはコマンド・コンテキストからの言語およびロケール情報を使用して、プロパティー・ファイル、イメージ・ファイル、その他の国/地域別環境に依存したコンテンツを検索するフォルダーを判別します。例えば、コマンド・コンテキストが en_US 表示フォーマットを示す場合には、site_root/en_US/sensitivetext.properties ファイルを使用し、 site_root/en_US/images/ ディレクトリーからイメージを検索します。

表 15. すべてのストアおよび言語に対して 1 つのテンプレート・プログラミング・モデル

テンプレート	/webapp/common/web/template/template.jsp すべての表示フォーマットに対して、同じテンプレートが使用されます。
組み込みページ・コンポーネント	/webapp/common/web/template/header.jsp /webapp/common/web/template/footer.jsp 共通ページ・コンポーネントがこのディレクトリーにあります。
国/地域別環境に依存しないイメージ・ファイル	/webapp/common/web/images/image.gif イメージが共通ディレクトリー内にあり、すべての表示フォーマットに対して使用されます。
プロパティー・ファイル (プロパティー・ファイルを保管する方法を、次のうちのいずれかから選択する)	/webapp/language_LocaleA/web/sensitivetext.properties /webapp/language_LocaleB/web/sensitivetext.properties それぞれの表示フォーマットには、別個のプロパティー・ファイルがあります。異なる表示フォーマット用のプロパティー・ファイルは、同じ基底名を持ち、その名前のファイル拡張子の前に、ロケール・サフィックス xx_XX が付加されています。それらは同じディレクトリーにあります。ディレクトリー名は、LANGUAGE テーブルの LOCALENAME 列に表示された、 *Language_Locale* の組み合わせに基づいています。この方法の例として、サンプル・ストアを参照してください。**または**、次の方法を使用することができます : /webapp/properties/sensitivetext_Language_LocaleA.properties /webapp/properties/sensitivetext_Language_LocaleB.properties 複数のプロパティー・ファイルが単一ディレクトリー内に保管されていますが、それぞれにロケール固有のファイル名があります。
国/地域別環境に依存するイメージ・ファイル	/webapp/language_LocaleA/web/images/image.gif /webapp/language_LocaleB/web/images/image.gif それぞれの表示フォーマットに対して、別個の翻訳済みイメージが保管されています。ファイル名は同じですが、それぞれ適用される表示フォーマットの名前に対応した別個のディレクトリーに配置されています。 *Language_Locale* の組み合わせは、 LANGUAGE テーブルの LOCALENAME 列に表示された表示フォーマットを表します。

すべてのストアおよび言語に対して 1 つのテンプレートのディレクトリー構造

このモデルは、各ストアおよび各言語ごとのルック・アンド・フィールが非常に似ている場合に適しています。 1 セットの JSP テンプレートを保守するだけで済みますが、一連のプロパティー・ファイルを管理する必要があります。この方法は、保守のためのテンプレート管理モデルで、 1 つのテンプレートを変更するだけでよいので、サイト全体のページ設計の変更を可能にします。

例えば、2 つのストア・ロケーションがあり、それぞれ米国英語とカナダ・フランス語を表示する場合には、 JSP テンプレートを次のように編成することができます : /webapp/common/web/template/abc.jsp

この JSP テンプレートのプロパティー・ファイルのパスは、次のように保管されます : /webapp/common/web/properties/en_US/abc.properties
/webapp/common/web/properties/fr_CA/abc.properties
この場合、JSP ファイルの登録時に、ファイル・タイプだけをファイル・レジストリーに組み込む必要があります。この方法を使用すれば、すべてのストアおよびす

べてのロケールに対して、 JSP ファイルを 1 セットだけ登録すれば済みます。ここで、プロパティー・ファイルは、国/地域別環境に依存した情報が含まれているので、別個に保管する必要があります。一方、テンプレート自体は、国/地域別環境に全く依存しないので、共通ディレクトリーに保管されます。このテンプレート管理ストラテジーの作動例については、サンプル・ストアを参照してください。

言語ごとに 1 つのテンプレートのディレクトリー構造

このモデルを使用するには、各ストアのテンプレート・ディレクトリー内に、そのストアがサポートするそれぞれの言語ごとに別個のディレクトリーを作成する必要があります。これらの言語依存ディレクトリー内のそれぞれに、それぞれ異なるテンプレートを保管しなければなりません。このモデルではプロパティー・ファイルは不要です。なぜなら、ストアおよびロケールの各組み合わせには、独自の JavaServer Page テンプレートがあるからです。

例えば、2 つのストア・ロケーションがあり、それぞれ米国英語とカナダ・フランス語を表示する場合には、 JSP テンプレートを次のように編成することができます : /webapp/StoreA/web/template/en_US/abc.jsp /webapp/StoreA/web/template/fr_CA/abc.jsp /webapp/StoreB/web/template/en_US/abc.jsp /webapp/StoreB/web/template/fr_CA/abc.jsp この場合、JSP テンプレートの登録時に、ロケールおよびファイル・タイプをファイル・レジストリーに組み込む必要があります。それぞれのストアおよびロケールには、登録済みテンプレートの完全なセットが必要です。

ストアごとに 1 つのテンプレートのディレクトリー構造

このモデルの JSP テンプレートは、ストア内で共用されます。ただし共用は、単一ストアに限られます。このモデルでは、言語形式間でテンプレートを共用できるようにするための、JSP インクルード・ファイルが必要です。

例えば、2 つのストア・ロケーションがあり、それぞれ米国英語とカナダ・フランス語を表示する場合、 JSP テンプレートを次のように編成することができます : /webapp/StoreA/web/template/abc.jsp /webapp/StoreB/web/template/abc.jsp

このテンプレート管理モデル内のプロパティー・ファイルのパスは、次の例のようなものになります : /webapp/StoreA/web/properties/en_US/abc.properties
/webapp/StoreA/web/properties/fr_CA/abc.properties
/webapp/StoreB/web/properties/en_US/abc.properties
/webapp/StoreB/web/properties/fr_CA/abc.properties

このモデルの JSP テンプレートの登録時には、ファイル・タイプだけをファイル・レジストリーに組み込む必要があります。各ストアは、独自の JSP ファイルの完全なリストを登録する必要があります。

ストアへの言語の追加

次のようにして、既存のストアに新規言語サポートを追加します。

1. その言語がサイトで使用できることを確認します。サポートされる 10 の言語のリストは、『マルチリンガル・サポート』で参照してください。その言語が使用できる場合には、ステップ 3 に進み、使用できない場合には、次のステップに進んでください。

2. その言語の新規表示フォーマットを作成します。ストア・プロファイル・ノートブックを使用して、ストアのサポートする言語リストに、その言語を追加します。
3. 各国語ファイル (たとえば FashionFlow の場合には infashiontext_locale.properties) を、次のロケーションにコピーします : *AppServer*/installedApps/*host*/WC_demo1.ear/Stores.war/WEB-INF/classes/storeDir tax.xml、store.xml、fulfillment.xml、catalog.xml、businesspolicy.xml、contract.xml、accesscontrol.xml、shipping.xml などのローダーを使用して翻訳および移植する必要のある XML ファイルは他にも多くあります。

グローバル化されたオンライン・カタログの作成

グローバル化されたサイトに適した柔軟なオンライン・カタログを作成するには、それぞれの商品に対して、サポートしたい各言語ごとまたは各国/地域別環境ごとに 1 つずつ、複数の詳細情報を組み込みます。国/地域別環境が異なれば、単に言語だけでなく、それ以上のものもしばしば異なることを考慮に入れてください。例えば、ある特定のデータを表現する仕方が異なるかもしれません。例えば、ある国/地域別環境では 10 進数はコンマを使用して表示し、他の国/地域別環境ではピリオドを使用して表示します。

1. ストアがサポートするそれぞれの言語ごとに、カタログを作成する必要があります。次のうちのいずれか 1 つのカタログ作成方法を選択します。
   - ローダー・パッケージ、またはユーザーの選択するカタログ・ツールを使用してカタログを作成する。
   - カタログ・データを XML ファイルで作成し、それをローダー・パッケージを使用してデータベースにロードするか、または管理コンソールを使用してストア・アーカイブ形式でそれを発行します。詳細については、『カタログの作成』を参照してください。
   - 既存のカタログを、ローダーでの使用に適した XML ファイル・フォーマットに変換した後、情報をデータベースにロードします。詳細については、『ローダー・パッケージ』を参照してください。
   - サンプル・ストア・カタログをベースとして使用してカタログを作成し、商品管理ツールを使用して情報を変更します (小さい変更を行う場合に、これが有効です)。
2. 作成するそれぞれのカタログごとに、次のタイプの情報を表示する方法を考慮してください。

   オンライン・カタログ商品
   それぞれのカタログ・エントリーに対して、複数言語の説明があります。

   商品説明
   説明の言語および句は、異なる顧客グループに対して異なるフィーチャーを強調するように変更することができます。

   価格 価格は料金および他の配送費用を反映するように変更することができ、また異なる通貨で表示することができます。

   国/地域別環境形式
   日付、名前、計算単位、その他のデータは、国/地域別環境に適するように形式設定することができます。

商品イメージ
異なる顧客に対して異なる商品イメージを表示することができます。

グローバリゼーション資産の管理

Web 資産を管理するため、すべてのストアおよび言語に対して 1 つの JSP テンプレート (各ページの基本設計と国/地域別環境に依存しない情報を組み込んでいる) を使用する、グローバル化されたプログラミング・モデルを採用することをお勧めします。他の国/地域別環境に依存したテキストは、リソース・バンドルまたはプロパティー・ファイルを使用して、実行時にページに追加されます。

どの Web 資産を翻訳するかを判別してください。このリストには、次のものが含まれます。ページに表示されるバナー、イメージ、アプレット、テキスト、メッセージ、その他の国/地域別環境に依存したコンテンツ。これらのコンポーネントのいくつかは、サイトでサポートされるそれぞれの言語または国/地域別環境ごとに 1 つずつ、複数のバージョンを作成します。グローバル・ストアでの資産の管理方法の例については、サンプル・ストアを参照してください。

プロパティー・ファイルの翻訳

次のようにして、プロパティー・ファイルを翻訳します。

1. 任意のテキスト・エディターを使用して、プロパティー・ファイルを開きます。
2. 次の事柄に注意して、プロパティー・ファイル内のテキストを翻訳します。
   - キーワードは翻訳しないでください。キーワードは、等号の左のコンテンツです。翻訳元 : lastName.Label=Last
     新規翻訳 : lastName.Label=Nom de famille
   - オプション属性については、セミコロンの右の値だけを翻訳します。翻訳元 : title.Options=MR;Mr.|MRS;Mrs.|MS;Ms.
     翻訳 : title.Options=MR;M.|MRS;Mme.|MS;Mlle. 翻訳元 : publishPhone.Options=Y;Yes|N;No
     翻訳 : publishPhone.Options=Y;Oui|N;Non
   - オプションで、コメント、つまりポンド記号 (#) で始まる行を翻訳してください。
3. プロパティー・ファイルをテキストとして保管します。プログラミング・モデルを使用している場合には、
   a. プロパティー・ファイルは同じ名前だが、ロケール固有のディレクトリーに保管されているならば、
      - ファイルを正確なディレクトリーに保管します。
   b. プロパティー・ファイルは同じディレクトリーに保管されているが、ロケールが名前に付加されるならば、
      - 適切なロケールをファイル名に付加します。拡張子は .properties でなければなりません。
4. プロパティー・ファイルに Latin 1 でも Unicode でもない文字が含まれている場合には、 native2ascii コンバーターを使用して、データを非 ASCII フォーマットから Unicode ascii 表記に変換してください。この処理は、プロパティー・ファイルに含まれるデータを、プラットフォームから独立したものにします。native2ascii コンバーターは、次のディレクトリーにあります :

WC_installdir¥jdk¥bin

400 *WC_installdir*/Java400/jdk13/bin

native2ascii コンバーターの追加情報については、次のサイトを参照してください：
www.java.sun.com

第 9 部 ストアのパッケージ化

第 35 章 ストアのパッケージ化

ストアの使用目的が、これをサンプルとして他人に送付すること、管理コンソールで発行ユーティリティーを使用して発行すること、または別のサーバーまたは別のプラットフォームにデプロイすることである場合、これをストア・アーカイブの形でパッケージ化できます。

一般に、ストア・アーカイブは以下のファイルで構成されます。

- Web 資産: HTML ファイル、JSP ファイル、イメージ、グラフィックス、および組み込みファイルなど、ストア・ページの作成に使用するファイル。
- プロパティー・リソース・バンドル : ストア・ページのテキストが入っています。ストアが複数の言語をサポートする場合、ストア・アーカイブには、サポートされる言語ごとに複数のリソース・バンドル、さらにデフォルトのリソース・バンドル (ロケールを含まない) が入っています。たとえば、`AddressText_en_US.properties` および `AddressText.properties` などです。
- ストア・データ資産: データベースにロードされるデータ。ストア・データ資産には、キャンペーン、カタログ・エントリー、通貨、フルフィルメント情報、価格設定、配送、ストア、税情報などのデータが含まれます。ストア・データ資産の詳細なリストについては、 123 ページの『第 6 部 ストア・データの開発』を参照してください。

  WebSphere Commerce で提供されるサンプルのストア・アーカイブにあるストア・データ資産は、ローダー・パッケージに有効な、整形式の XML ファイルの一部です。ストア・データ資産 XML ファイルは移植可能であることが意図されており、データベースの特定のインスタンスに固有な、生成された基本キーを含めるべきではありません。その代わりに、ストア発行時に ID リゾルバーによって解決される内部別名が使用されます。これらの規則を使用すると、サンプルのストア・アーカイブを移植可能にすることができます。詳細については、 355 ページの『第 9 部 ストアのパッケージ化』を参照してください。

  ローダー・パッケージの詳細については、 379 ページの『第 37 章 ストア・データのロードの概要』を参照してください。

  注: ストア・データ資産には、契約を作成するために必要な情報を提供する、契約情報も含まれます。契約情報はローダー・パッケージによってロードされません。この情報により契約を作成するコマンドに対する入力データが備えられます。
- 支払資産: WebSphere Commerce Payments の構成情報です。支払い情報はローダー・パッケージによってロードされません。この情報により WebSphere Commerce Payments を構成するコマンドに対する入力データが備えられます。
- 記述子: ストア・アーカイブおよびその発行方法について記述する XML ファイルです。これらのファイルには、`store-refs.xml`、`ibm-wc-load.xml`、`unpack.xml`、および `ForeignKeys.dtd` が含まれます。

サンプル・ストア・アーカイブに含まれるファイルは、次のような構造に分類されます。

- *ストア・ディレクトリー*
  - JSP ファイル、HTML: 機能領域によってサブディレクトリーに分類されます。たとえば、ShoppingArea、AuctionArea、CustomerServiceArea などです。これらの各領域は、セクションに分類されます。たとえば、ShoppingArea は CatalogSection、CheckoutSection、DiscountSection および ShopcartSection に分割されます。これらの各セクションも、必要ならさらにセクションに分割できます。
  - イメージ: サンプル・ストア・アーカイブに含まれるイメージは、ロケールによって分類されます。
- SAR-INF
  - 次のファイルを含む、このストア・アーカイブの発行に特定の情報が入っています。
    - `store-refs.xml`: このストア・アーカイブで使用される発行パラメーターを定義します。
    - プロパティー・ファイル: このストア・アーカイブの発行パラメーターを記述するのに使用されるテキスト。サンプル・ストア・アーカイブには、このプロパティー・ファイルのロケールに特定のバージョンも含まれます。
    - `unpack.xml`: アンパックされるストア・アーカイブ中の資産、アンパックの方法、およびアンパック先を決定します。

  注: `store-refs.xml` および `unpack.xml` ファイルに関する詳細は、365 ページの『第 36 章 ストア全体の発行』を参照してください。
- WEB-INF
  - プロパティー・ファイル: ストアのプロパティー・ファイルは、 WEB-INF では次のディレクトリー構造にあります。
    - Classes
      - *Store directory*
  - ストア・データ資産: ストアのデータ資産。XML 形式で、 WEB-INF では次のディレクトリー構造にあります。
    - ストア

      注: 複合ストア・アーカイブには、追加のビジネス・モデル・ディレクトリーが含まれます。

      - *ストア・ディレクトリー*
        - データ: ストアに複数の言語が入っている場合、ロケールに特定の XML ファイルはロケールによって分類されます。データ・ディレクトリーには、ストア・アーカイブの発行に必要な次のファイルも入っています。
          - `ForeignKeys.dtd`: 発行パラメーター値を保管します。これらのパラメーターは、ストア・データ資産 XML によって参照されるエンティティー名前値のペアです。
          - `ibm-wc-load.xml`: データのロードを制御します。
          - `store-data-assets.xml`: ロードされるすべてのストア・データを含む XML ファイル。

注:

1. ForeignKeys.dtd および ibm-wc-load.xml ファイルに関する詳細は、 365 ページの『第 36 章 ストア全体の発行』を参照してください。
2. XML ファイル形式の支払構成情報も、データ・ディレクトリーに入っています。

– ストア・フロー変更用のファイル: ストアのフローが WebSphere Commerce アクセラレーターのストア・ツールを使用して変更できる場合、必要なファイルは次のディレクトリー構造にあります。

- xml
  - tools
    – stores
      - *Store directory*
        - devtools
          – flow

ストア・アーカイブの作成

ストア・アーカイブのパッケージ構造は柔軟です。ここで説明する指示は、WebSphere Commerce で提供されるサンプルの構造を反映しますが、この構造は必要に応じて変更することができます。ストア・アーカイブには、store-refs.xml ファイルおよび unpack.xml ファイルが入っている SAR-INF ディレクトリーが含まれていなければなりません。これらの 2 つのファイルで定義されるパスは、ストア・アーカイブの構造と一貫している必要があります。

ストアをストア・アーカイブとしてパッケージ化するには、次のようにします。

1. WebSphere Commerce に含まれているサンプル・ストア・アーカイブの構造や内容を調べます。

   ストア・アーカイブ・ファイルは以下のディレクトリーにあります。

   - *WC_installdir*/samplestores/*businessmodel*

   ストア・アーカイブを表示するには、解凍プログラムを使います。
2. ストアの WebSphere Commerce Server に、一時ディレクトリーを作成します。たとえば、*mystore* とします。
3. 次のサブディレクトリーを作成します。
   - *Store directory* (ストアの名前)
   - SAR-INF
   - WEB-INF
4. *Store directory* ディレクトリーで以下のようにします。
   - 機能領域で、JSP ファイルのサブディレクトリーを作成します。サンプル・ストア・アーカイブの例を参照してください。 JSP ファイル、および必要な HTML ファイルがあれば、これらのサブディレクトリーにコピーします。
   - イメージ・ファイルにサブディレクトリーを作成します。ストアで複数の言語をサポートする場合は、ロケール名を使用することにより、言語特有の情報を

入れるためのサブディレクトリーを作成します。たとえば、en_US のようにします。イメージ・ファイルをこれらのサブディレクトリーにコピーします。

5. SAR-INF ディレクトリーで以下のようにします。
   a. ストア・アーカイブのための store-refs.xml ファイルを作成します。サンプル・ストア・アーカイブの既存の store-refs.xml ファイルを例として使用して、自分のストア用に store-refs.xml ファイルを作成します。 XML の仕様については、以下に示すディレクトリーに含まれているディスクリプター store-refs.dtd を参照してください。
      - *WC_installdir*/xml/sar

      注: store-refs.xml ファイルについての詳細は、 365 ページの『第 36 章 ストア全体の発行』を参照してください。
   b. (オプション) ストア・アーカイブのパラメーターを、管理コンソールからの発行中にユーザーが選択できるようにしたい場合、これらのパラメーターを記述するプロパティー・ファイルを作成します。このファイルを、properties という名前のサブディレクトリーに保存します。
   c. サンプル・ストア・アーカイブの既存の unpack.xml ファイルを例として使用して、自分のストア用に unpack.xml ファイルを作成します。 unpack.xml ファイルは、ストア・アーカイブのアンパック方法を決定します。 XML の仕様については、以下に示すディレクトリーに含まれているディスクリプター unpack.dtd を参照してください。
      - *WC_installdir*/xml/sar

   注: unpack.xml ファイルについての詳細は、 365 ページの『第 36 章 ストア全体の発行』を参照してください。
6. WEB-INF ディレクトリーで以下のようにします。
   a. ストアのプロパティー・ファイルに、次のサブディレクトリー構造を作成します。
      - classes
        - *Store directory*
   b. プロパティー・ファイルを *Store directory* にコピーします。
   c. ストアのデータ資産に、次のサブディレクトリー構造を作成します。
      - stores
        - *Store directory*
          - データ: ストアで複数の言語をサポートする場合は、ロケール名を使用することにより、言語特有の情報を入れるためのサブディレクトリーを作成します。たとえば、en_US のようにします。
   d. データ資産をデータ・ディレクトリーおよび対応するサブディレクトリーにコピーします。
   e. サンプル・ストア・アーカイブの既存の ibm-wc-load.xml ファイルを例として使用して、自分のストア用に ibm-wc-load.xml ファイルを作成します。 ibm-wc-load.xml ファイルは、ストア・データのロード方法を決定します。 XML の仕様については、以下に示すディレクトリーに含まれているディスクリプター ibm-wc-load を参照してください。
      - *WC_installdir*/xml/sar

注: ibm-wc-load.xml ファイルについての詳細は、 365 ページの『第 36 章 ストア全体の発行』を参照してください。

f. (オプション) ストア・アーカイブのパラメーターを、管理コンソールからの発行中にユーザーが選択できるようにしたい場合、これらのパラメーターの値を保管する ForeignKeys.dtd ファイルを作成します。サンプル・ストア・アーカイブの既存の ForeignKeys.dtd ファイルを例として使用して、自分のストア用に ForeignKeys.dtd ファイルを作成します。

注: ForeignKeys.dtd ファイルについての詳細は、 365 ページの『第 36 章 ストア全体の発行』を参照してください。

7. *Store directory*、SAR—INF ディレクトリー、および WEB-INF ディレクトリーから構成される ZIP ファイルを作成します。その ZIP ファイルの名前は *storearchivename.sar* とします。
8. 管理コンソールを使用してストア・アーカイブを発行したい場合は、 377 ページの『管理コンソールに対してストア・アーカイブを使用可能にする』を参照してください。

サンプル・ストア・アーカイブの作成

ストアをストア・アーカイブとしてパッケージ化したなら、管理コンソールにおいてそれをサンプル・ストアとして使用できます。ストア・アーカイブをサンプル・ストア・アーカイブとしてパッケージ化するには、次のようにします。

1. ストア・アーカイブ・ファイルを、以下に示すディレクトリーに保存します。
   - *WC_installdir*/samplestores
2. (オプション) プレビュー・ページを作成します。ストア・ページのプレビューを管理コンソールに表示するためには、まずプレビュー・ページを作成する必要があります。以下のようにします。
   a. (オプション) 管理コンソールの発行ユーティリティーで、発行するストア・アーカイブを選択してから **「プレビュー (Preview)」**をクリックします。表示されるページは、プレビュー・ページと呼ばれます。それらのページは、事前に定義された買い物の流れのサンプルを示すものであり、サンプル・ストアのプレビューとして使用される HTML ファイルです。
   b. プレビュー・ページに表示する買い物の流れを決定します。
   c. (オプション) 発行済みストアの中にサンプル・データを作成します。たとえば、アイテムをショッピング・カートに追加し、いくつかの配送先住所と請求先住所を作成します。このストアからプレビュー・ページを作成し、データを入れることによってページがリアルなものになります。
   d. Internet Explorer を使用して、ストアを表示します。ページごとに「ファイル」→「名前を付けて保存」を選択することにより、 HTML を保存します。さらに、スタイルシート (.css) とイメージも保存する必要があります。それらのファイルは、以下に示すディレクトリーに保存します。
      - *stylesheet*.css
        - *WC_installdir*/wc.ear/SiteAdministration.war/tools/devtools/preview/*locale*/*businessmodel*/*storedir*/
      - HTML

- *WC_installdir*/wc.ear/SiteAdministration.war/tools/devtools/preview/*locale*/*businessmodel*/*storedir*

- ロケールに依存しないイメージ
  - *WC_installdir*/wc.ear/SiteAdministration.war/tools/devtools/preview/images/*businessmodel*/*storedir*
- ロケールに依存するイメージ
  - *WC_installdir*/wc.ear/SiteAdministration.war/tools/devtools/preview/*locale*/*businessmodel*/*storedir*/images

e. イメージと `css` ファイルの位置が変更されているため、 HTML ページに含まれるイメージと `css` ファイルへの参照を変更する必要があります。参照を変更したなら、 HTML ページをブラウザーで開いた場合にイメージが正常に表示されることを確認してください。

f. HTML ページに含まれるリンクを、コマンドではなく、HTML ファイルを参照するリンクに変更します。

3. 発行ユーティリティーが管理コンソールで表示されるように、ストア・アーカイブを `sarregistry.xml` ファイルに追加します。詳細については、 377 ページの『管理コンソールに対してストア・アーカイブを使用可能にする』を参照してください。

第 10 部 ストアの発行

機能するストアを作成するには、ストアフロント Web 資産を WebSphere Commerce Server に発行し、ストア・データを WebSphere Commerce データベースに発行する必要があります。

ここに含まれる章では、WebSphere Commerce に備わっている発行オプションについて説明します。

- 365 ページの『第 36 章 ストア全体の発行』 - この章では、ストアがストア・アーカイブのフォームになっている場合に、管理コンソールかコマンド行の発行ユーティリティーを使用してストア全体 (ストアフロントとストア・データ資産) を発行する方法について説明します。
- 379 ページの『第 37 章 ストア・データのロードの概要』 - この章では、ローダー・パッケージを使用してストア・データ資産をデータベースに発行する方法について説明します。
- 429 ページの『第 38 章 WebSphere Commerce データベース資産グループのロード』 - この章では、ローダー・パッケージを使用してストア・データ資産のグループやストア・データ全体をデータベースに発行する方法について説明します。
- 443 ページの『第 39 章 ビジネス・アカウントと契約の発行』 - この章では、アカウント、契約、および商品セット資産の発行について説明します。
- 447 ページの『第 40 章 ストアフロント資産とストア構成ファイルの発行』 - この章では、ストアフロント資産とストア構成ファイルの発行について説明します。

第 36 章 ストア全体の発行

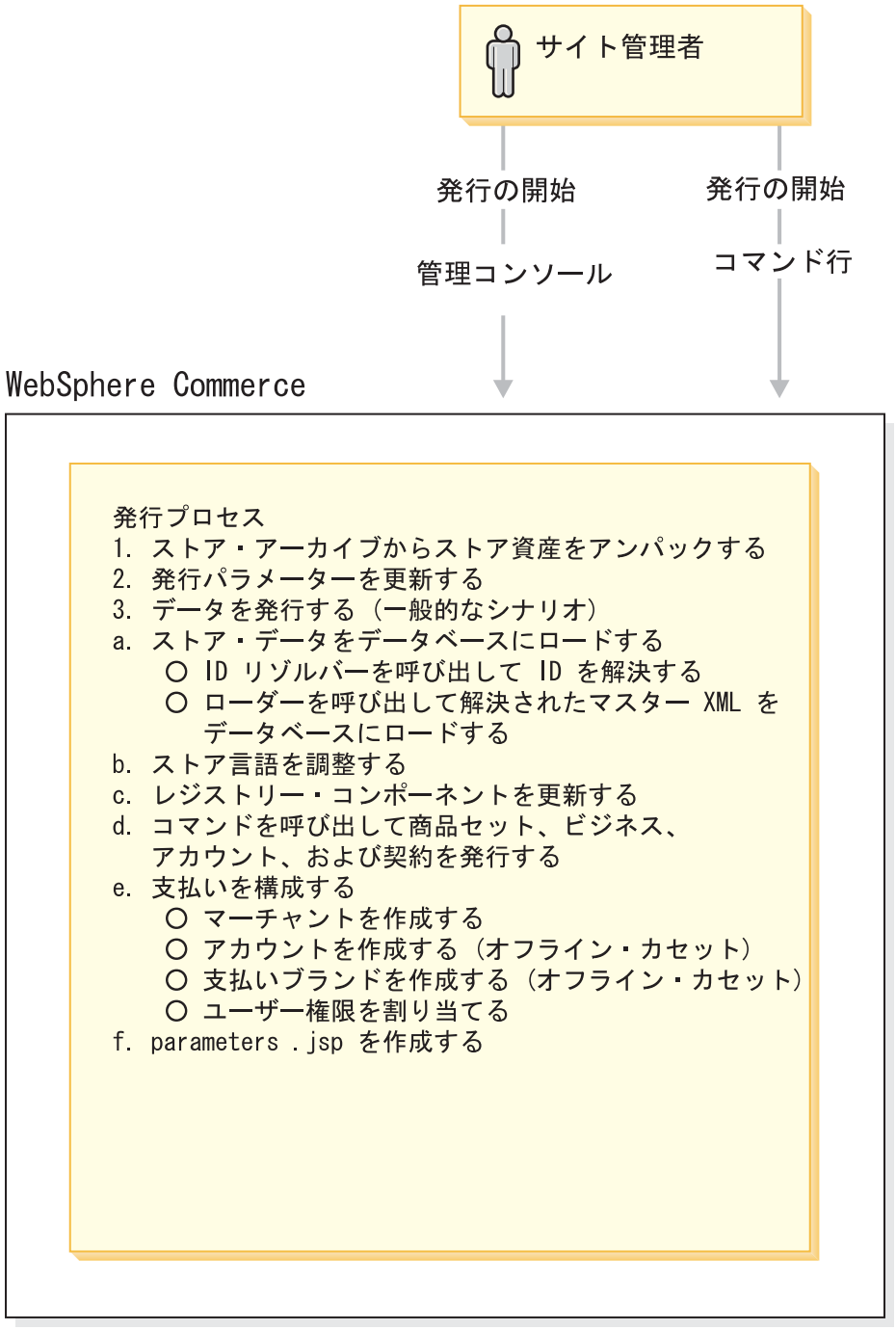
機能するストアを作成するには、ストアフロント Web 資産を WebSphere Commerce Server に発行し、ストア・データを WebSphere Commerce データベースに発行する必要があります。この章では、ストアがストア・アーカイブのフォームになっている場合に、管理コンソールかコマンド行の発行ユーティリティーを使用してストア全体 (ストアフロントとストア・データ資産) を発行する方法について説明します。

注: ストアをストア・アーカイブとしてパッケージしないほうが好ましい場合は、個々の資産を 1 つずつ発行することができます。詳細については、 379 ページの『第 37 章 ストア・データのロードの概要』、 429 ページの『第 38 章 WebSphere Commerce データベース資産グループのロード』、 443 ページの『第 39 章 ビジネス・アカウントと契約の発行』、 447 ページの『第 40 章 ストアフロント資産とストア構成ファイルの発行』を参照してください。

WebSphere Commerce での発行について

管理コンソールやコマンド行で使用できる発行オプションを使うと、ストア全体 (ストアフロントとストアのデータ資産) を一度にまとめて発行できます。このオプションを使用するためには、ストア資産がストア・アーカイブのフォームでパッケージされていなければなりません。ストアをストア・アーカイブとしてパッケージする処理については、 355 ページの『第 9 部 ストアのパッケージ化』を参照してください。

次の図は、発行プロセスの手順を簡単にまとめたものです。



発行の開始

ストアを発行するためには、ストア管理者権限が必要です。サイト管理者は、次のいずれかの方法を使用して、発行のプロセスを開始することができます。

- 管理コンソール
- コマンド行

どちらの方法で発行を行う場合も、発行したいストア・アーカイブの指定が必要です。

注: 管理コンソールを使用してストア・アーカイブを発行するには、管理コンソールが正しい位置にあるか、登録済みでなければなりません。詳細については、377 ページの『管理コンソールに対してストア・アーカイブを使用可能にする』を参照してください。

その後、可能であれば、ストア ID (固有にストアを識別する名前)、ストア・ディレクトリー (JSP ファイルおよびイメージの発行先の固有の位置)、および組織 (ストア・アーカイブの発行先の組織) を含む、選択されたパラメーターの値を変更することができます。

管理コンソールの発行パラメーター

管理コンソールの発行ユーティリティーにある発行パラメーターは、各ストア・アーカイブの store-refs.xml ファイルによって定義されます。

ConsumerDirectStore.sar ファイルからの store-refs.xml の次の例をご覧ください。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE store-refs SYSTEM "store-refs.dtd">
<store-refs
    target-dtd="WEB-INF/stores/FashionFlow/data/ForeignKeys.dtd"
    deploy-descriptor="WEB-INF/stores/FashionFlow/data/ibm-wc-load.xml"
 resource-bundle="/SAR-INF/properties/publishNLS">
ref id="storeDir" entity="STORE_DIR" >
        <input type="text"/>
</ref>
<ref id="storeIdent" entity="STORE_IDENTIFIER" >
        input type="text"/>
</ref>
<ref id = "parentOrg" entity="ORGANIZATION_DN">
 <input type="member" />
</ref>
</store-refs>
```

このファイルは、ConsumerDirectStore.sar ファイルの 3 つの発行パラメーターを定義します。

- `<ref id="storeDir" entity="STORE_DIR" >`
  `<input type="text"/>`

  このエンティティーは、発行パラメーターのストア・ディレクトリーを作成します。

- `<ref id="storeIdent" entity="STORE_IDENTIFIER" >`
  `< input type="text"/>`

  このエンティティーは、発行パラメーターのストア ID を作成します。

- `<ref id = "parentOrg" entity="ORGANIZATION_DN">`
  `<input type="member" />`

  このエンティティーは、発行パラメーターの組織を作成します。

ここで

- ref id は、ストア参照リソース・バンドル属性が指定するプロパティー・ファイルでキーとして使用されます。これは、変換可能なパラメーター名、および発行パラメーター・ページに表示される説明を取得するのに使用されます。

- entity は、このパラメーターが編集する target-dtd の ENTITY の名前です。
- input type は、パラメーターが画面に表示される方法を制御します。入力タイプがテキストである場合、パラメーターは編集可能フィールドに表示されます。入力タイプがメンバーである場合、すべての既存の組織はドロップダウン・リストに表示されます。読み取り専用パラメーターは編集できません。

これらのパラメーターにユーザーが入力する値は、target-dtd ファイルで識別されるファイルに保管されます。 target dtd は以下のコードで定義されます。

```
target-dtd="WEB-INF/stores/ConsumerDirect/data/ForeignKeys.dtd
```

このファイルもストア・アーカイブの一部で、ストア・データ資産でアンパックされます。各パラメーターに対応するエンティティー値は、アンパックされたファイルで更新されます。ストア・アーカイブ内の DTD は更新されません。パラメーターの値は、パブリッシュがインスタンス化されるまでこのファイル (この場合 ForeignKeys.dtd) に保管されます。

最後に、サンプル・ストアのようにストアが複数の言語で発行される場合、発行パラメーターとそれに伴う説明は、ロケールに特定のファイルに見つかります。各発行パラメーターのフィールド・ラベルおよび説明は、 store-refs.xml のリソース・バンドル属性で定義されるプロパティー・ファイルにあります。発行中、発行は管理コンソールで使用される言語に特定のロケールを探します。 stores-ref.xml ファイルは、以下のファイルも定義します。

- resource-bundle="/SAR-INF/properties/publishNLS"

注: 発行パラメーターは、管理コンソールを介してのみ使用可能です。コマンド行でストア・アーカイブを発行すると、パラメーター値を指定できません。ストア・アーカイブにあるデフォルト値が使用されます。

デプロイ記述子は、発行プロセスの発行データ部分を制御するファイル (ibm-wc-load.xml) の位置を指定します。たとえば、 deploy-descriptor="WEB-INF/stores/FashionFlow/data/ibm-wc-load.xml のようになります。

パラメーターの選択後、**「終了 (Finish)」**をクリックして発行を開始します。管理コンソールやコマンド行を使用して発行のプロセスを開始した後は、操作は一切必要ありません。先の図やこの章でリストされているその他のステップは、すべて、WebSphere Commerce システムによって実行されます。

管理コンソールやコマンド行を使用してストア・アーカイブを発行する方法についての詳しい説明は、 WebSphere Commerce オンライン・ヘルプ の『ストア・アーカイブの発行』を参照してください。

ストア・アーカイブから資産をアンパックする

管理コンソールの発行ウィザードで**「終了 (Finish)」**をクリックした後、またはコマンド行から発行を実行した後、WebSphere Commerce はストア・アーカイブから WebSphere Commerce Server に資産をアンパックします。資産のアンパックは、ストア・アーカイブの SAR-INF ディレクトリーにある unpack.xml ファイルによって制御されます。

以下の unpack.xml file は、ConsumerDirect.sar ファイルからの例です。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ibm-wc-unpack SYSTEM "unpack.dtd">
<ibm-wc-unpack>
 <unpack>
  <include file="*"/>
  <exclude file="SAR-INF/*"/>
  <exclude file="*.zip"/>
  <exclude file="*.war"/>
  <exclude file="FashionFlow/devtools/flow/ui/*.properties"/>
  <exclude file="xml/*"/>
  <rename-store-dir target-name="FashionFlow">
   <store-dir path="WEB-INF/stores/FashionFlow" />
   <store-dir path="FashionFlow" />
   <store-dir path="WEB-INF/classes/FashionFlow" />
   <store-dir path="WEB-INF/xml/tools/stores/FashionFlow" />
  </rename-store-dir>
 </unpack>
 <unpack dest="${wc:ToolsStoresPropertiesPath}">
  <include file="FashionFlow/devtools/flow/ui/*.properties"/>
  <rename-store-dir target-name="FashionFlow">
   <store-dir path="FashionFlow" />
  </rename-store-dir>
 </unpack>
 <unpack dest="${wc:instanceDir}">
  <include file="xml/member/MemberRegistrationAttributes.xml"/>
 </unpack>
</ibm-wc-unpack>
```

unpack.xml ファイルは、アンパックするファイル (include および exclude エレメントを使用)、ファイルのアンパック先になるディレクトリー (unpack dest エンティティーを使用) を決定し、さらにディレクトリーの名前を変更します (rename-store-dir エンティティーを使用)。

デフォルトでは、アンパックはそのストア・アーカイブのすべてのファイルをアンパックします。しかし、指定すれば、ストア・アーカイブ内の特定のファイルだけをアンパックすることもできます。またデフォルトでは、インスタンス XML の DevTools エレメントからの StoreDocRoot および StoreWebPath パスを組み合わせることによって取得されるパスにファイルをアンパックします。このパスは、Stores Web モジュールのドキュメント・ルートを指します。しかし、指定すれば、上記の例 <unpack dest="${wc:ToolsStoresPropertiesPath}"> のように、別の場所のファイルをアンパックします。アンパックが変数を受け入れることに注目してください。この "${wc:ToolsStoresPropertiesPath}" の場合、ToolsStoresPropertiesPath 変数は *instance*.xml の devtools エレメントの属性です。

発行パラメーターの更新

ストア・アーカイブから資産をアンパックした後、WebSphere Commerce は発行ウィザードで作成または選択された発行パラメーター値を使って DTD ファイル (サンプル・ストアでは ForeignKeys.dtd) を更新します。たとえば、オリジナルのファイルに <!ENTITY STORE_IDENTIFIER "FashionFlow"> が入っていた場合、更新されたファイルには <!ENTITY STORE_IDENTIFIER "MyFashion"> が入っています。

注: 発行パラメーターは、管理コンソールを介してのみ使用可能です。コマンド行でストア・アーカイブを発行すると、発行パラメーターを選択できません。

発行データ

ファイルがアンパックされ、発行パラメーターが更新された後、発行プロセスにスケジュールされたジョブが作成されます。ストア・アーカイブ発行用のスケジュールされたジョブ番号は、管理コンソールの発行ユーティリティーに表示されます。

スケジューラーが発行のジョブを実行すると、通常 WebSphere Commerce は以下のアクションを完了させます。

- ストア・アーカイブ内の XML からデータベースにストア・データをロードする
- ストア・データを調整する
- レジストリー・コンポーネントを更新する
- コマンドを呼び出して、ビジネス・アカウントと契約を発行する
- 支払いの構成
- parameters.jsp ファイルを作成する

発行ジョブは、各ストア・アーカイブにある ibm-wc-load.xml ファイルに制御されます。このファイルは、stores-refs.xml ファイル中のデプロイ記述子属性によって指定されます。

ibm-wc-load.xml

ibm-wc-load.xml ファイルは、発行ジョブで完了されるタスク、およびこれらのタスクのシーケンスを決定します。以下に、ibm-wc-load.xml ファイルの例を挙げます。

```
<data-deploy base-dir="." default-target="all">
    <asset id="master" location="store-data-assets.xml"/>
    <asset id="resolved.master" location="store-data-assets.resolved.xml"/>
    <asset id="foreignKeys" location="ForeignKeys.dtd" type="dtd"/>
    <asset id="pmconfigfile" location="paymentinfo.xml"/>
    <deploy-task-cmd name="configPM" class="com.ibm.commerce.tools.devtools.
publish.tasks.payment.ConfigurePaymentTaskCmd"/>
    <target id="all">
     <task name="idresolve">
              param name="infile" value="${asset:master}" />
              param name="outfile" value="${asset:resolved.master}" />
     </task>
     <task name="massload">
              param name="infile" value="${asset:resolved.master}" />
     param name="maxerror" value="1" />
              param name="noprimary" value="error" />
    </task>
    <task name="configPM">
              param name="paymentConfigFilename" value="${asset:pmconfigfile}" />
              param name="storeIdentifier" value="
${asset:foreignKeys#STORE_IDENTIFIER}" />
              param name="organizationDN" value="
${asset:foreignKeys#ORGANIZATION_DN}" />
        </task>
   </target>
</data-deploy>
```

ここで

- base-dir は、発行される情報のディレクトリーです。 "." は、情報が ibm-wc-load.xml ファイルと同じディレクトリーにあることを示します。
- default-target は、実行されるターゲットの ID です。発行中に実行されるターゲットは 1 つだけです。

- asset id は、発行される資産に割り当てられる ID です。資産は、発行プロセス中に複数のタスクが動作すると ID を割り当てられます。
- location は、基本ディレクトリー (base-dir) に関連した、 ID を割り当てられる資産の位置です。
- deploy-task-cmd name は発行プロセス内で使用されるタスク・コマンドに割り当てられる短縮名です。
- deploy-task-cmd class は発行プロセス内で使用されるタスク・コマンドの完全な名前です。
- target id は、グループとして実行されるタスクのシーケンスに指定される名前です。複数のターゲットを定義することができますが、発行中に実行されるのは、デフォルト・ターゲットが参照する 1 つのターゲットだけです。
- task name は完了するタスクの名前です。このサンプルでは、タスクの短縮名が使用されることに注意してください。 ibm-wc-load.xml ファイルに新しいタスクを追加することができますが、どの新規タスクも次のコマンドを拡張する必要があります。 com.ibm.commerce.tools.devtools.publish.tasks.DeployTaskCmd.
- param name はタスクのパラメーター名です。入力パラメーターは、名前値のペア・ストリングとしてタスクに渡されます。
- value パラメーターの値です。値が変数でも構わないことに注目してください。これらの変数は、タスクの実行時に解決されます。

ストア・アーカイブ内の XML ファイルからデータベースにストア・データをロードする

ストア・アーカイブからデータベースに XML ファイルのストア・データをロードする際、 WebSphere Commerce は以下を行います。

ID リゾルバーを呼び出して ID を解決する: ローダー・パッケージ・ユーティリティーの 1 つ、ID リゾルバーは、ストア・アーカイブ XML ファイル内の XML エレメントに ID を生成します。たとえば、ID リゾルバーはサンプル・ストア XML ファイルで使用されている @ 別名を固有の値に置き換えます。サンプル・ストアで使用されている内部別名解決については、 489 ページの『付録 B. データの作成』を参照してください。

注: ID リゾルバーは、すでに発行されているストアを再発行するときの ID 解決にも使用できます。たとえば、ストア・アーカイブを一度発行した後で、そのストア・アーカイブの全体または一部を再発行する必要が生じた場合、 ID リゾルバーは、データベースから固有の ID を取り出し、それらの ID を再発行プロセスで使用します。

ID リゾルバーとその他のローダー・パッケージ・コンポーネントについては、379 ページの『第 37 章 ストア・データのロードの概要』を参照してください。

発行ユーティリティーで ID リゾルバーを呼び出すときは、どの ID リゾルバー方式を使用するかを指定する必要があります。 ID リゾルバーには、ID リゾルバーへの入力を処理する方式がいくつかあります。具体的にあげると、まず、オリジナルのデータに ID がある場合にデータを扱う方式 (更新方式) と、オリジナルのデータに ID がない場合の方式 (ロード方式) があります。また、一部の ID があって、一部の ID がない場合には、混合方式が使用されます。発行ユーティリティーが使用する方式は、 WebSphere Commerce 構成ファイル *instance_name*.xml で指定でき

ます。発行は、デフォルトで混合方式を使用します。 ID リゾルバーの方式の詳細については、 383 ページの『ID Resolve コマンド』を参照してください。

発行では、ID リゾルバーで使用するカスタマイザー・ファイルも指定する必要があります。デフォルトのカスタマイザー・ファイルは、 DBConnectionCustomizer またはOracleConnectionCustomizer です。

Oracle

OracleConnectionCustomizer カスタマイザー・ファイルは、以下のディレクトリーにあります。

- *WAS_installdir*/installedApps/*cell_name*/WC_*instance_name*.ear/properties

DB2

DBConnectionCustomizer ファイルは、以下の ZIP ファイルにあります。

- *WAS_installdir*/installedApps/*cell_name*/WC_*instance_name*.ear/properties
- 400 WAS_*userdir*/WAS_*instance_name*/installedApps/*cell_name*/WC_*instance_name*.ear/lib/loader/IdResGen.zip

注: 独自のカスタマイザー・ファイルを指定する場合は、 *instance_name*.xml ファイルの DevTools セクションにある以下の属性の値を変更する必要があります。

- IDResolverCustomizerFile="*myIDResolverCustomizerFile*"

store-data-asset.xml: 各サンプル・ストア・アーカイブには、store-data-asset.xml ファイルが含まれていなければなりません。 store-data-asset.xml ファイルは、発行中に含められるストア・アーカイブに、すべてのデータ資産ファイルのプレースホルダーを入れます。

以下は、ConsumerDirect.sar の store-data-asset.xml ファイルの一部で、プレースホルダーを示します。

```
<?xml version="1.0"?>
<!DOCTYPE import SYSTEM "store-data-assets.dtd">

<import>
 &modelorg.xml;
 &modelorgrole.xml;
 &storeorg.xml;
 &storeorgrole.xml;
 &fulfillment.xml;
 &store.xml;
 &en_US_store.xml;
 &en_US_fulfillment.xml;
 &catalog.xml;
 &en_US_catalog.xml;
 &tax.xml;
```

発行中、store-data-asset.xml ファイル中のプレースホルダーを使って識別されるすべてのデータ資産は、store-data-asset.xml ファイルに統合され、 1 つの大きなデータ・ファイルが作成されます。

ID リゾルバーは、store-data-asset.xml とそれに対応する DTD ファイル store-data-asset.dtd を使用して ID を解決します。 ID が解決されると、ID リゾルバーは、固有の ID が含まれた store-data-asset.resolved.xml ファイルを作成します。なお、ID 解決プロセスの途中でエラーが発生した場合、ローダー・パッケージは、 messages.txt ファイルにエントリーを追加します。詳細については、375 ページの『ログ・ファイルの発行』を参照してください。

ローダー・パッケージを呼び出して、解決されたマスター XML ファイルをデータベースにロードする: ローダー・パッケージは、解決された store-data-asset.resolved.xml をデータベースにロードします。なお、ロード・プロセスの途中でエラーが発生した場合、ローダー・パッケージは、 messages.txt ファイルにエントリーを追加します。

ローダー・パッケージの詳細については、 379 ページの『第 37 章 ストア・データのロードの概要』を参照してください。

管理コンソールやコマンド行の発行ユーティリティーでローダー・パッケージを呼び出すときは、どのローダー方式を使用するかを指定する必要があります。管理コンソールでは、以下のローダー方式を使用できます。

- SQL インポート
- インポート
- ロード

注: 管理コンソールは、デフォルトで SQL インポート方式を使用します。管理コンソールやコマンド行の発行ユーティリティーが使用する方式は、WebSphere Commerce 構成ファイル *instance_name*.xml の DevTools 要素の LoaderMode 属性で指定します。

- SQL インポート: この方式では、 Java Database Connectivity (JDBC) を使用してデータの挿入と更新を行います。これは、操作の点で最も柔軟性の高い方式ですが、大量のデータを少ないテーブルにインポートするときには、最も処理に時間がかかる方式でもあります。この方式では、列レベルの更新が可能です。 SQL インポートの使用をお勧めします。

  注: SQL インポート方式は、最も安全な方式です。無効なデータがあった場合でも、データベースが破壊されることはありません。 SQL インポート方式でロードするには、レコードがデータベース・スキーマの制約を満たしている必要があります。その他のローダー方式は、処理速度に重点を置いているので、それほど検査を行わずにデータベースに大量のロードを実行します。したがって、その他の方式を使用する場合は、データが正しいことを確認しておく必要があります。
- インポート: これは、 DB2 にもともと組み込まれているインポート機能を使用する方式であり、平均的な処理速度と柔軟性でセル・レベルの更新を実行できます。この方式は、Oracle では使用できません。

- ロード: これは、RDBMS にもともと組み込まれている機能 (DB2 Load または SQLLoad) を使用する方式であり、大量のデータを少ないテーブルにロードするときには、最も処理が速い方式です。

ロード・コマンドの方式については、 393 ページの『Load コマンド』を参照してください。

管理コンソールやコマンド行の発行ユーティリティーでは、ローダーで使用するカスタマイザー・ファイルも指定する必要があります。 WebSphere Commerce 構成ファイル *instance_name*.xml でカスタマイザー・ファイルを指定しない場合、発行コードは、デフォルトのカスタマイザー・ファイル MassLoadCustomizer を使用します。

注: 独自のカスタマイザー・ファイルを指定する場合は、 *instance_name*.xml ファイルの DevTools セクションにある以下の属性の値を変更する必要があります。

- LoaderCustomizerFile="*myLoaderCustomizerFile*"

  デフォルトでは、 WebSphere Commerce 構成ファイル *instance_name*.xml には、この属性の値が指定されていません。

ストア言語の調整

サンプル・ストア・アーカイブには、WebSphere Commerce がサポートするすべての言語のデータが入っています。そのため、ローダー・パッケージがストア・データをロードすると、すべての言語情報がストアにロードされます。しかし、ストアがサポートできるのは、ストアがあるインスタンスがサポートする言語のみです。調整ストア言語タスクを使用すると、インスタンスでサポートされる言語のみが確実にストアで使用できます。インスタンスに、追加の言語のサポートを追加したい場合、構成マネージャーを使用します。詳細については、WebSphere Commerce オンライン・ヘルプ、「構成マネージャー」を参照してください。

レジストリー・コンポーネントを更新する

発行プロセスでは、レジストリー・コンポーネントの更新も行います。 com.ibm.commerce.scheduler.commands.RefreshRegistryCmd を呼び出すことによって、 WebSphere Commerce のすべてのレジストリーの更新を発行してください。詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

コマンドを呼び出して、ビジネス・アカウントと契約を発行する

一部のストア・データ・ベース資産 (契約、ビジネス・アカウント) は、ローダー・パッケージではロードできません。そのため、発行プロセスでは、対応するコマンドをさらに呼び出して、これらの資産を WebSphere Commerce Server に発行します。それらのコマンドは次のとおりです。

- AccountImport — ストア・アーカイブ内の businessaccount.xml ファイルからビジネス・アカウントを作成します。
- ContractImportApprovedVersion — ストア・アーカイブ内の contract.xml ファイルから契約をインポートします。
- ProductSetPublish — ビジネス・アカウントと契約を作成する前に、データベース内の商品セット・データとカタログを同期化します。管理コンソールやコマンド

行発行ユーティリティーによって ProductSetPublish コマンドを呼び出すと、そのコマンドから、次に AccountImport コマンドと ContractImportApprovedVersion コマンドが呼び出されます。

ビジネス・アカウントと契約を発行するための詳細については、 443 ページの『第 39 章 ビジネス・アカウントと契約の発行』を参照してください。

支払いの構成

発行プロセスには、支払いの構成ステップも含まれます。 WebSphere Commerce は、WebSphere Commerce Payments をサポートします。支払い処理の方式として WebSphere Commerce Payments を使用する計画がある場合は、 237 ページの『第 21 章 決済手段』で説明されている方法に従って支払い XML ファイルを作成する必要があります。発行されるストア・アーカイブに支払い XML ファイルが組み込まれている場合、 WebSphere Commerce は、発行のプロセスで、以下の支払いの構成を完了させます。

- マーチャントの作成
- アカウントの作成 (オフライン・カセット専用)
- `paymentinfo.xml` で指定されているブランドの作成 (オフライン・カセット専用)
- ユーザー権限の割り当て

エラー処理: 発行プロセスの支払い構成のフェーズでエラーが発生した場合は、発行ログ (『ログ・ファイルの発行』を参照) でエラー・メッセージを確認できます。

parameters.jsp ファイルを作成する

発行プロセスでは、`parameters.jsp` ファイルが作成されます。このファイルには、storeId パラメーターが含まれます . サンプル・ストアの index.jsp ファイルは、このパラメーターを使ってストアを立ち上げます。

`parameters.jsp` ファイルは、以下のディレクトリーにあります。

- *WAS_installdir*/installedApps/*cell_name*/WC_*instance_name*.ear/Stores.war/*storedir*/include
- 400 *WAS_userdir*/*WAS_instance_name*/installedApps/*cell_name*/WC_*instance_name*.ear/Stores.war/*storedir*/include

エラー処理

発行プロセスの資産発行のフェーズでエラーが発生した場合は、発行ログ (『ログ・ファイルの発行』を参照) か、管理コンソールの「発行の要約」ページで、エラー・メッセージを確認できます。

ログ・ファイルの発行

発行プロセスの資産発行のフェーズで発生したエラーはすべて、以下のログおよびトレース・ファイルに書き込まれます。

- `activity.log`: WebSphere Application Server ログ。 WebSphere Commerce のすべてのエラー・メッセージは、activity.log に書き込まれます。発行が失敗した場合は、まず最初にこのログを調べてください。 `activity.log` は以下に示すディレクトリーにあります。
  - *WAS_installdir*/logs

– 400 *WAS_userdir*/logs

– Developer *WCDE_installdir*/*workspace_name*/.metadata/plugins/com.ibm.etools.server.core/tmp0/logs

- `SystemOut.log and SystemErr.log`: SystemOut.log は、以下に示すディレクトリーにあります。ストア発行中の標準出力および標準エラーに書き込まれる情報が入っています。`SystemOut.log` および `SystemErr.log` は以下に示すディレクトリーにあります。
  – *WAS_installdir*/logs/*instance_name*
  – 400 *WAS_userdir*/*WAS_instance_name*/logs/*WC_instance_name*

  注: `SystemOut.log` および `SystemErr.log` は、開発環境では使用できません。標準出力または標準エラーに書き込まれる情報は、WebSphere Studio コンソールに表示され、次の位置に取り込まれます。
    – workspaceDir¥.metadata¥.plugins¥com.ibm.etools.server.core¥tmp0¥logs¥server1¥trace.log
- `messages.txt`: 発行プロセスのローダー・パッケージ部分で生成されたエラー・メッセージが含まれています。発行が失敗した場合は、まず最初にこのログを調べてください。これらのエラー・メッセージに示される行番号や列番号は、`store-data-asset.resolved.xml` ファイルを参照するものです。`messages.txt` は、以下に示すディレクトリーにあります。
  – WC_*installdir*/instances/*instance_name*/logs
  – 400 WC_*userdir*/instances/*instance_name*/logs
  – Developer WCDE_*installdir*/Commerce/logs

  これらのエラー・メッセージに示される行番号や列番号も、`store-data-asset.xml` ファイルを参照することがあります。
- `trace.txt`: 発行プロセスのローダー・パッケージ部分に関するトレース情報が含まれています。このファイルには、発行プロセスの ID リゾルバー部分についてのメッセージも含まれます。 trace.txt は、デフォルトではオフになっています。`trace.txt` は、以下のディレクトリーにあります。
  – WC_*installdir*/instances/*instance_name*/logs
  – 400 WC_*userdir*/instances/*instance_name*/logs

    注: デフォルトでは、トレースはオンになっており、循環ログ・ファイルとして生成されます。
  – Developer WCDE_*installdir*/Commerce/instances/*instance_name*/logs
- `trace.log`: WebSphere Application Server トレース・ログの一部。WC_DEVTOOLS トレース・コンポーネントが使用可能な場合、発行プロセスのトレース情報が含まれます。ロギングの使用可能化に関する詳細は、 WebSphere Commerce 管理ガイドの構成に関する章を参照してください。 `trace.log` は以下のディレクトリーにあります。
  – *WAS_installdir*/logs/*WC_instance_name*
  – 400 *WAS_userdir*logs/*WC_instance_name*

– Developer *workspaceDir*¥.metadata¥.plugins¥com.ibm.etools.server.core¥tmp0¥logs¥server1

- 400 RESWCSID.txt: 発行プロセスの ID リゾルバー部分からのメッセージが含まれます。これらのエラー・メッセージに示される行番号や列番号は、入力ファイル、たとえば store-data-assets.xml ファイルを参照するものです。RESWCSID.txt は、以下のディレクトリーにあります。
  – WC_*userdir*/instances/*instance_name*/logs

trace.txt ファイルおよび messages.txt ログ・ファイルの構成 (つまり、ログ・レベルその他のオプションの調整) を行うには、以下のファイルを編集してください。

- WC_*installdir*/instances/*instance_name*/xml/loader/WCALoggerConfig.xml
- 400 WC_*userdir*/instances/*instance_name*/xml/loader/WCALoggerConfig.xml
- Developer WCDE_*installdir*/Commerce/instances/*instance_name*/xml/loader/WCALoggerConfig.xml

管理コンソールに対してストア・アーカイブを使用可能にする

管理コンソールからストア・アーカイブを発行するには、次のうちいずれかの方法で、ストア・アーカイブを管理コンソールに対して使用可能にしなければなりません。

- SARRegistry.xml ファイルでストア・アーカイブを登録する
- ストア・アーカイブを適当なストア・アーカイブ・ストア・アーカイブにコピーする

SARRegistry.xml ファイルでストア・アーカイブを登録する

管理コンソールからストア・アーカイブを発行し、ストアをプレビューするには、SARRegistry.xml ファイルでストア・アーカイブを登録する必要があります。SARRegistry.xml ファイルは、以下のディレクトリーにあります。

- *WC_installdir*/xml/tools/devtools
- 400 *WC_userdir*/xml/tools/devtools

ストア・アーカイブを登録するには、ストア・アーカイブ・ファイルのパスを、任意のプレビュー・ファイルへのパスと共に SARRegistry.xml ファイルに含めます。次の例は、SARRegistry.xml ファイルで消費者向けサンプル・ストア・アーカイブを登録するエントリーを例示します。

```
<!-- Consumer Direct -->
 <SampleSAR fileName="ConsumerDirect.sar" relativePath="ConsumerDirect">
  <view name="ConsumerDirect" />
  <view name="default" />
  <html locale="de_DE" featureFile="" sampleSite="de_DE/B2C/FashionFlow/
index.html"/>
  <html locale="en_US" featureFile="" sampleSite="en_US/B2C/FashionFlow/
index.html"/>
  <html locale="es_ES" featureFile="" sampleSite="es_ES/B2C/FashionFlow/
index.html"/>
  <html locale="fr_FR" featureFile="" sampleSite="fr_FR/B2C/FashionFlow/
```

```
index.html"/>
  <html locale="it_IT" featureFile="" sampleSite="it_IT/B2C/FashionFlow/
index.html"/>
  <html locale="ja_JP" featureFile="" sampleSite="ja_JP/B2C/FashionFlow
index.html"/>
  <html locale="ko_KR" featureFile="" sampleSite="ko_KR/B2C/FashionFlow/
index.html"/>
  <html locale="pt_BR" featureFile="" sampleSite="pt_BR/B2C/FashionFlow/
index.html"/>
  <html locale="zh_CN" featureFile="" sampleSite="zh_CN/B2C/FashionFlow/
index.html"/>
  <html locale="zh_TW" featureFile="" sampleSite="zh_TW/B2C/FashionFlow/
index.html"/>
 </SampleSAR>
```

ここで

- `fileName` はストア・アーカイブの名前です。
- `relativePath` は、WebSphere Commerce 構成ファイル *instance_name.xml* の DevTools エレメントの sampleSarPath 属性に関連するディレクトリー・パスです。
- `view name` は、発行ユーザー・インターフェースから、ストア・アーカイブが表示されるビュー名です。ストア・アーカイブが、複数のビューに表示されることに注目してください。たとえば、消費者向けのビューとデフォルトのビューの両方に表示されます。
- `html locale` は、HTML プレビュー・ファイルのロケールです。
- `featureFile` フィーチャー・ファイルは推奨されませんし、今後使用されません。

ストア・アーカイブを適当なストア・アーカイブ・ストア・アーカイブにコピーする

管理コンソールからストア・アーカイブを発行したいが、プレビュー・ページを含める予定がない、またはストアが管理コンソールを表示する発行ビューを指定しない場合、ストア・アーカイブを次のディレクトリーにコピーするだけで構いません。

- *WC_installdir*/instances/*instance_name*/sar

ストア・アーカイブがデフォルトのビューに表示されます。

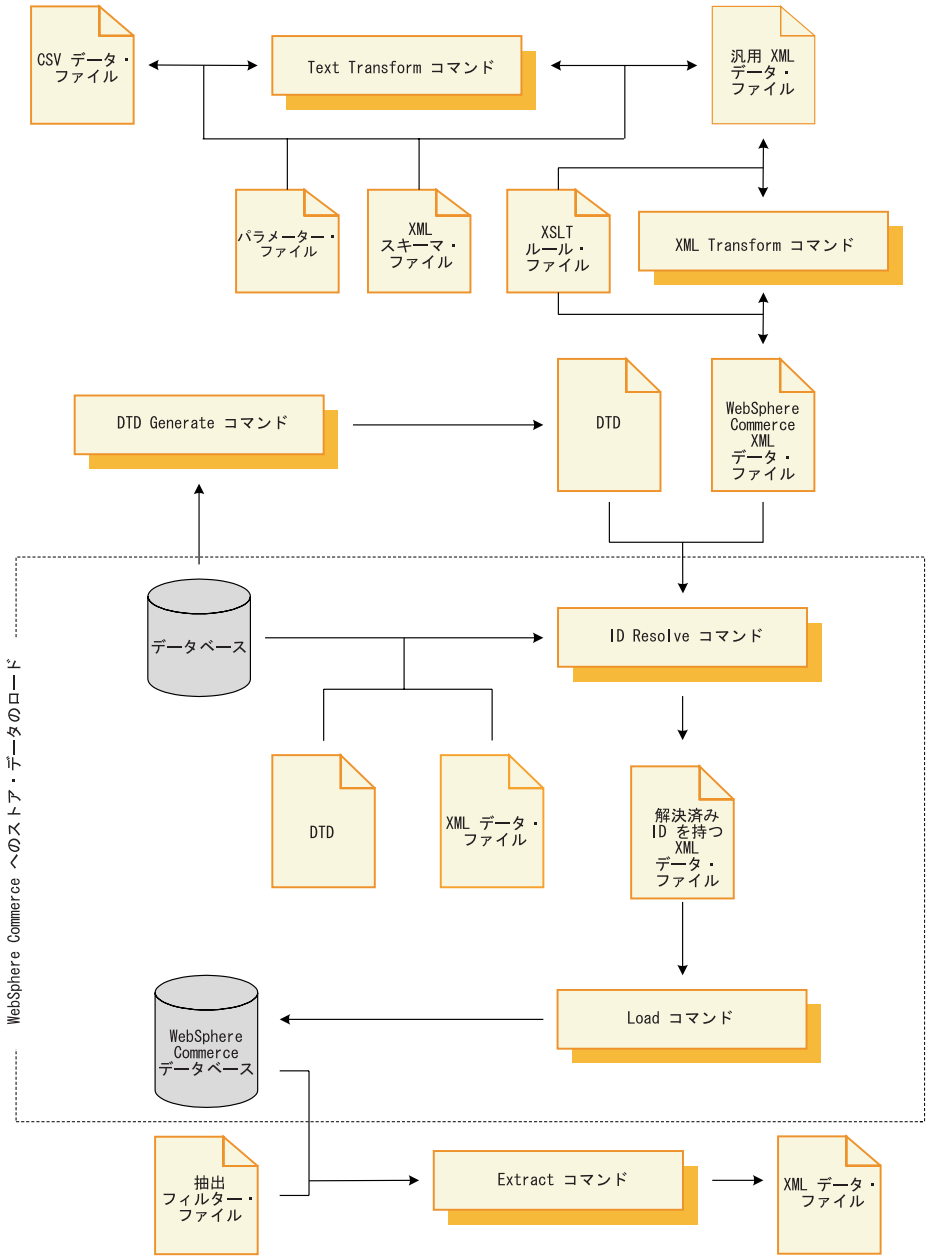
第 37 章 ストア・データのロードの概要

ストア・データを作成したなら、それをストア・アーカイブとしてパッケージし、WebSphere Commerce 管理コンソールを使用して発行できます。あるいは、WebSphere Commerce ローダー・パッケージを使用して、 WebSphere Commerce Server データベースに直接ロードすることもできます。 WebSphere Commerce のデータベース資産グループのロード・プロセスについては、 429 ページの『第 38 章 WebSphere Commerce データベース資産グループのロード』と 437 ページの『データベース資産グループのロード』を参照してください。

ローダー・パッケージには、ロードするデータを準備し、実際にストアにロードするための、6 つのコマンド行ユーティリティーと 2 つの関連する管理用ツールがあります。これらのコマンドやツールでは、情報の管理に Extensible Markup Language (XML) データ・ファイルを使用しています。

WebSphere Commerce でのデータ・ロードについて

次の図は、ローダー・パッケージ・コマンドを使用して実行できる、データの準備、ロード、および抽出のプロセスを示しています。



点線で囲まれた部分に、2 つのプロセスが示されていることに注目してください。これが、ストア・データを WebSphere Commerce Server データベースにロードする際に最も一般的に使用されるプロセス、つまり、ID の解決とデータのロードです。この章では、これらのプロセスに焦点を合わせます。

WebSphere Commerce Server データベースにロードするデータの準備については、109 ページの『第 5 部 ストア・データの概要』を参照してください。

WebSphere Commerce Server データベースにデータをロードする際には、次の 2 つのローダー・パッケージ・コマンド行ユーティリティーが一般的に使用されます。

- **ID Resolve コマンド**

  ローダー・パッケージを使用して、XML データを WebSphere Commerce Server データベースにロードするには、 XML エレメントを、ロード先の WebSphere Commerce Server データベースのスキーマに直接対応付ける必要があります。データベース・スキーマの固有キーや基本キーに対応する属性を持つすべての XML エレメントには固有の ID が必要であり、データベース・スキーマのヌルにできないすべての列には、ヌル以外の値で定義されている対応属性が必要です。 ID リゾルバーは、 XML エレメントを修飾する固有キー属性や基本キー属性に対応する固有 ID を生成する機能を持っています。

  注: 本書で言う ID とは、データベース・テーブルの 1 つの列の値であり、その値によって、各行を一意に識別することになります。 ID リゾルバーを使用して ID を生成する場合は、 KEYS テーブルまたは SUBKEYS テーブルから基本値を取得して、その値に連番を付ける形で、データベース・テーブル内の各行の ID を解決することになります。

  このコマンドの詳細については、 383 ページの『ID Resolve コマンド』、 416 ページの『ローダー・パッケージ・コマンドおよびスクリプトの使用』、 416 ページの『ID 解決の例』を参照してください。

- **Load コマンド**

  ローダーは、データベースにデータをロードするための入力として、妥当かつ整形式の XML ファイルを使用します。 XML 文書のエレメントはデータベースのテーブル名にマップされ、エレメント属性は列にマップされます。

  注: 妥当な文書と整形式の文書の要件については、 World Wide Web Consortium (W3C) の XML 仕様を参照してください。

  このコマンドの詳細については、393 ページの『Load コマンド』、416 ページの『ローダー・パッケージ・コマンドおよびスクリプトの使用』、424 ページの『データ・ロードの例』を参照してください。

この章では、主にこれらのコマンドに焦点を合わせます。

データの管理には、以下のローダー・パッケージ・コマンド行ユーティリティーも使用できます。

- **DTD Generate コマンド**

  DTD ジェネレーターは、 XML データをロードするターゲット・データベースのテーブルと列を説明する、文書型定義 (DTD) を生成します。加えて、DTD ジェネレーターでは、データベースの XML スキーマも生成できます。

  DTD ジェネレーターは、 WebSphere Commerce データベース・スキーマに基づいて DTD を作成することができます。サンプル・ストア・アーカイブに含まれている DTD を使用し、データベース・スキーマを変更しない場合、通常、DTD ジェネレーターを使用して DTD を生成する必要はありません。

  詳細については、405 ページの『DTD Generate コマンド』を参照してください。

- **Extract コマンド**

抽出プログラムは、データベースに対するクエリーを用い、選択されたデータのサブセットをデータベースから XML 文書に抽出します。

このコマンドを使用して、データベースのデータを XML フォーマットに抽出できます。

詳細については、408 ページの『Extract コマンド』を参照してください。

- **Text Transform コマンド**

  テキスト変換ツールは、文字区切り変数フォーマットと XML データ・フォーマットの間でのデータ変換を行います。

  データベースのデータを直接 XML フォーマットに抽出できない場合は、まずデータを文字区切り変数フォーマットで保存してから、このコマンドでそのデータを XML フォーマットに変換できます。

  詳細については、411 ページの『Text Transform コマンド』を参照してください。

- **XML Transform コマンド**

  XML 変換プログラムは、XML 文書内のデータを代替の XML フォーマットに変換します。変換のためのマッピング・ルールの定義には、Extensible Stylesheet Language (XSL) が使用されます。

  このコマンドを使用して、 XML データを、ロード先の WebSphere Commerce データベースのスキーマに直接対応したフォーマットに変換できます。

  詳細については、413 ページの『XML Transform コマンド』を参照してください。

この章は、これらのコマンドを主要な論点としていません。これらのコマンドの詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

WebSphere Commerce ローダー・パッケージには、データ管理機能の管理を支援するための以下のツールも組み込まれています。

- **テキスト変換ツール**

  テキスト変換ツールは、 Text Transform コマンドを使用して、文字区切り変数フォーマットと XML データ・フォーマットの間で、データ変換の処理を行うのに役立ちます。

- **XSL エディター**

  XSL エディターは、XML 変換プログラムで使用できる、 XSL ファイルを編集するための視覚的なインターフェースを提供します。 XML フォーマットへのデータ変換を行うためのマッピング規則を定義するときは、XSL エディターを使用して、ソース DTD のエレメントからターゲット DTD のエレメントへのアソシエーションを確立します。

この章は、これらのツールを主要な論点としていません。これらのツールの詳細については、最新版の WebSphere Commerce オンライン・ヘルプを参照してください。

ストア・データをロードするためのローダー・パッケージ・コマンド

ID Resolve コマンド

このコマンドは、XML データ・エレメントをデータベースにロードするために必要な、 XML データ・エレメントの固有 ID を生成します。必要な固有 ID がソースの XML データにすでに含まれている場合は、 ID リゾルバーを実行する必要はありません。

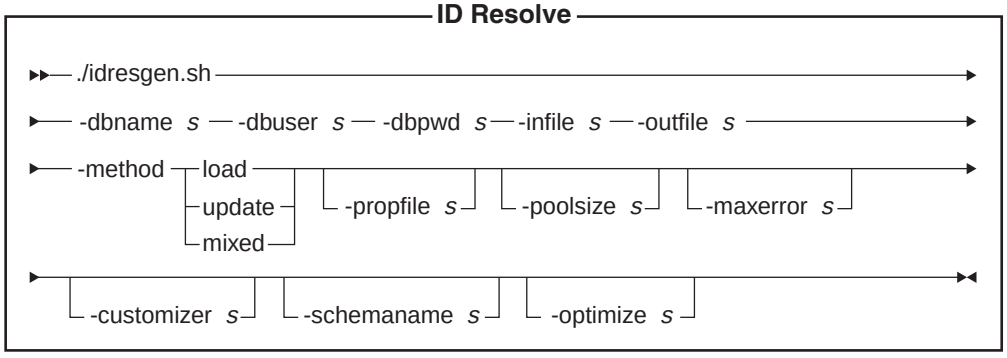
WebSphere Commerce データベース・スキーマは、テーブル間のさまざまな関係を表すのに用いられる、テーブル内の基本キーや外部キーを定義します。したがって、 WebSphere Commerce の XML エレメントには、固有 ID の付いた対応属性を指定する必要があります。 WebSphere Commerce Server データベースでは、 ID を解決する必要のあるテーブルが KEYS および SUBKEYS テーブルで定義されます。これらのテーブルは、Websphere Commerce では「*1 次テーブル*」と呼ばれています。 KEYS テーブルと SUBKEYS テーブルの詳細については、 WebSphere Commerce オンライン・ヘルプを参照してください。

注: KEYS テーブルや SUBKEYS テーブルに定義されていないテーブルの ID を解決することが必要になった場合には、 ID リゾルバーを実行する前に、そのテーブルを SUBKEYS テーブルに追加してください。

WebSphere Commerce の XML エレメントと属性は、データベース間やデータベース・インスタンス間で移植できるようになっているため、 ID は基本的に内部別名で表すことになっています。データを WebSphere Commerce Server データベースにロードするには、まずこれらの別名を有効な数値 ID に解決する必要があります。詳細については、489 ページの『付録 B. データの作成』を参照してください。

注:

1. 上の図は、主にコマンド・パラメーターを参照するための図です。
2. このコマンドのパラメーターとして指定するファイル名の先頭には、相対パスや絶対パスを付けてもかまいません。



注: このコマンドのパラメーターとして指定するファイル名の先頭には、相対パスや絶対パスを付けてもかまいません。

パラメーター値:

-dbname

AIX Linux Solaris ターゲット・データベースの名前。

400 これは、リレーショナル・データベース・ディレクトリー (WRKRDBDIRE) で表示されるのと同じ名前です。

-dbuser

AIX Linux Solaris データベースに接続しているユーザーの名前。 400 これは通常、インスタンス・ユーザー名と同じです。

-dbpwd

データベースに接続しているユーザーのパスワード

-infile テーブル・レコードを含む XML 文書の名前

-outfile

生成される出力 XML ファイルの名前 (このファイルをローダーへの入力として使用できます)

-method

入力ファイルの処理で使用する方式。コマンドは入力ファイルを、データベースにレコードが存在していないかのように (ロード)、または入力オブジェクトの ID がすでに存在しているかのように (更新) 処理することができます。データベースに存在するレコードと存在しないレコードがある場合は、混合方式を使用します。デフォルトの方式はロードです。

-propfile

「名前 = 値」の形式で Java プロパティーを指定したテキスト・ファイル。このプロパティー・ファイルで、 ID リゾルバーによる ID の解決方法を設定します。このファイルでは、 1 次行の ID が必要なテーブルの検索に、どの 1 次エントリーの列を使用するかを記述します。また、外部キー ID 検索に使用する列名や、主テーブル (CATEGORY、PRODUCT など) のクエリーに使用する選択述部を定義します。このファイルの中で、 ID を含まない固有索引が定義されているテーブルについては、エントリーを省略できます。このパラメーターはオプションです。 IdResolveKeys.properties がデフォルトのファイルです。 次の例のいずれかに示されているようにして、このプロパティー・ファイルを指定できます。

AIX Linux Solaris

```
-propfile WC_installdir/my_directory/file_name.properties
-propfile WC_installdir/my_directory/file_name
```

400

```
-propfile WC_userdir/my_directory/file_name.properties
-propfile WC_userdir/my_directory/file_name
```

現行ディレクトリーにこのファイルがある場合は、次の例に示すようにして同じファイルを指定できます。

```
-propfile file_name.properties
```

クラスパスのシステム環境変数で指定されたディレクトリーにこのファイルが存在する場合、同じファイルを次の例のように指定できます。

-propfile *file_name*

ここで、*my_directory* はユーザー定義のディレクトリー、 *file_name* は使用するプロパティー・ファイルの名前です。

ID リゾルバーで使用する新しいプロパティー・ファイルを作成して指定するための詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

-poolsize

予約する ID の数。このパラメーターはオプションです。デフォルトの数は 50 です。

-maxerror

ID リゾルバーが終了した後のエラーの数。このパラメーターはオプションです。デフォルト値は 1 です。

-customizer

使用するカスタマイザー・プロパティー・ファイルの名前。このパラメーターはオプションです。カスタマイザー・プロパティー・ファイルでは、 ID リゾルバーの動作方法を設定します。デフォルト・ファイルは次のとおりです。

AIX Linux Solaris DB2ConnectionCustomizer.properties

400 ISeries_RESWCSID_Customizer.properties

インスタンスをツールボックス・ドライバーを使用するように構成した場合は、ツールボックス・ドライバーに提供されている Toolbox_RESWCSID_Customizer カスタマイザー・ファイルを使用してください。 -dbname パラメーターにホスト名を指定することも必要です。idresgen.sh スクリプトを呼び出す例を挙げます。

```
./idresgen.sh -dbname MY.HOSTNAME.CA -dbuser instance -dbpwd mypass
-infile /path/infile.xml -outfile /path/outfile.xml -method sqlimport
-customizer Toolbox_RESWCSID_Customizer
```

次の例のいずれかに示されているようにして、カスタマイザー・プロパティー・ファイルを指定できます。

AIX Linux Solaris

-customizer *WC_installdir/my_directory/file_name*.properties

-customizer *WC_installdir/my_directory/file_name*

400

-customizer *WC_userdir/my_directory/file_name*.properties
-customizer *WC_userdir/my_directory/file_name*

ここで、*my_directory* はユーザー定義のディレクトリー、 *file_name* は使用したいプロパティー・ファイルの名前です。

現行ディレクトリーにこのファイルがある場合は、次の例に示すようにして同じファイルを指定できます。

-customizer *file_name*.properties

クラスパスのシステム環境変数で指定されたディレクトリーにこのファイルが存在する場合、同じファイルを次の例のように指定できます。

```
-customizer file_name
```

新しいカスタマイザー・プロパティー・ファイルを作成して指定するための詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

-schemaname

ターゲット・データベース・スキーマの名前。このパラメーターはオプションです。

このパラメーターを指定しないでコマンドを実行すると、カスタマイザー・プロパティー・ファイル内で、 SchemaName の値を指定した「名前 = 値」のペアが検索されます。プロパティー・ファイルにそのペアが存在すれば、そこに指定されている値が使用されます。このパラメーターをコマンド行でも指定せず、プロパティー・ファイルにもその指定がなければ、デフォルトで、データベース内の KEYS テーブルのスキーマ名が使用されます。

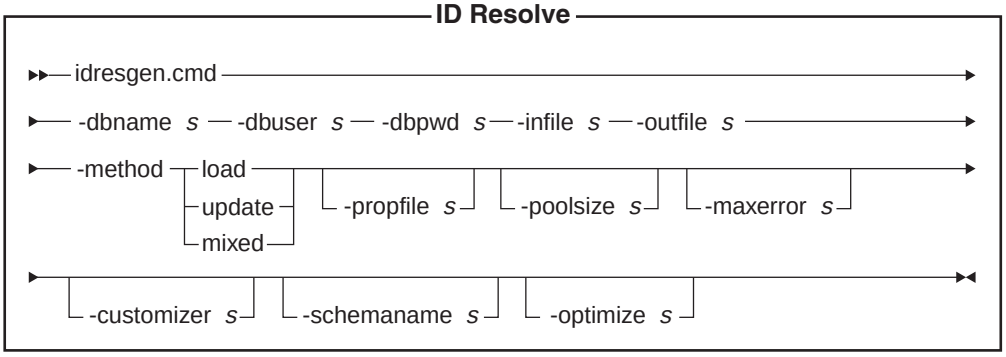
400 -schemaname パラメーターをコマンド行でも指定せず、プロパティー・ファイルにもその指定がなければ、デフォルトで、-dbuser パラメーターの値が使用されます。

-optimize

-optimize no

IdResolver は重複レコード検査をスキップしてから、解決済みレコードを出力ファイルに書き込みます。このオプションを使用すると、ユーザーは IdResolver の最適化機能をオフにすることができます。

Windows



パラメーター値:

-dbname
ターゲット・データベースの名前

-dbuser
データベースに接続しているユーザーの名前

-dbpwd
データベースに接続しているユーザーのパスワード

-infile テーブル・レコードを含む XML 文書の名前

-outfile
生成される出力 XML ファイルの名前 (このファイルをローダーへの入力として使用できます)

-method
入力ファイルの処理で使用する方式。コマンドは入力ファイルを、データベースにレコードが存在していないかのように (ロード)、または入力オブジェクトの ID がすでに存在しているかのように (更新) 処理することができます。データベースに存在するレコードと存在しないレコードがある場合は、混合方式を使用します。デフォルトの方式はロードです。

-propfile
「名前 = 値」の形式で Java プロパティーを指定したテキスト・ファイル。このプロパティー・ファイルで、 ID リゾルバーによる ID の解決方法を設定します。このファイルでは、 1 次行の ID が必要なテーブルの検索に、どの 1 次エントリーの列を使用するかを記述します。また、外部キー ID 検索に使用する列名や、主テーブル (CATEGORY、PRODUCT など) のクエリーに使用する選択述部を定義します。このファイルの中で、 ID を含まない固有索引が定義されているテーブルについては、エントリーを省略できます。このパラメーターはオプションです。 IdResolveKeys.properties がデフォルトのファイルです。 次の例のいずれかに示されているようにして、このプロパティー・ファイルを指定できます。

```
-propfile WC_installdir¥my_directory¥ file_name.properties
```

```
-propfile WC_installdir¥my_directory¥file_name
```
現行ディレクトリーにこのファイルがある場合は、次の例に示すようにして同じファイルを指定できます。

```
-propfile file_name.properties
```
クラスパスのシステム環境変数で指定されたディレクトリーにこのファイルが存在する場合、同じファイルを次の例のように指定できます。

```
-propfile file_name
```

ここで、*my_directory* はユーザー定義のディレクトリー、 *file_name* は使用したいプロパティー・ファイルの名前です。

ID リゾルバーで使用する新しいプロパティー・ファイルを作成して指定するための詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

-poolsize

予約する ID の数。このパラメーターはオプションです。デフォルトの数は 50 です。

-maxerror

ID リゾルバーが終了した後のエラーの数。このパラメーターはオプションです。デフォルト値は 1 です。

-customizer

使用するカスタマイザー・プロパティー・ファイルの名前。このパラメーターはオプションです。カスタマイザー・プロパティー・ファイルでは、 ID リゾルバーの動作方法を設定します。 DB2ConnectionCustomizer.properties がデフォルトのファイルです。次の例のいずれかに示されているようにして、カスタマイザー・プロパティー・ファイルを指定できます。

```
-customizer WC_installdir¥my_directory¥file_name.properties
-customizer WC_installdir¥my_directory¥file_name
```

現行ディレクトリーにこのファイルがある場合は、次の例に示すようにして同じファイルを指定できます。

```
-customizer file_name.properties
```

クラスパスのシステム環境変数で指定されたディレクトリーにこのファイルが存在する場合、同じファイルを次の例のように指定できます。

```
-customizer file_name
```

ここで、*my_directory* はユーザー定義のディレクトリー、 *file_name* は使用するプロパティー・ファイルの名前です。新しいカスタマイザー・プロパティー・ファイルを作成して指定するための詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

-schemaname

ターゲット・データベース・スキーマの名前。このパラメーターはオプションです。

このパラメーターを指定しないでコマンドを実行すると、カスタマイザー・プロパティー・ファイル内で、 SchemaName の値を指定した「名前 = 値」のペアが検索されます。プロパティー・ファイルにそのペアが存在すれば、そこに指定されている値が使用されます。このパラメーターをコマンド行でも指定せず、プロパティー・ファイルにもその指定がなければ、デフォルトで、データベース内の KEYS テーブルのスキーマ名が使用されます。

-optimize

-optimize no

IdResolver は重複レコード検査をスキップしてから、解決済みレコードを出力ファイルに書き込みます。このオプションを使用すると、ユーザーは IdResolver の最適化機能をオフにすることができます。

解決のテクニック:

ID リゾルバーは、プロパティー・ファイルが使用されているかどうかに応じ、以下の 2 つか 3 つのテクニックを組み合わせて ID を解決します。

- **内部別名解決**

内部別名 ID 解決法では、ソース XML 文書の固有キー (ID) の代わりに別名が使用されます。これで別名は、そのエレメントを参照するために、XML ファイル内の他の場所で使用できます。

内部別名は、XML ファイルを通じて一貫して使用される必要があります。たとえば、住所録 ID ADDRBOOK_ID の別名が @addrbook_1 だとすると、そのファイル中のその ID に対するすべての外部キー参照では、 @addrbook_1 を使用しなければなりません。

別名は、特定の XML ファイルにおける一時的なものであることに注意してください。別名は保存されません。また、同じ別名を再び設定しない限り、別名は別の XML ファイルでは使用できません。ただし、管理コンソールの発行においては、すべてのデータを通じて別名解決が実行されるように XML ファイルが連結されます。

- **固有索引解決**

  ID リゾルバーは、データベース・スキーマを分析して、要件を満たす固有索引があるかどうかも判別できます。 ID リゾルバーが固有索引を探すのは、分析されるテーブルのエントリーがプロパティー・ファイル内にない場合や、プロパティー・ファイルそのものがない場合だけです。これらの条件が整うと、固有索引のチェックが実行されます。固有索引は、それが存在しており、テーブルの基本キーを含んでいなければ、有効と見なされます。

- **プロパティー・ファイル指定**

  ID リゾルバーでは、1 次行の ID が必要なテーブルの検索にどの基本エントリーの列を使用するかを記述するために、代替の Java プロパティー・ファイルを使用します。

WebSphere Commerce に用意されているサンプル・ストア・アーカイブは、 XML ファイルで内部別名を使用しています。そのため、データベース間でストア・アーカイブの移植が可能になっています。固有索引やプロパティー・ファイルによる指定方法でも、データベース間の移植性を実現できますが、その場合は、ユーザーが固有列をいつでも変更できるため、後で ID の解決のためにそれらの方法を使用するときに問題が発生します。たとえば、ユーザーが固有列を変更すると、プロパティー・ファイルの定義でもその列名を変更する必要があります。しかし、内部別名の方法を使った場合は、データベース内の変更によって、 XML ファイルやプロパティー・ファイルの変更が必要になるわけではありません。 WebSphere Commerce 管理コンソールやローダー・パッケージを使用して発行が行われる際、ID リゾルバーは別名を固有な値に置き換えます。データがロードされると、別名はユーザーからは認識されません。詳細については、489 ページの『付録 B. データの作成』を参照してください。

ID リゾルバーは以下のプロセスを使用します。

- 入力 XML データに、すでにハードコーディング ID (例えば、″12345″) がある基本テーブルのエレメントが含まれている場合、ID リゾルバーは、そのエレメントに新しい ID を作成しません。
- 入力 XML データに、ID のない基本テーブルのエントリーが含まれている場合、 ID リゾルバーは、データベースを調べてこのエレメントの行がすでに存在するかどうかを確認します。

データベースでのエレメントの検索には、固有キーを作成するために使用される、エレメント内の他の列が必要です。これらの他の列は、プロパティー・ファイルで指定できます。あるいは、ID リゾルバーに使用する列を決定させることも可能です。

- プロパティー・ファイルが使用されていて、そのプロパティー・ファイル内に分析するテーブルのエントリーがある場合、 ID リゾルバーは、プロパティー・ファイルで指定された列を使用して固有キーを作成します。
- 使用されているプロパティー・ファイルがないか、分析されるテーブルのエントリーがプロパティー・ファイルにない場合、ID リゾルバーは、固有索引解決法を使用します。

  固有索引解決法では、ID を見つける手段として、テーブル上の指定された固有索引のうち、任意のものが使用されます。たとえば、CATALOG テーブルでは MEMBER_ID と IDENTIFIER が固有索引になるため、これを CATALOGDSC テーブルの外部キー CATALOG_ID の解決点として用いることができます。

同じ固有キーを持つ行が存在する場合、そのエレメントはすでにデータベース内に存在しているものと判断されます。そのような行が存在しなければ、エレメントは新しいデータと見なされます。

- エレメントがデータベースに行としてすでに存在する場合は、その ID が取り出され、後で使用できるように保存されます。エレメントがデータベースにない場合は、ID リゾルバーにより、 KEYS テーブルか SUBKEYS テーブルの使用可能な値を使用して新しい ID が生成されます。
- XML 文書でエレメントに内部別名 (例えば、″@store_id_1″ など) が指定されている場合は、同じ内部別名を使用して後で ID を検索できるよう、その別名と ID が関連付けられます。
- XML 文書内の後続のエレメントが 1 次テーブルのエレメントを参照する必要が生じた場合、 1 次テーブルのエレメントに内部別名があれば、その内部別名 (例えば、″@store_id_1″) を使用し、内部別名がなければ、検索列の値 (例えば、″@WC2001@100″) を使用します。どちらの場合でも、指定された値を使用して実際の ID が検索され、検出された ID で値が置き換えられます。
- 出力 XML 文書が生成される時点では、すべての基本テーブルのエレメントが実際の ID を持つようになり、それらの基本テーブルのエレメントを参照するすべてのエレメントは、上で言及した内部別名や検索列の値ではなく、実際の ID を使用して参照を行うようになります。これが、XML 文書が完全に解決された状態です。

ID Resolve コマンドの方式:

ID Resolve コマンドでは、入力ファイルを処理するための方式として、ロード方式、更新方式、混合方式のいずれかを選択できます。

ロード方式:

ID リゾルバーのロード方式は、データベースにロードされるすべての新規レコードの新規 ID を生成するために使用されます。

注: ID リゾルバーのロード方式を指定する場合、入力ファイル内のレコードが事前にデータベース内に存在するべきではありません。 ID リゾルバーとともにロード方式を使用し、ソース XML ファイル内のレコードがすでにターゲット・

データベースに存在していると、ローダーはデータのロード時にエラーを生成します。 ID リゾルバーは ID 解決時に XML ファイル内のレコードに新規の基本キーを割り当てます。ただし、データベースにデータをロードする際、エラーが生成されます。ローダーは重複するレコードを処理する時点でも停止しませんが、エラーを報告し、重複するレコードはデータベースにロードしません。

以下の例を使用すると、データベースに対して新しいデータ・エレメントの ID が生成されます。

- AIX Linux 400 Solaris

```
./idresgen.sh -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method load -customizer customizer -schemaname wcsadmin
```

- Windows

```
idresgen -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method load -customizer customizer -schemaname wcsadmin
```

注: 該当する ID Resolve コマンドまたはスクリプトの場所については、 416 ページの『ローダー・パッケージ・コマンドおよびスクリプトの使用』を参照してください。

更新方式:
ID リゾルバーの更新方式を指定する場合、入力ファイル内のレコードは事前にデータベース内に存在している必要があります。 ID リゾルバーは、 389 ページで説明されている方法で、データベース内の ID を探し出します。データベースにレコードが存在しない場合、 ID リゾルバーはそのレコードの ID を解決することができず、エラーが発生したと指摘します。

以下の例を使用すると、データベースにすでに存在するデータ・エレメントの ID が探し出されます。

- AIX Linux 400 Solaris

```
./idresgen.sh -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method update -customizer customizer -schemaname wcsadmin
```

- Windows

```
idresgen -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method update -customizer customizer -schemaname wcsadmin
```

注: 該当する ID Resolve コマンドまたはスクリプトの場所については、 416 ページの『ローダー・パッケージ・コマンドおよびスクリプトの使用』を参照してください。

混合方式:
入力データ・ファイルに、すでにデータベースに存在するレコードと新規レコードが含まれる場合、混合方式を使用して ID リゾルバーを実行しなければなりません。この方式を使用すると、ID リゾルバーはレコードがデータベースに存在しない場合にのみ、レコードの新規 ID を作成します。新規 ID が作成されない場合に、データベースから既存の ID を取得します。以下の例を使用すると、新規データの ID が生成され、データベースにすでに存在するデータ・エレメントの ID が探し出されます。

- AIX Linux 400 Solaris

```
./idresgen.sh -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method mixed -customizer customizer -schemaname wcsadmin
```

- Windows

```
idresgen -dbname db -dbuser user -dbpwd pwd -infile input.xml
-outfile output.xml -method mixed -customizer customizer -schemaname wcsadmin
```

注:

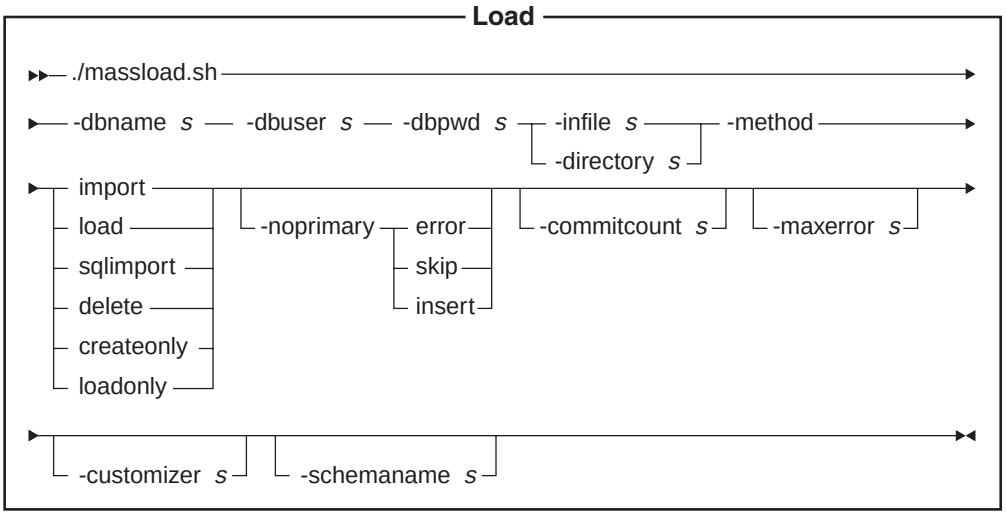
1. 該当する ID Resolve コマンドまたはスクリプトの場所については、 416 ページの『ローダー・パッケージ・コマンドおよびスクリプトの使用』を参照してください。
2. 混合方式は、管理コンソールで推奨される方式です。

このコマンドを実行するために使用するファイルの設定とカスタマイズの詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

Load コマンド

このコマンドは、XML 入力ファイルをターゲット・データベースにロードします。

AIX Linux 400 Solaris



注:

1. 上の図は、主にコマンド・パラメーターを参照するための図です。
2. このコマンドのパラメーターとして指定するファイル名の先頭には、相対パスや絶対パスを付けてもかまいません。

パラメーター値:

-dbname

AIX Linux Solaris ターゲット・データベースの名前

400 これは、リレーショナル・データベース・ディレクトリー (WRKRDBDIRE) で表示されるのと同じ名前です。

-dbuser

AIX Linux Solaris データベースに接続しているユーザーの名前。

400 これは通常、インスタンス・ユーザー名と同じです。

-dbpwd

データベースに接続しているユーザーのパスワード

-infile 入力 XML ファイルの名前

-directory

後述の -method パラメーターに loadonly 以外のオプションを使用している場合は、 -infile パラメーターを使用します。 loadonly 方式を使用するときは、-infile パラメーターを -directory パラメーターに置き換えることが必要で、置き換えないとエラーになります。loadonly 方式を使用するときの -directory パラメーターの値には、

後述の createonly 方式で作成された MassLoadOutputFiles ディレクトリーの完全修飾パス指定します。

-method

入力データを使用してデータベースを変更する際にローダーが使用する操作の方式

load ロード (load) 方式では、データベース・ベンダーのネイティブ・ローダーが使用されます。 Oracle データベースの場合、ロード方式はローカルとリモートの両方で使用できますが、 DB2 データベースの場合、ロード方式はローカルでしか使用できません。

400 ロード・メソッドは、ビット・データまたは DBCLOB フィールドをサポートしません。

import

インポート (import) 方式では、インポート (import) または更新 (update) のオプションがデータベース・ベンダーで使用できる場合には、それが使用されます。インポートまたは更新のオプションが使用できない場合、 JDBC を使用する SQL ステートメントを使用してデータベースが更新されます。デフォルトはインポート (import) です。

400 インポート・メソッドが使用できるのは、ローカル・データベースのみです。

sqlimport

SQL インポート (sqlimport) 方式は、ローカル・データベースでもリモート・データベースでも使用できます。

delete

削除 (delete) 方式では、データベースからデータが削除されます。

createonly

インスタンス作成時のパフォーマンスを向上させるには、createonly 方式を使用します。 createonly 方式を使用すると、データをデータベースにロードせずに、一括ロード・データ (MLD) ファイルを作成できます。この方式を使用して作成されたファイル (.mld and .cmd ファイル) は、 MassLoadOutputFiles というディレクトリーに置かれます。 AIX Linux Solaris このディレクトリーは Load コマンドを実行するディレクトリーのサブディレクトリーとして作成され、作業ディレクトリーになります。したがって、作業ディレクトリーは書き込み可能でなければなりません。

400 このディレクトリーは、インスタンスのルート・ディレクトリーにある temp ディレクトリーのサブディレクトリーとして作成されます。ディレクトリーのデフォルト位置は、*WC_userdir*/instances/*instance_name*/temp/MassLoadOutputFiles です。

次に示すのは、createonly 方式を使用して Load コマンドを実行する例です。

```
./massload.sh -dbname mall -dbuser db2admin -dbpwd db2admin
-infile WC_installdir/data/example.xml
-method createonly
```

400

```
./massload.sh -dbname mall -dbuser db2admin -dbpwd db2admin
-infile WC_userdir/data/example.xml
-method createonly
```

その後、ネイティブ・データベース・ロード・ユーティリティーから、後述の loadonly 方式を使用して Load コマンドを実行することにより、作成した MLD ファイルを WebSphere Commerce データベースにロードすることができます。

注: プログラムは、データベース製品が使用するネイティブ・データベース・ロード・ユーティリティーに関する情報を、カスタマイザー・プロパティー・ファイルから入手します。

loadonly

前述の createonly 方式を使用して作成された MLD ファイルをロードするには、 loadonly 方式を使用します。 loadonly 方式を使用するときは、-directory パラメーターを使用することが必要で、使用しないとエラーになります。

注: loadonly 以外の方式を使用した場合は、指定した -infile パラメーターに代わって -directory パラメーターが使用されます。

-directory パラメーターの値には、 createonly 方式を使用して作成された MassLoadOutputFiles ディレクトリーの完全修飾パス指定します。

次に示すのは、loadonly 方式 (および必須の -directory パラメーター) を使用して Load コマンドを実行する例です。

```
./massload.sh -dbname mall -dbuser db2admin -dbpwd db2admin
-method loadonly -directory WC_installdir/bin/MassLoadOutputFiles
-schemaname wcsadmin
```

この方式を使用して Load コマンドを実行する場合は、必ず -schemaname パラメーターを使用してターゲット・データベース・スキーマの名前を指定します。指定しないと、プログラムは MassLoadOutputFiles ディレクトリーとそのファイルが最初に作成されたときに取得したデータベース・スキーマの名前を使用します。
loadonly メソッドを使用すると、エラーやその他のメッセージは .log の拡張子を持つファイルに保管されます。これらのログ・ファイルは、-directory パラメーターに指定した MassLoadOutputFiles ディレクトリーに書き込まれます。
loadonly 方式は、インスタンス作成**のみ**に使用してください。それ以外に使用すると、期待どおりの結果にならないことがあります。

-noprimary

入力ファイルの中でレコードの基本キーが欠落している場合に、ローダーが実行するアクション

- error オプションを指定した場合、欠落している基本キーをエラーとして報告して終了します。
- skip オプションの場合、入力ファイルの中に基本キーのないレコードがあればスキップされます。
- insert オプションの場合は、データの挿入または削除が試行されます。

このパラメーターはオプションです。デフォルトのアクションは error です。

-commitcount

SQL import 動作方式を使用している場合、ここに指定する数のレコードが処理されたなら、データベースのコミットが発生します。このパラメーターはオプションです。デフォルトの数は 1 です。

-maxerror

SQL import 動作方式において、ここに指定する数のエラーが発生すると、ローダーは終了します。このパラメーターはオプションです。デフォルト値は 1 です。

-customizer

使用するカスタマイザー・プロパティー・ファイルの名前。このパラメーターはオプションです。カスタマイザー・プロパティー・ファイルでは、ローダー・プログラムの動作方法を設定します。デフォルト・ファイルは次のとおりです。

AIX Linux Solaris MassLoadCustomizer.properties

400 `ISeries_LODWCSDTA_Customizer.properties`

インスタンスをツールボックス・ドライバーを使用するように構成した場合は、ツールボックス・ドライバーに提供されている `Toolbox_LODWCSDTA_Customizer` カスタマイザー・ファイルを使用してください。 -dbname パラメーターにホスト名を指定することも必要です。`massload.sh` スクリプトを呼び出す例を挙げます。

```
./massload.sh -dbname MY.HOSTNAME.CA -dbuser instance -dbpwd mypass
-method sqlimport -customizer Toolbox_LODWCSDTA_Customizer
-infile /path/file.xml
```

次の例のようにカスタマイザー・プロパティー・ファイルを指定できます。

AIX Linux Solaris

```
-customizer WC_installdir/my_directory/file_name.properties
-customizer WC_installdir/my_directory/file_name
```

400

```
-customizer WC_userdir/my_directory/file_name.properties
-customizer WC_userdir/my_directory/file_name
```

現行ディレクトリーにこのファイルがある場合は、次の例に示すようにして同じファイルを指定できます。

```
-customizer file_name.properties
```

クラスパスのシステム環境変数で指定されたディレクトリーにこのファイルが存在する場合、同じファイルを次の例のように指定できます。

-customizer *file_name*

ここで、*my_directory* はユーザー定義のディレクトリー、 *file_name* は使用するプロパティー・ファイルです。新しいカスタマイザー・プロパティー・ファイルを作成して指定するための詳細については、 WebSphere Commerce オンライン・ヘルプを参照してください。

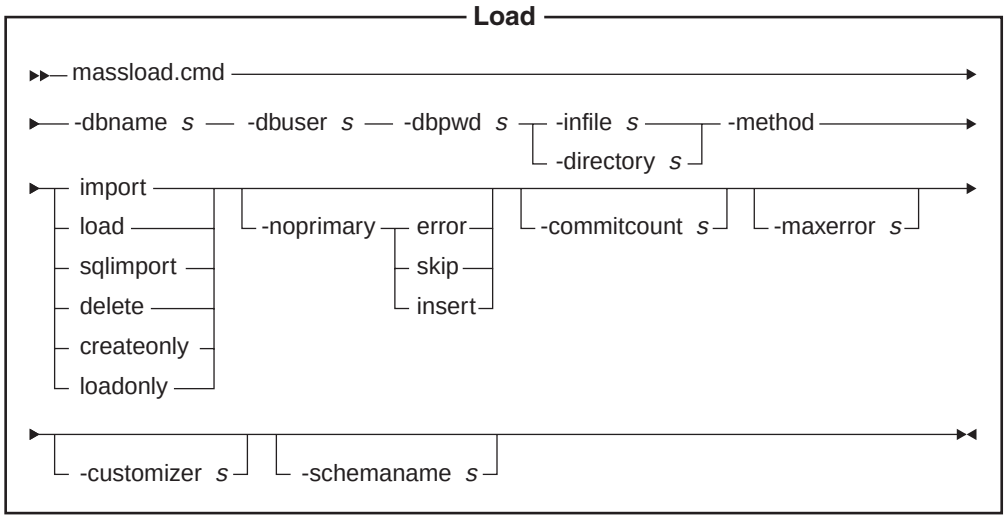
-schemaname

ターゲット・データベース・スキーマの名前。このパラメーターはオプションです。

このパラメーターを指定しないでコマンドを実行すると、カスタマイザー・プロパティー・ファイル内で、 SchemaName の値を指定した「名前 = 値」のペアが検索されます。プロパティー・ファイルにそのペアが存在すれば、そこに指定されている値が使用されます。このパラメーターをコマンド行でも指定せず、プロパティー・ファイルにもその指定がなければ、デフォルトで、データベース内の KEYS テーブルのスキーマ名が使用されます。

400 -schemaname パラメーターをコマンド行でも指定せず、プロパティー・ファイルにもその指定がなければ、デフォルトで、-dbuser パラメーターの値が使用されます。

Windows



注: このコマンドのパラメーターとして指定するファイル名の先頭には、相対パスや絶対パスを付けてもかまいません。

パラメーター値:

-dbname

ターゲット・データベースの名前

-dbuser

データベースに接続しているユーザーの名前

-dbpwd

データベースに接続しているユーザーのパスワード

-infile 入力 XML ファイルの名前

-directory

後述の -method パラメーターに loadonly 以外のオプションを使用している場合は、 -infile パラメーターを使用します。 loadonly 方式を使用するときは、-infile パラメーターを -directory パラメーターに置き換えることが必要で、置き換えないとエラーになります。loadonly 方式を使用するときの -directory パラメーターの値には、後述の createonly 方式で作成された MassLoadOutputFiles ディレクトリーの完全修飾パス指定します。

-method

データをデータベースに挿入する際にローダーが使用する操作の方式

load ロード (load) 方式では、データベース・ベンダーのネイティブ・ローダーが使用されます。 Oracle データベースの場合、ロード方式はローカルとリモートの両方で使用できますが、 DB2 データベースの場合、ロード方式はローカルでしか使用できません。インポート (import) 方式では、ローカル・データベースにもリモート・データベースにもデータをロードできますが、基本的には、リモート DB2 データベースにデータをロードするときに使用します。

import

インポート (import) 方式では、インポート (import) または更新 (update) のオプションがデータベース・ベンダーで使用できる場合には、それが使用されます。インポートまたは更新のオプションが使用できない場合、 JDBC を使用する SQL ステートメントを使用してデータベースが更新されます。デフォルトはインポート (import) です。

sqlimport

SQL インポート (sqlimport) 方式は、ローカル・データベースでもリモート・データベースでも使用できます。

delete

削除 (delete) 方式では、データベースからデータが削除されます。

createonly

インスタンス作成時のパフォーマンスを向上させるには、createonly 方式を使用します。 createonly 方式を使用すると、データをデータベースにロードせずに、一括ロード・データ (MLD) ファイルを作成できます。この方式を使用して作成されたファイル (.mld and .cmd ファイル) は、 MassLoadOutputFiles というディレクトリーに置かれます。このディレクトリーは Load コマンドを実行するディレクトリーのサブディレクトリーとして作成され、作業ディレクトリーになります。したがって、作業ディレクトリーは書き込み可能でなければなりません。

次に示すのは、createonly 方式を使用して Load コマンドを実行する例です。

```
massload -dbname mall -dbuser db2admin -dbpwd db2admin -infile
WC_installdir¥data¥example.xml
-method createonly
```

その後、ネイティブ・データベース・ロード・ユーティリティーから、後述の loadonly 方式を使用して Load コマンドを実行することにより、作成した MLD ファイルを WebSphere Commerce データベースにロードすることができます。

注: プログラムは、データベース製品が使用するネイティブ・データベース・ロード・ユーティリティーに関する情報を、カスタマイザー・プロパティー・ファイルから入手します。

loadonly

前述の createonly 方式を使用して作成された MLD ファイルをロードするには、 loadonly 方式を使用します。 loadonly 方式を使用するときは、-directory パラメーターを使用することが必要で、使用しないとエラーになります。

注: loadonly 以外の方式を使用した場合は、指定した -infile パラメーターに代わって -directory パラメーターが使用されます。

-directory パラメーターの値には、 createonly 方式を使用して作成された MassLoadOutputFiles ディレクトリーの完全修飾パス指定します。

次に示すのは、loadonly 方式 (および必須の -directory パラメーター) を使用して Load コマンドを実行する例です。

```
massload -dbname mall -dbuser db2admin -dbpwd db2admin -method
loadonly -directory WC_installdir¥bin¥MassLoadOutputFiles
-schemaname wcsadmin
```

この方式を使用して Load コマンドを実行する場合は、必ず -schemaname パラメーターを使用してターゲット・データベース・スキーマの名前を指定します。指定しないと、プログラムは MassLoadOutputFiles ディレクトリーとそのファイルが最初に作成されたときに取得したデータベース・スキーマの名前を使用します。
loadonly メソッドを使用すると、エラーやその他のメッセージは .log の拡張子を持つファイルに保管されます。これらのログ・ファイルは、-directory パラメーターに指定した MassLoadOutputFiles ディレクトリーに書き込まれます。
loadonly 方式は、インスタンス作成**のみ**に使用してください。それ以外に使用すると、期待どおりの結果にならないことがあります。

-noprimary

入力ファイルの中でレコードの基本キーが欠落している場合に、ローダーが実行するアクション

- error オプションを指定した場合、欠落している基本キーをエラーとして報告して終了します。
- skip オプションの場合、入力ファイルの中に基本キーのないレコードがあればスキップされます。
- insert オプションの場合は、データの処理 (挿入または削除) が試行されます。

このパラメーターはオプションです。デフォルトのアクションは error です。

-commitcount

SQL import 動作方式を使用している場合、ここに指定する数のレコードが処理されたなら、データベースのコミットが発生します。このパラメーターはオプションです。デフォルトの数は 1 です。

-maxerror

SQL import 動作方式において、ここに指定する数のエラーが発生すると、ローダーは終了します。このパラメーターはオプションです。

-customizer

使用するカスタマイザー・プロパティー・ファイルの名前。このパラメーターはオプションです。カスタマイザー・プロパティー・ファイルでは、ローダー・プログラムの動作方法を設定します。デフォルトのファイルは MassLoadCustomizer.properties です。次の例のようにカスタマイザー・プロパティー・ファイルを指定できます。

```
-customizer WC_installdir¥my_directory¥file_name.properties
```

現行ディレクトリーにこのファイルがある場合は、次の例に示すようにして同じファイルを指定できます。

```
-customizer file_name.properties
```

クラスパスのシステム環境変数で指定されたディレクトリーにこのファイルが存在する場合、同じファイルを次の例のように指定できます。

```
-customizer file_name
```

ここで、*my_directory* はユーザー定義のディレクトリー、 *file_name* は使用するプロパティー・ファイルの名前です。新しいカスタマイザー・プロパティー・ファイルを作成して指定するための詳細については、WebSphere Commerce オンライン・ヘルプの最新版を参照してください。

-schemaname

ターゲット・データベース・スキーマの名前。このパラメーターはオプションです。

このパラメーターを指定しないでコマンドを実行すると、カスタマイザー・プロパティー・ファイル内で、 SchemaName の値を指定した「名前 = 値」のペアが検索されます。プロパティー・ファイルにそのペアが存在すれば、そこに指定されている値が使用されます。このパラメーターをコマンド行でも指定せず、プロパティー・ファイルにもその指定がなければ、デフォルトで、データベース内の KEYS テーブルのスキーマ名が使用されます。

Load コマンドの方式:

データをロードする前に、使用可能な処理方式のうちのどの方式を使用すると最も良い結果が得られるか判別する必要があります。

ロード方式:

次のいずれかの状況におけるロード方式を検討してください。

- ソース・データがクリーンであり、データベースにデータが入っていない

  注: クリーン・データとは、データのロード先のテーブルの制約を違反していないデータのことです。

- ソース・データがクリーンであり、ロード中のデータがデータベースに入っていない

- ソース・データがクリーンであり、 1 つ以上のターゲット・テーブルに基本キーが含まれておらず、ロード中のデータがデータベースに入っていない
- データベースがローカル DB2 データベースである
- データベースがローカルまたはリモートの Oracle データベースである
- ロード中、他のユーザーまたはアプリケーションからデータベースにアクセスされない

400 ロード方式を使用すると、データはデータベースにロードされます。データがすでに存在する場合、重複キー・エラーの結果としてコマンドは失敗し、重複エラーのメッセージが表示されます。

ロード方式を使用する際、以下の制限があります。

- 400 ロード方式はビット・データ・フィールドまたは DBCLOB フィールドでデータの挿入または更新ができない。データベースへの新規レコードの挿入だけが行われる。既存のレコードの場合はエラーが発生する。
- DB2 ロード方式を使用すると、データベースへの新規レコードの挿入だけが行われる。既存のレコードの更新は行われない。ロード方式は、ローカルの DB2 データベースだけに使用できるものであって、リモートのデータベースには使用できません。

インポート方式:

AIX Linux Solaris Windows DB2 でインポート方式を使用する場合も、データはデータベースにロードされます。データがすでに存在する場合、削除はされないものの、新しい値で更新されます。次のいずれかの状況におけるこの方式を検討してください。

- データベース管理システムが DB2 である
- データがクリーンかどうか分からない
- 同種データの多数セットを列レベルで更新する必要がある
- データのインポート先の全テーブルに基本キーがある

400 iSeries でインポート方式を使用する場合も、データはデータベースにロードされます。データがすでに存在する場合、削除はされないものの、新しい値で更新されます。次のいずれかの状況におけるこの方式を検討してください。

- データがクリーンかどうか分からない
- データがすでにデータベース内に存在する
- データのインポート先の全テーブルに基本キーがある

インポート方式を使用する際、以下の制限があります。

- インポート方式を使用するためには、データベース管理システムが DB2 でなければならない。
- 400 インポート方式はビット・データ・フィールドまたは DBCLOB フィールドでデータの挿入または更新ができず、ローカル・データベースでしか使用できない。

- インポート方式の場合、ローダーは基本キーが定義されているテーブルだけを挿入または更新を行う。この方式は、基本キーを持たないテーブルにデータを挿入したり、更新したりできません。入力レコードの列に 1 次の値しかない場合、そのレコードは拒否されます。

SQL インポート方式:
SQL インポート方式の場合、データを更新したりデータベースにデータを挿入したりするために、 JDBC または SQL ステートメントが使用されます。データが存在しない場合には挿入され、データが存在する場合、それらのデータは更新されます。次のいずれかの状況におけるこの方式を検討してください。
- 既存のデータを更新しており、列レベルの更新を必要とする
- データの中にクリーンではないものがある
- データベースがローカルにない

注: 商品アドバイザー検索スペースの同期を使用する場合には、データのロードに SQL インポート方式を使用する必要があります。

削除方式:
削除方式は、入力 XML 文書にあるデータをデータベースから削除するために使用します。エレメントには、基本キーの値かまたはテーブルの固有索引が含まれていなければなりません。「カスケード削除」を使用可能にして、従属データを別のテーブルに持つデータを削除すると、その従属データも削除されます。

AIX、Linux、Solaris、および Windows システム用の createonly 方式:
インスタンス作成時のパフォーマンスを向上させるには、 createonly 方式を使用します。 createonly 方式を使用すると、データをデータベースにロードせずに、一括ロード・データ (MLD) ファイルを作成できます。その後、ネイティブ・データベース・ロード・ユーティリティーから、 loadonly 方式を使用して Load コマンドを実行することにより、作成した MLD ファイルを WebSphere Commerce データベースにロードすることができます。

AIX、Linux、Solaris、および Windows システム用の loadonly 方式:
createonly 方式を使用して作成された MLD ファイルをロードするには、 loadonly 方式を使用します。 loadonly 方式は、インスタンス作成**のみ**に使用してください。それ以外に使用すると、期待どおりの結果にならないことがあります。

方式の比較:
- **SQL インポート方式とロード方式の比較**

  SQL インポート方式は、データの整合性を検査し (外部参照を含む)、既存のデータの更新を可能にしますが、ロード方式はこれを行いません。
- **インポート方式と SQL インポート方式の比較**

  インポート方式と SQL インポート方式のどちらも同じ機能を実行します。一般に、インポート方式の方が早いですが、一時ファイル用のディスク・スペースを必要とします。

  インポート方式は、基本キーを定義したテーブルだけを挿入または更新しますが、 SQL インポート方式では、テーブルに基本キーを定義する必要はありません。
- **使用されるデータベース製品に基づく方式の比較**

インポート方式とロード方式は、 DB2 用に最適化された固有のユーティリティーを使用しますが、 SQL インポート方式は、 JDBC 呼び出し (これは多数のデータベース製品で一般的) を使用します。

パフォーマンスに関する考慮事項:
ローダーを使用して大量の文書をデータベースにロードする際は、以下の項目を考慮してください。

- **Java 仮想マシン (JVM) のヒープ・サイズ**

  JVM ヒープに割り当てられているデフォルトの最大メモリー量は 64 MB です。これを増やさない場合、 JVM はロード・プロセス中にメモリー不足になってしまう可能性があります。 Java ヒープに割り当てられている最大メモリー量は、Java コマンドの JVM -mx オプションで変更できます。

- **トレースのロギング**

  トレース・ロガーは大量の XML 文書のロードするときに、 JVM ヒープを使い果たしてしまう可能性があります。トレース情報は大抵、実行が失敗した場合にそれをデバッグするために使用されます。ロード・プロセスのトレースが不要な場合は、トレースをオフにしてください。トレースをオフにすると、パフォーマンスは著しく向上します。ロギング構成 XML 文書を変更することで、トレースをオフにします。ロギング構成 XML 文書の変更の詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

- **コミット数**

  ローダーが SQL インポート・モードを操作するときのデフォルトのコミット数は 1 です。したがって、デフォルトではデータベースを更新するたび、またはデータベースに追加するたびにトランザクションがコミットされます。大量文書を扱う場合のローダーのパフォーマンスを向上させるには、コミット数を増やす必要があります。 `input.xml` ファイルのサイズを考慮に入れて、ファイル中でレコード数より大きいコミット・カウントを使用できます。これによって、エラーが発生した場合に `input.xml` ファイル全体のロールバックが可能です。

  ローダーのコミット数は Load コマンドの -commitcount *count* オプションを使用して変更します (*count* は、トランザクションをコミットする前に実行されるステートメントの数です)。

- **ロギング構成**

  まれに、以下の状況のいずれかのために、データ・ロード中の進行が低速になることがあります。

  – ローダーを呼び出したユーザーに、ディレクトリーに対する書き込み許可がない、あるいはロギング構成文書で指定されたファイルの更新許可がない。
  – ファイルのロケーションとしてロギング構成文書で指定されたディレクトリーが存在しない。
  – ファイルのロケーションとしてロギング構成文書で指定されたドライブに十分のスペースがない。

  これらの問題のいくつかを訂正するときに、ファイルのロケーションを変更するためにロギング構成文書 (デフォルトでは WCALoggerConfig.xml) を変更するが必要がある場合があります。ロギング構成 XML 文書の変更の詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

このコマンドを実行するために使用するファイルの設定とカスタマイズの詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

データの変換および抽出に関するローダー・パッケージ・コマンド

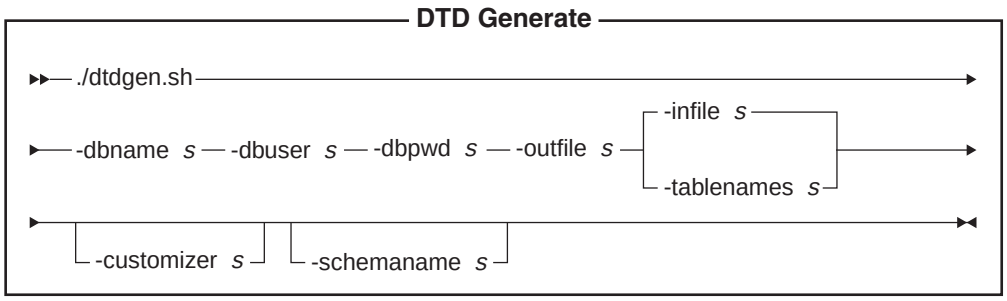
DTD Generate コマンド

このコマンドは、ローダー・パッケージで使用する DTD を作成します。その DTD は、ロード・プロセスを通じて使用されることになります。コマンドの呼び出し方によって、 DTD ジェネレーターが DTD だけを生成する場合と、 DTD とともに XML スキーマを生成する場合があります。

DTD ジェネレーターは、 WebSphere Commerce データベース・スキーマに基づいて DTD を作成することができます。サンプル・ストア・アーカイブに含まれている DTD を使用し、データベース・スキーマを変更しない場合、 DTD ジェネレーターを使用して DTD を生成する必要はありません。提供される DTD は、*WC_installdir*/xml/sar ディレクトリーにあります。

提供されている DTD を使用することをお勧めします。しかし、データベース・スキーマをカスタマイズする場合は、提供されている DTD を変更内容と一致するように編集するか、または DTD を新しく作成するかしなければなりません。

AIX Linux 400 Solaris



注:

1. 上の図は、主にコマンド・パラメーターを参照するための図です。
2. このコマンドのパラメーターとして指定するファイル名の先頭には、相対パスや絶対パスを付けてもかまいません。

パラメーター値:

-dbname

AIX Linux Solaris ターゲット・データベースの名前

400 これは、リレーショナル・データベース・ディレクトリー (WRKRDBDIRE) で表示されるのと同じ名前です。

-dbuser

AIX Linux Solaris データベースに接続しているユーザーの名前。

400 これは通常、インスタンス・ユーザー名と同じです。

-dbpwd

データベースに接続しているユーザーのパスワード

-outfile

出力 DTD ファイルの名前 (.dtd 拡張子を伴う場合がある)

-infile 各行にデータベース・テーブル名を含む入力ファイルの名前

-tablenames

コンマで区切られ、引用符 ("") で囲まれるテーブルの名前

-customizer

使用するカスタマイザー・プロパティー・ファイルの名前。このパラメーターはオプションです。カスタマイザー・プロパティー・ファイルでは、DTD ジェネレーターの動作方法を設定します。デフォルト・ファイルは次のとおりです。

AIX Linux Solaris `DB2ConnectionCustomizer.properties`

400 `ISeries_GENWCSDTD_Customizer.properties`

インスタンスをツールボックス・ドライバーを使用するように構成した場合は、ツールボックス・ドライバーに提供されている `Toolbox_GENWCSDTD_Customizer` カスタマイザー・ファイルを使用してください。 -dbname パラメーターにホスト名を指定することも必要です。`dtdgen.sh` スクリプトを呼び出す例を挙げます。

```
 ./dtdgen.sh -dbname MY.HOSTNAME.CA -dbuser instance -dbpwd mypass
-outfile /path/out.dtd
  method sqlimport -customizer Toolbox_GENWCSDTD_Customizer
 -infile /path/file.xml
```

次の例のようにカスタマイザー・プロパティー・ファイルを指定できます。

AIX Linux Solaris

`-customizer` *`WC_installdir/my_directory/file_name`*`.properties`

`-customizer` *`WC_installdir/my_directory/file_name`*

400

`-customizer` *`WC_userdir/my_directory/file_name`*`.properties`

`-customizer` *`WC_userdir/my_directory/file_name`*

現行ディレクトリーにこのファイルがある場合は、次の例に示すようにして同じファイルを指定できます。

```
-customizer file_name.properties
```

クラスパスのシステム環境変数で指定されたディレクトリーにこのファイルが存在する場合、同じファイルを次の例のように指定できます。

```
-customizer file_name
```

ここで、*my_directory* はユーザー定義のディレクトリー、 *file_name* は使用するプロパティー・ファイルの名前です。新しいカスタマイザー・プロパティー・ファイルを作成して指定するための詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

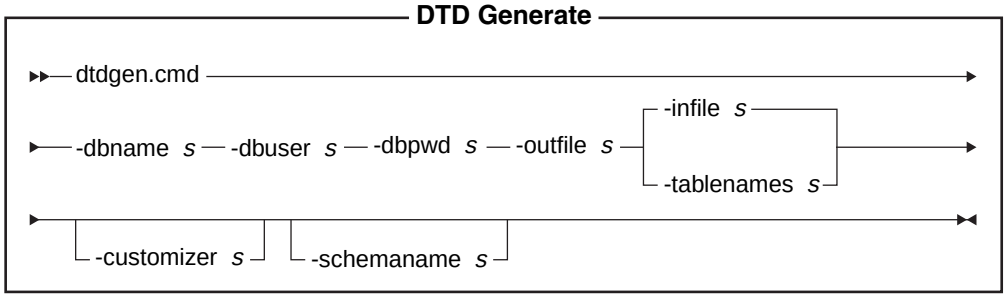
-schemaname

ターゲット・データベース・スキーマの名前。このパラメーターはオプションです。

このパラメーターを指定しないでコマンドを実行すると、カスタマイザー・プロパティー・ファイル内で、 SchemaName の値を指定した「名前 = 値」のペアが検索されます。プロパティー・ファイルにそのペアが存在すれば、そこに指定されている値が使用されます。このパラメーターをコマンド行でも指定せず、プロパティー・ファイルにもその指定がなければ、デフォルトで、データベース内のテーブルのスキーマ所有者が使用されます。

400 このパラメーターをコマンド行でも指定せず、プロパティー・ファイルにもその指定がなければ、デフォルトで、データベース・ユーザーの名前が使用されます。

Windows



注: このコマンドのパラメーターとして指定するファイル名の先頭には、相対パスや絶対パスを付けてもかまいません。

パラメーター値:

-dbname

ターゲット・データベースの名前

-dbuser

データベースに接続しているユーザーの名前

-dbpwd

データベースに接続しているユーザーのパスワード

-outfile
出力 DTD ファイルの名前

-infile 各行にデータベース・テーブル名を含む入力ファイルの名前

-tablenames
コンマで区切られるテーブルの名前

-customizer
使用するカスタマイザー・プロパティー・ファイルの名前。このパラメーターはオプションです。カスタマイザー・プロパティー・ファイルでは、DTD ジェネレーターの動作方法を設定します。
DB2ConnectionCustomizer.properties がデフォルトのファイルです。次の例のようにカスタマイザー・プロパティー・ファイルを指定できます。

```
-customizer WC_installdir¥my_directory¥file_name.properties
```

```
-customizer WC_installdir¥my_directory¥file_name
```

現行ディレクトリーにこのファイルがある場合は、次の例に示すようにして同じファイルを指定できます。

```
-customizer file_name.properties
```

クラスパスのシステム環境変数で指定されたディレクトリーにこのファイルが存在する場合、同じファイルを次の例のように指定できます。

```
-customizer file_name
```

ここで、*my_directory* はユーザー定義のディレクトリー、 *file_name* は使用するプロパティー・ファイルの名前です。新しいカスタマイザー・プロパティー・ファイルを作成して指定するための詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

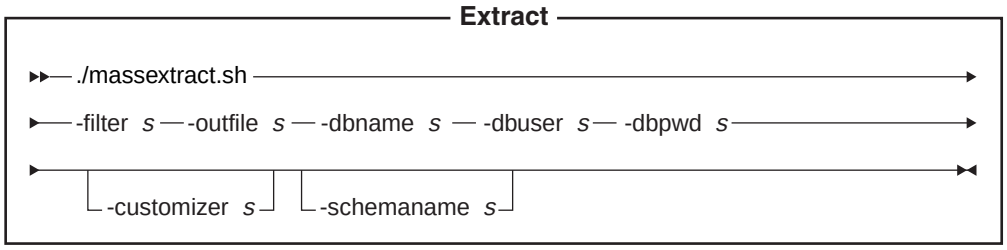
-schemaname
ターゲット・データベース・スキーマの名前。このパラメーターはオプションです。このパラメーターを指定しないでコマンドを実行すると、カスタマイザー・プロパティー・ファイル内で、 SchemaName の値を指定した「名前 = 値」のペアが検索されます。プロパティー・ファイルにそのペアが存在すれば、そこに指定されている値が使用されます。このパラメーターをコマンド行でも指定せず、プロパティー・ファイルにもその指定がなければ、デフォルトで、データベース内のテーブルのスキーマ所有者が使用されます。

このコマンドを実行するために使用するファイルの設定とカスタマイズの詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

Extract コマンド
このコマンドは、データベースから選択されたデータのサブセットを XML ファイルの形式で抽出します。

抽出プログラムを使用してデータベースからデータを抽出するには、抽出フィルター・ファイルを使用して、抽出したいデータを指定しなければなりません。使用する抽出フィルターは、抽出したいデータの種類によって異なります。

AIX Linux 400 Solaris



注:

1. 上の図は、主にコマンド・パラメーターを参照するための図です。
2. このコマンドのパラメーターとして指定するファイル名の先頭には、相対パスや絶対パスを付けてもかまいません。

パラメーター値:

-filter 抽出フィルター・ファイルの名前

-outfile
抽出データが保管される出力 XML ファイルの名前

-dbname

AIX Linux Solaris データ抽出元のデータベースの名前

400 これは、リレーショナル・データベース・ディレクトリー (WRKRDBDIRE) で表示されるのと同じ名前です。

-dbuser

AIX Linux Solaris データ抽出元のデータベースのデータベース・ユーザー名

400 これは通常、インスタンス・ユーザー名と同じです。

-dbpwd
データ抽出元のデータベースのユーザー名に関連したパスワード

-customizer
使用するカスタマイザー・プロパティー・ファイルの名前。カスタマイザー・プロパティー・ファイルでは、抽出プログラムの動作方法を設定します。デフォルト・ファイルは次のとおりです。

AIX Linux Solaris DB2ConnectionCustomizer.properties

400 ISeries_EXTWCSDTA_Customizer.properties インスタンスをツールボックス・ドライバーを使用するように構成した場合は、ツールボックス・ドライバーに提供されている Toolbox_EXTWCSDTA_Customizer カスタマイザー・ファイルを使用してください。 -dbname パラメーターにホスト名を指定することも必要です。 massextract.sh スクリプトを呼び出す例を挙げます。

```
./massextract.sh -dbname MY.HOSTNAME.CA -dbuser instance -dbpwd mypass
 -filter/path/filter.xml
 -outfile /path/file.xml -customizer Toolbox_EXTWCSDTA_Customizer
```

次の例のようにカスタマイザー・プロパティー・ファイルを指定できます。

AIX Linux Solaris DB2ConnectionCustomizer.properties

```
-customizer WC_installdir/my_directory/file_name.properties
-customizer WC_installdir/my_directory/file_name
```

400

```
-customizer WC_userdir/my_directory/file_name.properties
-customizer WC_userdir/my_directory/file_name
```

現行ディレクトリーにこのファイルがある場合は、次の例に示すようにして同じファイルを指定できます。

```
-customizer file_name.properties
```

クラスパスのシステム環境変数で指定されたディレクトリーにこのファイルが存在する場合、同じファイルを次の例のように指定できます。

```
-customizer file_name
```

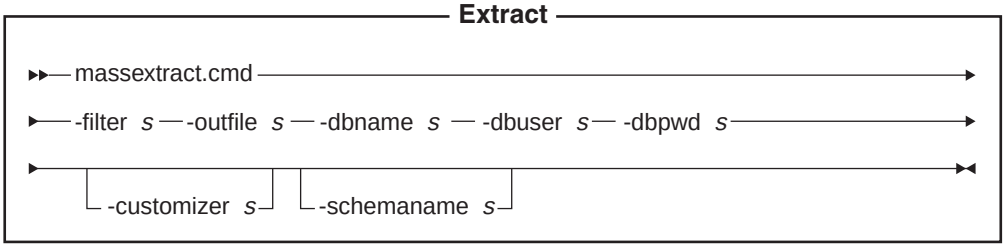
ここで、*my_directory* はユーザー定義のディレクトリー、 *file_name* は使用するプロパティー・ファイルの名前です。新しいカスタマイザー・プロパティー・ファイルを作成して指定するための詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

-schemaname

データ抽出元のデータベース・スキーマの名前。このパラメーターはオプションです。

このパラメーターを指定しないでコマンドを実行すると、カスタマイザー・プロパティー・ファイル内で、 SchemaName の値を指定した「名前 = 値」のペアが検索されます。プロパティー・ファイルにそのペアが存在すれば、そこに指定されている値が使用されます。このパラメーターをコマンド行でも指定せず、プロパティー・ファイルにもその指定がなければ、デフォルトで、データベース内のテーブルのスキーマ名が使用されます。 400 -schemaname パラメーターをコマンド行でも指定せず、プロパティー・ファイルにもその指定がなければ、デフォルトで、-dbuser パラメーターの値が使用されます。

Windows



注: このコマンドのパラメーターとして指定するファイル名の先頭には、相対パスや絶対パスを付けてもかまいません。

パラメーター値:

-filter　抽出フィルター・ファイルの名前

-outfile
抽出データが保管される出力 XML ファイルの名前

-dbname
データ抽出元のデータベースの名前

-dbuser
データ抽出元のデータベースのデータベース・ユーザー名

-dbpwd
データ抽出元のデータベースのユーザー名に関連したパスワード

-customizer
使用するカスタマイザー・プロパティー・ファイルの名前。カスタマイザー・プロパティー・ファイルでは、抽出プログラムの動作方法を設定します。 DB2ConnectionCustomizer.properties がデフォルトのファイルです。次の例のようにカスタマイザー・プロパティー・ファイルを指定できます。

```
-customizer WC_installdir¥my_directory¥file_name.properties
```

```
-customizer WC_installdir¥my_directory¥file_name
```

現行ディレクトリーにこのファイルがある場合は、次の例に示すようにして同じファイルを指定できます。

```
-customizer file_name.properties
```

クラスパスのシステム環境変数で指定されたディレクトリーにこのファイルが存在する場合、同じファイルを次の例のように指定できます。

```
-customizer file_name
```

ここで、*my_directory* はユーザー定義のディレクトリー、 *file_name* は使用するプロパティー・ファイルの名前です。新しいカスタマイザー・プロパティー・ファイルを作成して指定するための詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

-schemaname
データ抽出元のデータベース・スキーマの名前。このパラメーターはオプションです。

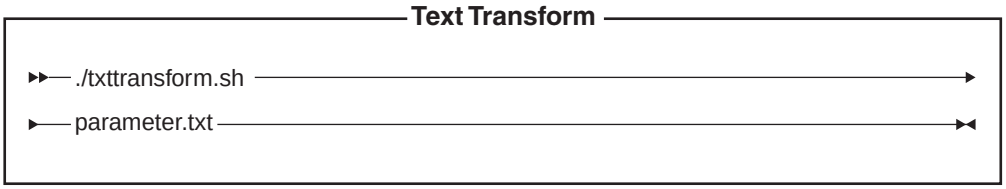
このパラメーターを指定しないでコマンドを実行すると、カスタマイザー・プロパティー・ファイル内で、 SchemaName の値を指定した「名前 = 値」のペアが検索されます。プロパティー・ファイルにそのペアが存在すれば、そこに指定されている値が使用されます。このパラメーターをコマンド行でも指定せず、プロパティー・ファイルにもその指定がなければ、デフォルトで、データベース内のテーブルのスキーマ名が使用されます。

このコマンドの詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

Text Transform コマンド

このコマンドは、文字区切り可変長形式のデータと XML 形式のデータの間で変換を行います。

AIX Linux 400 Solaris



注: 上の図は、主にコマンド・パラメーターを参照するための図です。

パラメーター値:
以下の値がパラメーター・ファイル (*parameter.txt*) の中で指定され、コンマによって区切られています。

input file
変換されるファイルの名前

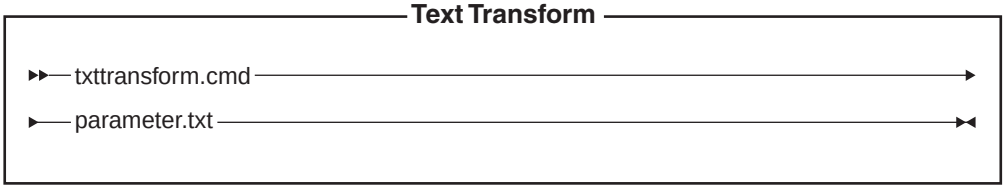
schema file
変換で使用される XML スキーマ・ファイルの名前

output file
変換データが保管される出力ファイルの名前

transformation method
データを出力ファイルに追加する際に使用される方式。新しいファイルを作成する場合は **Create** を、出力データを既存のデータ・ファイルに不可する場合は **Append** を指定してください。

このファイルは、「マニフェスト」ファイルまたは「コマンド」ファイルとも呼ばれます。これには、4 つのパラメーターごとの複数行が含まれることもあります。

Windows



パラメーター値:
以下の値がパラメーター・ファイル (*parameter.txt*) の中で指定され、コンマによって区切られています。

input file
変換されるファイルの名前

schema file
変換で使用される XML スキーマ・ファイルの名前

output file
変換データが保管される出力ファイルの名前

transformation method
データを出力ファイルに追加する際に使用される方式。新しいファイルを作

成する場合は **Create** を、出力データを既存のデータ・ファイルに不可する場合は **Append** を指定してください。

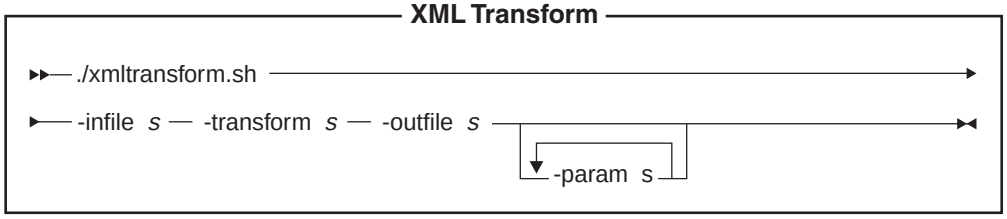
注: このファイルは、「マニフェスト」ファイルまたは「コマンド」ファイルとも呼ばれます。これには、4 つのパラメーターごとの複数行が含まれることもあります。

このコマンドの詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

XML Transform コマンド

このコマンドは、XML ファイルを別の XML 形式に変換します。

AIX Linux 400 Solaris



注:

1. 上の図は、主にコマンド・パラメーターを参照するための図です。
2. このコマンドのパラメーターとして指定するファイル名の先頭には、相対パスや絶対パスを付けてもかまいません。

パラメーター値:

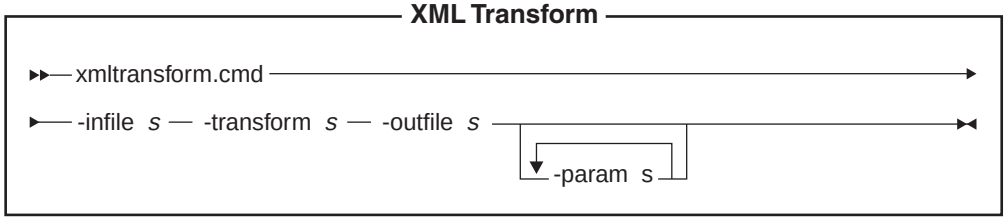
-infile 変換されるファイルの名前

-transform
変換 XSL マッピング・ルール・ファイル

-outfile
変換データが保管される出力 XML ファイルの名前

-param
XSL マッピング・ルール・ファイルに渡されるパラメーター。***このパラメーターはオプションです。*** 複数の name=value の対に渡すために、このパラメーターは何回でも指定できます。

Windows



注: このコマンドのパラメーターとして指定するファイル名の先頭には、相対パスや絶対パスを付けてもかまいません。

パラメーター値:

-infile 変換されるファイルの名前

-transform
変換 XSL マッピング・ルール・ファイル

-outfile
変換データが保管される出力 XML ファイルの名前

-param
XSL マッピング・ルール・ファイルに渡されるパラメーター。***このパラメーターはオプションです。*** 複数の name=value の対に渡すために、このパラメーターは何回でも指定できます。

このコマンドの詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

ローダー・パッケージ・コマンドに関連したツール

テキスト変換ツール

テキスト変換ツールは、 Text Transform コマンドを使用して、文字区切り変数フォーマットと XML フォーマットの間で、データ変換の処理を行うのに役立ちます。このツールには、以下のビューが備えられています。

1. 「テキスト・スキーマ・ファイルの編集 (Text Schema Edit)」ビュー。このビューで、変換に使用される XML スキーマ・ファイルを作成および変更することができます。
2. 「トランスフォーメーション・コマンドの編集 (Transformation Command Edit)」ビュー。このビューで、変換プロセスを実行するために使用する実際のコマンドを作成および変更することができます。
3. 「トランスフォーメーション・コマンド・プロセス (Transformation Command Process)」ビュー。このビューで、変換プロセスを立ち上げることができます。

このツールの詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

XSL エディター

XML 変換プログラムは、 XSL を使用して XML ファイルを別の XML ファイルに変換するためのルールを定義します。 XSL エディターにあるマッピング機能にはビジュアル・インターフェースが備えられており、これを使用してソース DTD 内のエレメントとターゲット DTD 内のエレメントとの関連付けを行うことができます。 2 つの DTD が備えられていますが、最初の (ソース) DTD に準拠する XML ファイルから 2 番目の (ターゲット) DTD に準拠するファイルへの変換方法を決定する XSL ルールを作成できます。

このツールの詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

ストア・データのロード

このセクションでは、ローダー・パッケージ・コマンド行ユーティリティーを使用して、WebSphere Commerce Server データベースにデータをロードする方法の例を示します。

注:

1. このセクションの例は、 Windows 環境で実行されます。その他の環境におけるこれらのコマンドの実行の詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。
2. ローダー・パッケージ・コマンド行ユーティリティーは DB2、DB2 for iSeries および Oracle データベースをサポートしていますが、以下の例では DB2 のコマンドとオプションだけを取り上げています。 DB2 以外のデータベースを使用している場合は、WebSphere Commerce オンライン・ヘルプにある説明に従って、カスタマイザー・プロパティー・ファイルを確実に変更してください。

WebSphere Commerce のデータベース資産グループのロード・プロセスについては、 429 ページの『第 38 章 WebSphere Commerce データベース資産グループのロード』 と437 ページの『データベース資産グループのロード』を参照してください。

ローダー・パッケージ・コマンドおよびスクリプトの使用

ローダー・パッケージ・コマンドを使用するには、 WebSphere Commerce 内の *WC_installdir*/bin ディレクトリーにあるスクリプトまたはコマンドを使用します。

スクリプトおよびコマンドは以下のとおりです。

- AIX Linux 400 Solaris

dtdgen.sh	DTD Generate シェル・スクリプト
idresgen.sh	ID Resolve シェル・スクリプト
massload.sh	Load シェル・スクリプト
massextract.sh	Extract シェル・スクリプト
txttransform.sh	Text Transform シェル・スクリプト
xmltransform.sh	XML Transform シェル・スクリプト

- Windows

dtdgen.cmd	DTD Generate コマンド
idresgen.cmd	ID Resolve コマンド
massload.cmd	Load コマンド
massextract.cmd	Extract コマンド
txttransform.cmd	Text Transform コマンド
xmltransform.cmd	XML Transform コマンド

ID 解決の例

ここで取り上げる ID 解決の例では、 Business ToolTech サンプル・ストアのストア資産ファイルを使用しています。

この例は、 WebSphere Commerce Server データベースに新しいストア資産をロードするというシナリオに基づいているので、ロード方式を使用することにします。

後から XML 文書内の特定のエレメントを変更する必要が生じた場合は、更新方式で変更できます。更新方式では新規の ID が割り当てられないので、ロード方式と比べて実行時間がかかりません。更新方式では、データベース・クエリーを実行して ID を探し出し、 ID が検出されない場合にはエラーが報告されます。このプロセスの動作の詳細については、 389のページから始まる説明を参照してください。

入力 XML ファイルの中に、データベースにすでに存在するエレメントと存在しないエレメントが混在する場合には、混合方式を使用します。混合方式では、まずデータベースの検索が行われ、レコードが検出されない場合に ID がエレメントに割り当てられます。混在しているかどうか分からない場合は、混合方式を使用してください。混合方式と比べてロード方式と更新方式の方が高いパフォーマンスを示しますが、混合方式を使用して生成された解決 XML ファイルの方が、ロード時にエラーが発生する可能性は低くなります。

ID リゾルバーの動作方法については、 390 ページの『ID Resolve コマンドの方式』を参照してください。

内部別名を使用した XML ファイルの ID 解決

内部別名を使用するデータを WebSphere Commerce Server データベースにロードする前にその ID を解決するには、以下の例のように ID Resolve コマンドを実行します。

注: この例では、WebSphere Commerce が Windows マシンにインストールされていると想定しています。別のオペレーティング・システム上に WebSphere Commerce をインストールしてある場合は、ご使用のオペレーティング・システムに合わせて適切な値に置き換えてください。

1. 作業ディレクトリーを作成します。

   この例では、*WC_installdir*/test ディレクトリーを作成します。

   注: *WC_installdir*/test を作業ディレクトリーとして使用しない場合は、この章の残りに部分に示されている例で *WC_installdir*/test の代わりに実際に使用する作業ディレクトリーの名前とパスに置き換えてください。

2. ID リゾルバーが検出できる場所に入力 XML ファイルと参照 DTD ファイルがあることを確認してください。

   この例では、以下のようにします。

   a. Windows コマンドで、以下のコマンドを入力します。

      ```
      copy WC_installdir¥samplestores¥
      B2BDirect¥B2BDirect.sar WC_installdir¥test
      ```

      これにより、B2BDirect.sar ファイルが *WC_installdir*¥test にコピーされます。

   b. Windows コマンド・プロンプトで、以下のコマンドを入力します。

      ```
      cd to WC_installdir¥test
      ```

   c. 以下のいずれかを行います。

      - Java がインストールされている場合は、 Windows コマンド・プロンプトから次のコマンドを実行します。

        ```
        jar -xvf B2BDirect.sar
        ```

これにより、企業向け ToolTech サンプル・ストアの XML ファイルが drive:¥WebSphere¥CommerceServer55¥test に抽出されます。

- 最新の unzip プログラム (WinZip や PKZIP など) を使用して、*WC_installdir*¥samplestores¥B2BDirect¥B2BDirect.sar の内容全体を *WC_installdir*¥test に抽出します。

これにより、Business ToolTech サンプル・ストアの XML ファイルが *WC_installdir*¥test¥WEB-INF¥stores¥BusinessDirect¥data¥ToolTech¥data に抽出されます。

d. Windows コマンドで、以下のコマンドを入力します。

```
copy WC_installdir¥xml¥sar¥store.dtd
WC_installdir¥test¥WEB-INF¥stores¥BusinessDirect¥data¥ToolTech¥data
```

これにより、store.dtd ファイルが *WC_installdir*¥test¥WEB-INF¥stores¥BusinessDirect¥data¥ToolTech¥data にコピーされます。

e. Windows コマンドで、以下のコマンドを入力します。

```
copy WC_installdir¥xml¥sar¥DBLoadMacros.dtd
WC_installdir¥test¥WEB-INF¥stores¥BusinessDirect¥data¥ToolTech¥data
```

これにより、DBLoadMacros.dtd ファイルが *WC_installdir*¥test¥WEB-INF¥stores¥BusinessDirect¥data¥ToolTech¥data にコピーされます。

f. Windows コマンドで、以下のコマンドを入力します。

```
copy WC_installdir¥xml¥sar¥fulfillment.dtd
WC_installdir¥test¥WEB-INF¥stores¥BusinessDirect¥data¥ToolTech¥data
```

これにより、fulfillment.dtd ファイルが *WC_installdir*¥test¥WEB-INF¥stores¥BusinessDirect¥data¥ToolTech¥data にコピーされます。

3. WebSphere Commerce Server データベースの適切なインスタンスを作成したときに、 WebSphere Commerce スキーマと必要なブートストラップ・データがロードされたことを確認します。

注: インスタンスの作成については、各オペレーティング・システム版の「*WebSphere Commerce インストール・ガイド*」を参照してください。

この例では、 *mall* という WebSphere Commerce Server データベース・インスタンスを使用します。基本キーと外部キーは、このデータベースの KEYS テーブルと SUBKEYS テーブルから取り込まれるので、このデータベースが正しくロードされないと、 ID リゾルバーは ID を解決できません。

4. fulfillment.xml ファイルの ID を解決するには、以下のようにします。

a. fulfillment.xml ファイルを編集し、以下を含めます。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE  fulfillment-asset SYSTEM "fulfillment.dtd">
<fulfillment-asset>
</fulfillment-asset>
```

fulfillment.xml は以下のようになります。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE  fulfillment-asset SYSTEM "fulfillment.dtd">
<fulfillment-asset>
<!--defaultshipoffset can be overridden in the STORITMFFC table.-->
<!--Now in ToolTech STORITMFFC.shippingoffset is set to 86400
  seconds which is one day--<
<ffmcenter
   ffmcenter_id="@ffmcenter_id_1"
   member_id="&MEMBER_ID;"
   name="ToolTech Home"
    defaultshipoffset="0"
    markfordelete="0"
 />
</fulfillment-asset>
```

b. もし以下の行がなければ、DBLoadMacros.dtd ファイルを編集して含めます。

```
<!ENTITY MEMBER_ID "-2001">
```

c. 以下のコマンドを入力して、 (FFMCENTER テーブルが定義されている) fulfillment.xml ファイルに対して ID リゾルバーを実行します。

```
idresgen -dbname mall -dbuser db2admin -dbpwd db2admin
-infile WC_installdir¥test¥WEB-INF¥stores¥BusinessDirect¥data¥ToolTech¥data¥
fulfillment.xml
-outfile WC_installdir¥test¥WEB-INF¥stores¥BusinessDirect¥data¥ToolTech¥data
¥fulfillment1.xml
-method load
```

ここで、

- モールを使用していない場合は、*mall* をターゲット・データベースの名前に変更する必要があります。
- db2admin を使用していない場合は、最初の *db2admin* をデータベースに接続しているユーザーの名前に変更する必要があります。
- db2admin を使用していない場合は、2 番目の *db2admin* をデータベースに接続しているユーザーのパスワードに変更する必要があります。

解決された fulfillment1.xml 出力のエレメントは、次のようになります。

```
<fulfillment-asset>
<ffmcenter
    FFMCENTER_ID="10001"
    MEMBER_ID="-2001"
   NAME="ToolTech Home"
   DEFAULTSHIPOFFSET="0"
    MARKFORDELETE="0"
/>
</fulfillment-asset>
```

注: これはサンプルです。実際の出力ファイルに含まれる値は、これとは異なっている場合があります。

5. store.xml ファイルの ID を解決するには、以下のようにします。

a. store.xml ファイルを編集し、以下を含めます。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE store-asset SYSTEM "store.dtd">
<store-asset>
</store-asset>
```

b. 生成された出力ファイル (fulfillment1.xml) から FFMCENTER_ID キーを取得し、

WC_installdir¥test¥WEB-INF¥stores¥BusinessDirect¥data¥ToolTech¥data にある store.xml の作業コピーの中のすべての @ffmcenter_id_1 を、ここで取得したキーに置換してください。

c. もし以下の行がなければ、DBLoadMacros.dtd ファイルを編集して含めます。

```
<!ENTITY MEMBER_ID "-2001">
<!ENTITY STORE_IDENTIFIER "ToolTech">
<!ENTITY STORE_DIR "ToolTech">
```

d. 以下のコマンドを入力します。

```
idresgen -dbname mall -dbuser db2admin -dbpwd db2admin
-infile WC_installdir¥test¥WEB-INF¥stores¥
BusinessDirect¥data¥ToolTech¥data¥store.xml-outfile
WC_installdir¥test¥WEB-INF¥stores¥
BusinessDirect¥data¥ToolTech¥data¥store1.xml -method load
```

ここで、

- モールを使用していない場合は、*mall* をターゲット・データベースの名前に変更する必要があります。
- db2admin を使用していない場合は、最初の *db2admin* をデータベースに接続しているユーザーの名前に変更する必要があります。
- db2admin を使用していない場合は、2 番目の *db2admin* をデータベースに接続しているユーザーのパスワードに変更する必要があります。

完全に解決された store1.xml 出力ファイルのエレメントは、次のようになります。

```
<store-asset>
<storeent
    STOREENT_ID="10151"
    MEMBER_ID="-2001"
    TYPE="S"
    IDENTIFIER="ToolTech"
    SETCCURR="USD"
 />
<store
    STORE_ID="10151"
   DIRECTORY="ToolTech"
    FFMCENTER_ID="10001"
    LANGUAGE_ID="-1"
   STOREGRP_ID="-1"
   ALLOCATIONGOODFOR="43200"
    BOPMPADFACTOR="0"
   DEFAULTBOOFFSET="2592000"
   FFMCSELECTIONFLAGS="0"
   MAXBOOFFSET="7776000"
   REJECTEDORDEXPIRY="259200"
    RTNFFMCTR_ID="10001"
   PRICEREFFLAGS="0"
   STORETYPE="B2B"
 />
<vendor
    VENDOR_ID="10001"
    STOREENT_ID="10151"
    VENDORNAME="Tooltech Vendor"
    MARKFORDELETE="0"
 />
<dispentrel
    AUCTIONSTATE="0"
    CATENTRY_ID="0"
    CATENTTYPE_ID="ProductBean"
    DEVICEFMT_ID="-1"
```

```
    DISPENTREL_ID="10001"
    MBRGRP_ID="0"
    PAGENAME="CatalogProductDisplay.jsp"
    STOREENT_ID="10151"
    RANK="0"
 />
<dispentrel
    AUCTIONSTATE="0"
    CATENTRY_ID="0"
    CATENTTYPE_ID="ItemBean"
    DEVICEFMT_ID="-1"
    DISPENTREL_ID="10002"
    MBRGRP_ID="0"
    PAGENAME="CatalogItemDisplay.jsp"
    STOREENT_ID="10151"
    RANK="0"
 />
<dispcgprel
    CATGROUP_ID="0"
    DEVICEFMT_ID="-1"
    DISPCGPREL_ID="10001"
    MBRGRP_ID="0"
    PAGENAME="CatalogCategories.jsp"
    STOREENT_ID="10151"
    RANK="0"
 />
<invadjcode
    ADJUSTCODE="PCNT"
    INVADJCODE_ID="10001"
    MARKFORDELETE="0"
    STOREENT_ID="10151"
 />
<invadjcode
    ADJUSTCODE="SPLG"
    INVADJCODE_ID="10002"
    MARKFORDELETE="0"
    STOREENT_ID="10151"
 />
<invadjcode
    ADJUSTCODE="DISC"
    INVADJCODE_ID="10003"
    MARKFORDELETE="0"
    STOREENT_ID="10151"
 />
<rtnreason
    REASONTYPE="C"
    RTNREASON_ID="10001"
    STOREENT_ID="10151"
    MARKFORDELETE="0"
    CODE="WPR"
 />
<rtnreason
    REASONTYPE="B"
    RTNREASON_ID="10002"
    STOREENT_ID="10151"
    MARKFORDELETE="0"
    CODE="DEF"
 />
<rtnreason
    REASONTYPE="M"
    RTNREASON_ID="10003"
    STOREENT_ID="10151"
    MARKFORDELETE="0"
    CODE="ERR"
 />
<rtnreason
    REASONTYPE="M"
```

```
        RTNREASON_ID="10004"
        STOREENT_ID="10151"
        MARKFORDELETE="0"
        CODE="WPS"
     />
    </store-asset>
```

注: これはサンプルです。実際の出力ファイルに含まれる値は、これとは異なっている場合があります。

6. store.xml ファイルに、以下のエレメントがあります。

```
<storeent
   STOREENT_ID="@storeent_id_1"
   MEMBER_ID="@seller_b2b_mbr_id"
    TYPE="S"
   IDENTIFIER="&STORE_IDENTIFIER"
   SETCURR="USD"
/>
```

store.xml のこのエレメントは、データベース内の storeeent テーブルにマッピングされており、そのエレメントの属性 STOREENT_ID MEMBER_ID、TYPE、IDENTIFIER、SETCURR は、そのテーブルの各列にマッピングされています。指定値の @storeent_id_1 は、 STOREENT_ID 属性の値の内部別名であり、&MEMBER_ID; は、エンティティー・パラメーターです。 &MEMBER_ID; エンティティーの値は、ローダーを使用してロードする前に置き換える必要があります。 &MEMBER_ID; の値は、 DBLoadMacros.dtd マクロ・ファイルで定義されており、そのファイルに基づいて値が置き換えられます。 ID リゾルバーは、@storeent_id_1 を見つけると、 1 次テーブルのキャッシュの中に storeent が存在するかどうかを調べます。これは 1 次テーブルなので、storeeent は存在します。 ID リゾルバーはそのテーブルのカウンターを取り出し、その値を増分して内部別名と置き換えます。 store.xml ファイル内にあるこの種の他のすべてのエントリーも、これと同じように処理されます。

7. 416 ページの『ローダー・パッケージ・コマンドおよびスクリプトの使用』でリストされている適切な ID Resolve コマンドまたはスクリプトを含むディレクトリーがパスに含まれていることを確認します。

   この例では、 Windows コマンド・プロンプトから以下のコマンドを入力します。

   ```
   cd WC_installdir¥bin
   ```

   ここで、*WC_installdir*¥bin がシステムの *WC_installdir*¥bin にない場合は、ID Resolve コマンド idresgen.cmd を含むディレクトリーの名前に変更してください。

8. Windows コマンドで、以下のコマンドを入力します。

   ```
   idresgen -dbname mall -dbuser wcs -dbpwd wcs1 -infile
   WC_installdir¥test¥WEB-INF¥stores¥BusinessDirect¥data¥ToolTech¥
   data¥store.xml -outfile
   WC_installdir¥test¥WEB-INF¥stores¥BusinessDirect¥data¥ToolTech¥
   data¥store1.xml-method load
   ```

   ここで、

   - モールを使用していない場合は、*mall* をターゲット・データベースの名前に変更する必要があります。
   - db2admin を使用していない場合は、最初の *db2admin* をデータベースに接続しているユーザーの名前に変更する必要があります。

- db2admin を使用していない場合は、2 番目の *db2admin* をデータベースに接続しているユーザーのパスワードに変更する必要があります。

store1.xml の中の最初の出力 XML フラグメントは、次のようになります。

```
<storeent
   STOREENT_ID="10001"
    MEMBER_ID="-2001"
    TYPE="S"
    IDENTIFIER="ToolTech"
    SETCCURR="USD"
/>
```

注: これはサンプルです。実際の出力ファイルに含まれる値は、これとは異なっている場合があります。

store1.xml の中の 2 番目の XML フラグメントは、次のようになります。

```
<store
   STORE_ID="10001"
   DIRECTORY="ToolTech"
   FFMCENTER_ID=""
    LANGUAGE_ID="-1"
   STOREGRP_ID="-1"
   ALLOCATIONGOODFOR="43200"
    BOPMPADFACTOR="0"
   DEFAULTBOOFFSET="2592000"
   FFMCSELECTIONFLAGS="0"
   MAXBOOFFSET="7776000"
   REJECTEDORDEXPIRY="259200"
   RTNFFMCTR_ID=""
   PRICEREFFLAGS="0"
   STORETYPE="B2B"
/>
```

注: これはサンプルです。実際の出力ファイルに含まれる値は、これとは異なっている場合があります。

以下のオプションのいずれかを使用して ID を解決できます。

- オプション 1:
  a. fulfillment.xml だけに存在する内容 (fulfillment.dtd への参照も含む) を store.xml に追加する形で、 fulfillment.xml ファイルと store.xml ファイルをマージしてから、以下の ffmcenter エレメントが store エレメントの前に来ていることを確認します。

     ```
     <ffmcenter
        FFMCENTER_ID="@ffmcenter_id_1"
        MEMBER_ID="&MEMBER_ID;"
        NAME="ToolTech Home"
        DEFAULTBOOFFSET="0"
         MARKFORDELETE="0"
     />
     ```

  b. マージされたファイルに対して ID リゾルバーを実行します。
- オプション 2: 437 ページの『データベース資産グループのロード』で説明されているプロセスを使って、ストア資産データ・グループをロードします。

ID リゾルバーで使用するプロパティー・ファイルの指定

-propfile パラメーターを使用すると、ID リゾルバーの解決方法を変更することができます。デフォルトのプロパティー・ファイルは IdResolveKeys.properties で

す。 IdResolveKeys.properties を変更して使用するには、このファイルをユーザー定義のディレクトリーにコピーし、必要な変更を加えてから、ID Resolve コマンドの呼び出し時にこの新規ファイルを指定します。デフォルトの IdResolveKeys.properties ファイルは、 *WC_installdir*/properties ディレクトリーにあります。

プロパティー・ファイルの指定は、内部別名の使用よりも優先されます。

以下は、store.xml ファイルのサンプル XML フラグメントです。

```
<store
   STORE_ID="@storeent_id_1"
   DIRECTORY="ToolTech"
   FFMCENTER_ID="@ffmcenter_id_1"
   LANGUAGE_ID="&en_US;"
   STOREGRP_ID="-1"
   ALLOCATIONGOODFOR="43200"
   BOPMPADFACTORr="0"
   DEFAULTBOOFFSET="2592000"
   FFMCSELECTIONFLAGS="0"
   MAXBOOFFSET="7776000"
   REJECTEDORDEXPIRY="259200"
   RTNFFMCTR_ID="@ffmcenter_id_1"
   PRICEREFFLAGS="0"
   STORETYPE="B2B"
/>
```

WC_installdir¥test¥WEB-INF¥stores¥BusinessDirect¥data¥ToolTech¥data¥myPropFile として -propfile を指定して ID リゾルバーを実行し、指定したファイル myPropFile.properties に以下のエントリーが含まれていたとします。

```
NAMEDELIMETER=@
SELECTDELIMETER=:
FFMCENTER=@FFMCENTER_ID@MEMBER_ID:10051 -2001
```

この場合、ストア・エレメントが処理されるときに、 ID リゾルバーはデータベースを参照して 10051 および -2001 の where 文節を持つ FFMCENTER テーブルを探します。その後、この値に対して戻される索引を使用して、 FFMCENTER_ID の ID を解決します。

このコマンドの使用については、 383 ページの『ID Resolve コマンド』を参照してください。

データ・ロードの例

必要に応じて XML ファイルの ID を解決したら、 WebSphere Commerce Server データベースにデータをロードする準備が整ったことになります。

注: XML データ内の ID を正しく解決した後は、ソース XML ファイルに以下の内容は含まれていないはずです。

- アット・マーク (@) で始まる単語
- アンパーサンド (&) 記号で始まる単語
- 空の引用符 ("") だけの ID

これらの記号の存在は、 XML ファイルをロードする準備が整っていないことを示すものとなります。

このセクションで説明するデータ・ロードの例では、 416 ページの『ID 解決の例』で解決された fulfillment1.xml ファイルを使用します。

WebSphere Commerce Server データベースにデータをロードするには、以下の例に従って Load コマンドを実行します。

1. 作業ディレクトリーを作成します。

   この例では、416 ページの『ID 解決の例』で作成した *WC_installdir*¥test¥WEB-INF¥stores¥ BusinessDirect¥data¥ToolTech¥data¥ というディレクトリーを使用します。

2. ローダーが検出できる場所に入力 XML ファイルがあることを確認してください。

   この例では、 416 ページの『ID 解決の例』で作成した fulfillment1.xml 出力ファイルが *WC_installdir*¥test¥WEB-INF¥stores¥BusinessDirect¥data¥ToolTech¥data¥ にあることを確認してください。

3. WebSphere Commerce Server データベースのバックアップがあることを確認します。バックアップがあれば、回復不能エラーが発生してもデータベースを復元できます。

   注: データベースのバックアップの詳細については、データベース製品に添付されているバックアップおよび回復の資料を参照してください。

4. 416 ページの『ローダー・パッケージ・コマンドおよびスクリプトの使用』でリストされている適切な Load コマンドまたはスクリプトを含むディレクトリーが、パスに含まれていることを確認します。

   この例では、 Windows コマンド・プロンプトから以下のコマンドを入力します。

   ```
   cd WC_installdir¥bin
   ```

   ここで、*WC_installdir*¥bin がシステムの *WC_installdir*¥bin にない場合は、Load コマンド massload.cmd を含むディレクトリーの名前に変更してください。

5. 解決済みの XML ファイルに対して Load コマンドを実行して、データをターゲット・データベースにロードします。

   この例では、 Windows コマンド・プロンプトから以下のコマンドを入力します。

   ```
   massload -dbname mall -dbuser db2admin -dbpwd db2admin -infile
   WC_installdir¥test¥WEB-INF¥stores¥BusinessDirect¥data¥ToolTech
   ¥data¥fulfillment1.xml -method sqlimport -commitcount 50
   ```

   ここで、

   - モールを使用していない場合は、*mall* をターゲット・データベースの名前に変更する必要があります。
   - db2admin を使用していない場合は、最初の *db2admin* をデータベースに接続しているユーザーの名前に変更する必要があります。
   - db2admin を使用していない場合は、2 番目の *db2admin* をデータベースに接続しているユーザーのパスワードに変更する必要があります。

ロードするエレメントの数は 50 未満ですが、この例では -commitcount に 50 という値を指定しています。これは、パフォーマンス上の理由です。デフォルトでは、コミット数は 1 になっています。このデフォルトを使用すると、各レコードごとのコミット操作がデータベースに書き込まれることになります。上記の例でこの数を 50 に設定することによって、ロードが成功した場合にのみデータベースの入出力が行われ、エラーが発生した場合にはデータベースに何も書き込まれないようにすることができます。ただし、大量のデータをロードする場合には、コミット数にエレメントの数より高い値を設定しないようお勧めします。これには次のような理由があります。

- コミット数の値を高くすると、多くのメモリーが消費されます。
- コミット数の値がエレメントの数よりも小さい場合は、少なくとも一部のデータがデータベースに書き込まれます。 -maxerror の値に応じて -commitcount の値を小さくすることによって、エラーの最大数を超過してツールが終了する前に、いくつかのデータをデータベースに書き込むことができます。 -maxerror のデフォルト値は 1 です。

-noprimary オプションのデフォルトは error なので、基本キーが欠落しているときにツールがエラーを報告し、終了します。

これらの例では、管理コンソールで使用される順序、また430 ページの『データベース資産のロードの順序』で説明されている順序でストア資産をロードしていないため、 416 ページの『ID 解決の例』で作成される store1.xml ファイルは、いくつかのテーブルの保全性制約に違反する場合があります。ロード方式を使用して変更なしで store1.xml をロードしようとすると、制約違反によってデータベースが保留状態になります。しかしながら、Load コマンドを使用したこの例では、説明を簡単にするために、fulfillment.xml の解決済みバージョン (その外部キーは、サンプル・ストアで定義されている MEMBER_ID の外部キーのみ) に基づいています。この例では、 416 ページの『ID 解決の例』で出力された解決済みの fulfillment1.xml ファイルをロードし、 SQL インポート方式を使用しています。XML ファイルの内容がきれいになっているかどうかが分からない場合は、この例で取り上げた SQL インポート方式を使用し、 -commitcount パラメーターと -maxerror パラメーターを正しく設定して、データベースを変更することなく、またデータベースの保全性を危険にさらすことなく、データベース制約違反が報告されるようにしてください。

このコマンドを実行すると、トレース・テキスト・ファイル (trace.txt) がデフォルトで実行サブディレクトリー (上の例では *WC_installdir*¥bin) に作成されます。

400 では、trace.txt が QIBM/Userdata/CommerceServer55/instances/*instance_name*/logs にデフォルトで作成されます。

WCALoggerConfig.xml ロギング構成ファイルで trace.txt の保管場所が別の場所に変更されている場合は、その場所でファイルを検査してください。WCALoggerConfig.xml のカスタマイズの詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

trace.txt ファイルには、コマンドが実行したアクションとその結果がリストされます。上の例に示されているようにコマンドとともに SQL インポート方式を使用する場合は、コミットされたレコードの数を示すエントリーが trace.txt の末尾に追加されます。

このコマンドの使用については、 393 ページの『Load コマンド』を参照してください。

第 38 章 WebSphere Commerce データベース資産グループのロード

データベース資産を発行する前に、データベース資産をすべて作成してストア・アーカイブ・ファイルにパッケージしておくことを望まない場合は、 WebSphere Commerce ローダー・パッケージを使って、データベース資産グループをロードできます。

この章では、まず WebSphere Commerce のデータベース資産グループと、グループを設定する方法について説明します。さらに、データベース資産グループを WebSphere Commerce データベースにロードするプロセスについて解説します。この部分を読む前に、 379 ページの『第 37 章 ストア・データのロードの概要』の情報を十分に確認するようにしてください。ローダー・パッケージを使用して、データベース資産グループをロードするために必要な情報が書かれています。

データベース資産グループ

作成プロセスとロード・プロセスを単純化するため、データベース資産はグループに分けられています。このデータベース資産グループ は、論理的に関連したテーブルのセットです。データ間の関係をロードするには、データが存在していなければならないため、データベース資産グループを編成する順序は、ロードを実行する上で重要です。

ストアのデータベース資産の全体をロードするには、 430 ページの『データベース資産のロードの順序』の説明に従う必要があります。データベース資産の 1 つのグループをロードするには、そのグループが論理的に完全であることを確認する必要があります。たとえば、ストア・アーカイブを発行するときには、カタログ・データベース資産を省略することもできます。カタログ・データベース資産は、後で発行できるからです。この場合、カタログに従属するデータベース資産 (在庫、価格表、一部の配送データと税データ) も発行しないでおきます。省略したデータを発行するときには、カタログ・データベース資産が論理的に完全であることを確認する必要があります。つまり、基本アイテム、カタログ・エントリー、属性などがそろっている必要があるわけです。さらに、論理的に完全な従属データベース資産も発行する必要があります。要するに、各 SKU には、該当する在庫、価格、配送、税が定義されている必要があるということです。この場合は、論理的に完全な関連カタログ・データをまとめて、*カタログ・データベース資産グループ* といいます。

本書のこれまでの章で取り上げた WebSphere Commerce データベース資産も、グループに配列できます。グループとは、論理的に完全なデータ・セットであり、そのグループごとのロードが可能です。各データベース資産グループは、 WebSphere Commerce のデータベース・テーブルで構成されており、 491 ページの『付録 C. データベース資産グループ』で説明されている外部従属関係を持っています。このテーブル・リストは、 WebSphere Commerce サンプル・ストアに基づいていますが、あらゆる汎用ストアにも当てはまります。各データベース資産グループのテー

ブル・リストは完全なものではなく、一般的な指針に過ぎないことに注意してください。ストアに特定の要求に基づいて、組み込んだり除外したりしなければならないテーブルがあるかもしれません。

データベース資産のロードの順序

データベース資産グループを正しくロードするには、決まった順序を守る必要があります。各グループは構造的に完全であると見なされており、他のデータベース資産グループに従属してはいません。しかし、データベース資産グループ内には、外部キー関係が存在します。そのような関係 (他のグループのデータとの関係) をデータベース資産グループの*外部従属関係* といいます。

データベース資産グループを WebSphere Commerce データベースにロードするには、そのグループの外部従属関係を満たしておく必要があります。データベース資産グループの外部従属関係として定義されているグループがあれば、そのグループを最初にロードしなければなりません。外部従属関係および関連したテーブルのリストについては、491 ページの『データベース資産グループの従属関係』を参照してください。

注: WebSphere Commerce ストアには、ストア所有者が必要です。デフォルト所有者として用意されているデフォルト組織を使用することもできます。

Business このグループをロードする場合は、デフォルト組織を使用する代わりに、新しい組織を作成してください。

データベース資産グループをロードする順序は、次のとおりです。

1. ブートストラップ・データだけに従属しているデータベース資産グループ。
   a. **組織**データベース資産を最初にロードします。
2. フルフィルメント所有者に従属しているデータベース資産グループ。
   a. **フルフィルメント**・データベース資産。組織データベース資産グループを除き、その他のデータベース資産グループには、このグループに定義されているデータへの直接的または間接的な外部従属関係があります。
3. ストア所有者組織に従属するデータベース資産グループ。
   a. **アクセス制御**・データベース資産は、ストア所有者組織 (ORGENTITY_ID) に従属しています。その他のデータベース資産グループには、このグループに定義されたデータに対する従属関係がありません。したがって、アクセス制御データベース資産はいつロードしてもかまいません。しかし、アクセス制御所有者は、ストア所有者と同じでなければなりません。
   b. **ストア**・データベース資産は、ストア所有者組織 (ORGENTITY_ID) に従属しています。

   ストアとは、配送センターを指す場合もあります。したがって、ストア所有者組織は、実行センターの所有者組織である場合もあります。
4. ストア・データベース資産に従属するデータベース資産グループ。以下のグループは、どのような順序でロードしてもかまいません。
   a. **キャンペーン**・データベース資産。
   b. **コマンド**・データベース資産。
   c. **通貨**データベース資産。

d. **ポリシー**・データベース資産。

e. **配送**データベース資産。

f. **税**データベース資産。

5. その他のデータベース資産グループ。

a. **カタログ**・データベース資産は、配送データベース資産グループと税データベース資産グループに従属しています。

b. **ストア・デフォルト**・データベース資産には、配送データベース資産グループに対する外部従属関係があります。配送データベース資産グループが存在していなければ、このグループにデータを読み込む必要はありません。

c. **契約**データベース資産は、組織資産に従属しています。契約データベース資産は、直接ロードされません。詳細については、 445 ページの『契約資産の発行』を参照してください。契約資産は、その他のデータベース資産グループの後にロードするようにしてください。

WebSphere Commerce サンプル・ストアで作成されるデータベース資産グループの内容については、 491 ページの『付録 C. データベース資産グループ』を参照してください。

ストアのロード

WebSphere Commerce 5.5 には、データベース資産のロードに役立つサンプル・ストアが用意されています。ストア全体の XML データを WebSphere Commerce データベースにロードする手順は、次のとおりです。

1. 以下の情報を確認します。

a. 489 ページの『付録 B. データの作成』.

b. 491 ページの『付録 C. データベース資産グループ』。 WebSphere Commerce のどのデータベース資産ファイルとデータベース・テーブルが影響を受けるかを確認しておく必要があります。

c. 379 ページの『第 37 章 ストア・データのロードの概要』。ローダー・パッケージの背景情報が記されています。

2. ストア・データベース資産の完全セットのロード・プロセスを計画します。437 ページの『データベース資産グループのロード』の手順で 1 つのデータベース資産グループをロードする場合も、ストア全体をロードする場合も、基本的なプロセスは変わりません。次のステップでは、それぞれのロード・プロセスに合わせて、以下のファイルを使用または作成します。

a. 各グループの 1 つ以上のデータベース資産ファイル。ストア全体をロードする場合は、作成済みのすべてのデータベース資産ファイルが必要です。たとえば、 *database asset*.xml ファイル (campaign.xml、catalog.xml、currency.xml に入っているファイル) と、ストアがサポートしているロケールごとにロケール固有の *database asset*.xml ファイルが必要になります。そのようなファイルの例が、WebSphere Commerce のサンプル・ストア・アーカイブに入っています。サンプル・ストア・アーカイブは、*WC_installdir*/samplestores ディレクトリーのビジネス・モデルによって編成されます。すべてのデータベース資産グループに、ロケール固有の情報が必要なわけではありません。

b. ストアのすべての XML データベース資産ファイルを統合し、 XML エンティティー参照とストア全体のルート・エレメントを含んだ新しい XML ファイル。このファイルをメイン・データベース資産グループ XML ファイルといいます。サンプル・パッケージにはこのファイルが含まれており、store-data-assets.xml という名前になっています。

c. データベース資産グループの XML ファイルが必要とするすべてのデータ・タイプを定義した新しい DTD ファイル。このファイルをメイン・データベース資産グループ *DTD ファイル* といいます。サンプル・パッケージにはこのファイルが含まれており、 store-data-assets.dtd という名前になっています。

d. 外部従属関係を定義した 2 番目の DTD ファイル。場合によっては、このファイルをメイン・データベース資産グループ *DTD ファイル* に組み込む必要があります。サンプル・パッケージにはこのファイルが含まれており、ForeignKeys.dtd という名前になっています。

e. すべての WebSphere WebSphere Commerce テーブルの定義を含んだ 3 番目の DTD ファイル。 wcs.dtd ファイルは WebSphere Commerce にすでに存在しており、*WC_installdir*/schema/xml ディレクトリーにあります。場合によっては、このファイルをメイン・データベース資産グループ *DTD ファイル* に組み込む必要があります。 WebSphere Commerce スキーマをカスタマイズしていない場合は、このファイルをそのまま使えます。

3. 本書のこれまでの章で取り上げた説明に従って、データベース資産 XML ファイルを作成します。資産に関する章で取り上げたタスクを完了していれば、その XML ファイルはすでに存在しています。データベース資産ファイルの先頭には、 DTD 宣言やページ・ディレクティブを含めないでください。ファイルを連結したときに競合が生じる可能性があるからです。ルート・エレメントが必要な唯一のファイルは、メイン・データベース資産グループ XML ファイルです。

   注: 複数の言語のデータベース資産ファイルがある場合は、各ファイルの先頭に、<?xml encoding = *locale specific encoding*> を指定する必要があります。たとえば、英語のデータベース資産ファイルでは <?xml encoding = "UTF-8"?> と指定し、フランス語のファイルでは <?xml encoding = "ISO-8859-1"?> と指定します。

4. ストア・データ全体のメイン・データベース資産グループ XML ファイルを作成します。このファイルには参照エンティティーを指定して、ストアのさまざまなデータベース資産 XML ファイルを組み込むようにします。外部参照エンティティーを使用して XML ファイルを連結すると、 ID Resolve コマンドとロード・プロセスがシンプルになります。さらに、一度に複数のグループをロードするときに、各 XML ファイルで使用されている内部別名は、同じグループ内または他のグループの別の XML データベース資産ファイルに対して外部の名前になります。 XML パーサーは、外部参照を、外部参照エンティティーによって参照されているファイルの内容で置き換えます。

   次に、ストア・データ全体をロードする例を示します。この例を参考にして、データベース資産グループ・ファイルを作成してください。

```
<?xml version="1.0"?>
<!DOCTYPE import SYSTEM "store-data-assets.dtd">
<import>
<!Fulfillment data group -->
```

```
&fulfillment.xml;

<!-- Store data group -->
&store.xml;
&en_US_store.xml;
&fr_FR_store.xml;

<!-- Tax data group -->
&tax.xml;
&en_US_tax.xml;
&fr_FR_tax.xml;
&taxfulfill.xml;

<!-- Shipping data group -->
&shipping.xml;
&en_US_shipping.xml;
&fr_FR_shipping.xml;
&shipfulfill.xml;

<!-- Catalog data group -->
&catalog.xml;
&en_US_catalog.xml;
&fr_FR_catalog.xml;
&storecatalog.xml;
&storefulfill.xml;
&offering.xml;
&store-catalog-tax.xml;
&store-catalog-shipping.xml;

<!-- Currency data group -->
&currency.xml;
&en_US_currency.xml;
&fr_FR_currency.xml;

<!-- Campaign data group -->
&campaign.xml;
&en_US_campaign.xml;
&fr_FR_campaign.xml;

<!-- Business policy data group -->
&businesspolicy.xml;
&en_US_businesspolicy.xml;
&fr_FR_businesspolicy.xml;

<!-- Access control data group -->
&accesscontrol.xml;
&en_US_accesscontrol.xml;
&fr_FR_accesscontrol.xml;

<!-- Other data groups -->
&command.xml;
&store-default.xml;
</import>
```

ここで、

- import は、 XML 文書のルート・エレメントです。ルート要素は、 WebSphere Commerce に用意されている wcs.dtd ファイルですでに定義されています。このルート要素には、 WebSphere Commerce データベース内のすべてのテーブルの定義が含まれています。しかし、WebSphere Commerce スキーマをカスタマイズした場合には、別のルート要素を使用しなければならないかもしれません。カスタマイズしたスキーマに合わせて新しい DTD ファイルを生成することも、既存の wcs.dtd ファイルを更新することもできます。

- store-data-assets.dtd は、次のステップで作成するメイン・データベース資産グループ DTD ファイルの名前です。コメント化されているテキストは、ストアのそれぞれのデータベース資産グループの開始点になっています。
- *&database asset*.xml; は、データベース資産 XML フラグメント・ファイルに対する XML エンティティー参照です。データベース資産のパスと位置は、データベース資産グループ DTD ファイルで定義されています。*&database asset*.xml; ファイルの名前は、各グループにすでに作成されたデータベース資産と一致するように変更されます。
- &*locale_database asset*.xml; は、ストアがサポートしている言語ごとに必要です。ストアで使用する言語が 1 つであれば、 1 つのファイルを参照するだけで十分です。ストアで複数の言語をサポートしていれば、各言語の参照が必要です。上記の抽出部分は、ストアが英語とフランス語をサポートしていることを前提としたものです。

5. 上記のエンティティーを定義したメイン・データベース資産グループ DTD ファイルと、データベース資産が必要とする他の DTD ファイルを作成します。

   次に、ストアのデータベース資産全体の例を示します。この例を参考にして、メイン・データベース資産グループ DTD ファイルを作成してください。

```
<!ENTITY % wcs.dtd SYSTEM "absolute path for WebSphere Commerce wcs.dtd file">
%wcs.dtd;

<!ENTITY % ForeignKeys.dtd SYSTEM "ForeignKeys.dtd">
%ForeignKeys.dtd;
<!ENTITY fulfillment.xml SYSTEM "data/fulfillment.xml">
<!ENTITY en_US_fulfillment.xml SYSTEM "data/en_US/fulfillment.xml">
<!ENTITY fr_FR_fulfillment.xml SYSTEM "data/fr_FR/fulfillment.xml">

<!ENTITY store.xml SYSTEM "data/store.xml">
<!ENTITY en_US_store.xml SYSTEM "data/en_US/store.xml">
<!ENTITY fr_FR_store.xml SYSTEM "data/fr_FR/store.xml">

<!ENTITY tax.xml SYSTEM "data/tax.xml">
<!ENTITY en_US_tax.xml SYSTEM "data/en_US/tax.xml">
<!ENTITY fr_FR_tax.xml SYSTEM "data/fr_FR/tax.xml">
<!ENTITY taxfulfill.xml SYSTEM "data/taxfulfill.xml">

<!ENTITY shipping.xml SYSTEM "data/shipping.xml">
<!ENTITY en_US_shipping.xml SYSTEM "data/en_US/shipping.xml">
<!ENTITY fr_FR_shipping.xml SYSTEM "data/fr_FR/shipping.xml">
<!ENTITY shipfulfill.xml SYSTEM "data/shipfulfill.xml">

<!ENTITY catalog.xml SYSTEM "data/catalog.xml">
<!ENTITY en_US_catalog.xml SYSTEM "data/en_US/catalog.xml">
<!ENTITY fr_FR_catalog.xml SYSTEM "data/fr_FR/catalog.xml">
<!ENTITY store-catalog.xml SYSTEM "data/store-catalog.xml">
<!ENTITY storefulfill.xml SYSTEM "data/storefulfill.xml">
<!ENTITY offering.xml SYSTEM "data/offering.xml">
<!ENTITY store-catalog-tax.xml SYSTEM "data/store-catalog-tax.xml">
<!ENTITY store-catalog-shipping.xml SYSTEM "data/store-catalog-shipping.xml">

<!ENTITY currency.xml SYSTEM "data/currency.xml">
<!ENTITY en_US_currency.xml SYSTEM "data/en_US/currency.xml">
<!ENTITY fr_FR_currency.xml SYSTEM "data/fr_FR/currency.xml">

<!ENTITY campaign.xml SYSTEM "data/campaign.xml">
<!ENTITY en_US_campaign.xml SYSTEM "data/en_US/campaign.xml">
<!ENTITY fr_FR_campaign.xml SYSTEM "data/fr_FR/campaign.xml">
```

```
<!ENTITY businesspolicy.xml SYSTEM "data/businesspolicy.xml">
<!ENTITY en_US_businesspolicy.xml SYSTEM "data/en_US/businesspolicy.xml">
<!ENTITY fr_FR_businesspolicy.xml SYSTEM "data/fr_FR/businesspolicy.xml">

<!ENTITY accesscontrol.xml SYSTEM "data/accesscontrol.xml">
<!ENTITY en_US_accesscontrol.xml SYSTEM "data/en_US/accesscontrol.xml">
<!ENTITY fr_FR_accesscontrol.xml SYSTEM "data/fr_FR/accesscontrol.xml">

<!ENTITY command.xml SYSTEM "data/command.xml">
<!ENTITY store-defaults.xml SYSTEM "data/store-defaults.xml">
```

ここで、

- wcs.dtd は、このデータベース資産グループの外部で定義されているデータを含んだ DTD ファイルです。 WebSphere Commerce に用意されているこのファイルでは、データベース資産グループ XML ファイルで使用されるルート要素も定義されています。
- ForeignKeys.dtd は、ルート要素以外の要素を定義する DTD ファイルです。このファイルには、このデータベース資産グループの外部従属関係のすべての XML エンティティー参照宣言と定義が含まれています。したがって、この XML ファイルには、このデータベース資産グループの一部としては作成されなかったものの、このグループよりも前にデータベースにロードされていなければならない外部キー値に対する参照があります。

  注: パスが正しく指定されていることを確認してください。この例では、メイン・データベース資産グループ DTD ファイルと同じディレクトリーにこのファイルが入っています。
- store.xml、 en_US_store.xml、 fr_FR_store.xml は、ストアで英語とフランス語をサポートしている場合に、メイン・データベース資産グループ XML ファイルで使用される外部参照エンティティーです。この参照を使用するには、 *&alias_name;* という形式でエンティティー参照を記述してください。
- *database asset*.xml は、データベース資産のロード元の XML ファイルの名前です。この名前は、すでに作成されている各グループのデータベース資産ファイルに合わせて変更してください。そのようなファイルの例が、WebSphere Commerce のサンプル・ストア・アーカイブに入っています。サンプル・ストア・アーカイブは、*WC_installdir*/samplestores ディレクトリーのビジネス・モデルによって編成されます。
- *locale_database asset*.xml ファイルは、ストアがサポートしている言語ごとに必要です。このファイルは、上記のディレクトリーにあります。ストアが単一の言語であれば、1 つのファイルだけを参照することになります。ストアが 2 つ以上の言語をサポートしている場合は、各言語ごとにロケール固有のファイルが必要です。上記の抽出部分は、ストアが英語とフランス語をサポートしていることを前提としたものです。

6. それぞれのデータベース資産グループは、外部従属関係を持つ場合があるため、自分のドメイン、つまり自分のデータのセットの外側で定義された情報を必要とします。このデータを DTD ファイルに指定できます。たとえば、ストア・データベース資産グループには、以下のような外部従属関係があります。

   ```
   bootstrap.LANGUAGE.LANGUAGE_ID、bootstrap.MEMBER.MEMBER_ID、
   bootstrap.SETCURR.SETCURR_ID、
   fulfillment.FFMCENTER.FFMCENTER_ID
   ```

1 つのデータベース資産グループまたはストア資産全体をロードする場合には、 WebSphere Commerce データベースからの外部従属関係を定義する必要があります。このデータを使用するには、対応する XML エンティティー参照に従ってください。たとえば、ffmcenter_id エンティティーで定義されているデータを使用するには、 XML ファイルに &ffmcenter_id; と記述します。次に、ストア・データベース資産の例を示します。この例を参考にして、ForeignKeys.dtd という DTD ファイルを作成してください。

```
<!ENTITY en_US "-1">
<!ENTITY fr_FR "-2">
<!ENTITY de_DE "-3">
<!ENTITY it_IT "-4">
<!ENTITY es_ES "-5">
<!ENTITY pt_BR "-6">
<!ENTITY zh_CN "-7">
<!ENTITY zh_TW "-8">
<!ENTITY ko_KR "-9">
<!ENTITY ja_JP "-10">
<!ENTITY MEMBER_ID "-2000">
<!ENTITY ffmcenter_id "10001">
```

ここで、

- MEMBER_ID は、ストアの所有者を識別する内部参照番号です。
- ffmcenter は、ストアの配送センターの参照番号です。ストアでは複数の実行センターを使用できるため、 ForeignKeys.dtd ファイルでも複数の実行センターを定義できます。
- *locale* は、ロケールごとの WebSphere Commerce 参照番号です (国と言語または地域と言語で示されます)。この値は、LANGUAGE データベース・テーブルにあります。

注: 既存のストア・アーカイブを複数のデータベース資産グループに分ける場合には、たとえば別名 @ffmcenter_id などへのすべての参照を、対応するエンティティー参照 &ffmcenter_id; で置き換えるようにしてください。

7. すべてのデータ・ファイルを作成したら、 383 ページの『ID Resolve コマンド』で説明されているように、メイン・データベース資産グループ XML ファイルに対して IDResolve コマンドを実行してデータを解決します。
8. 393 ページの『Load コマンド』で説明されているように、解決済みのデータ・ファイルに対して Load コマンドを実行します。ロード・プロセスを検査するには、以下のログ・ファイルを参照してください。
   - DB2 idresgen.db2.log と massload.db2.log
   - Oracle idresgen.oracle.log と massload.oracle.log

   これらのログ・ファイルは、以下のディレクトリーにあります。
   - AIX Linux Solaris Windows *WC_installdir*/logs
   - 400 *WC_userdir*/instances/*instance_name*/logs
9. Business 444 ページの『ビジネス・アカウント資産の発行』の手順に従って、AccountImport コマンドを実行します。
10. 必要に応じて、 445 ページの『契約資産の発行』の手順に従って、契約を発行します。

11. 447 ページの『WebSphere Commerce Server へのコピーによるストアフロント資産およびストア構成ファイルの発行』のタスクをすべて実行します。

データベース資産グループのロード

1 つのデータベース資産グループの XML データを WebSphere Commerce データベースにロードする手順は、次のとおりです。

1. 以下の情報を確認します。
   a. 489 ページの『付録 B. データの作成』
   b. 491 ページの『付録 C. データベース資産グループ』。 WebSphere Commerce のどの資産ファイルとデータベース・テーブルが影響を受けるかを確認しておく必要があります。
   c. 379 ページの『第 37 章 ストア・データのロードの概要』。ローダー・パッケージの背景情報が記されています。
2. ロード・プロセスを計画し、どのデータベース資産グループをロードするかを決定します。 431 ページの『ストアのロード』の手順でストア・データベース資産全体をロードする場合も、 1 つのデータベース資産グループをロードする場合も、基本的なプロセスは変わりません。次のステップでは、それぞれのロード・プロセスに合わせて、以下のファイルを使用または作成します。
   a. 選択したグループの 1 つ以上のデータベース資産ファイル。たとえば、ストア・データベース・グループ資産をロードする場合は、 `store.xml` ファイルと、ストアがサポートしている各ロケールごとの store.xml ファイルが必要です。そのようなファイルの例が、WebSphere Commerce のサンプル・ストア・アーカイブに入っています。サンプル・ストア・アーカイブは、*WC_installdir*/samplestores ディレクトリーのビジネス・モデルによって編成されます。すべてのデータベース資産グループに、ロケール固有の情報が必要なわけではありません。
   b. すべての XML データベース資産ファイルを統合し、 XML エンティティー参照とデータベース資産のルート・エレメントを含んだ新しい XML ファイル。このファイルをメイン・データベース資産グループ XML ファイルといいます。サンプル・パッケージにはこのファイルが含まれており、store-all-assets.xml という名前になっています。
   c. データベース資産グループの XML ファイルが必要とするすべてのデータ・タイプを定義した新しい DTD ファイル。このファイルをメイン・データベース資産グループ *DTD ファイル* といいます。サンプル・パッケージにはこのファイルが含まれており、 store-all-assets.dtd という名前になっています。
   d. 外部従属関係を定義した 2 番目の DTD ファイル。場合によっては、このファイルをメイン・データベース資産グループ *DTD ファイル* に組み込む必要があります。サンプル・パッケージにはこのファイルが含まれており、Non*database asset group*ForeignKeys.dtd という名前になっています。
   e. すべての WebSphere WebSphere Commerce テーブルの定義を含んだ 3 番目の DTD ファイル。 wcs.dtd ファイルは WebSphere Commerce にすでに存在しており、*WC_installdir*/schema/xml ディレクトリーにあります。 場合によっては、このファイルをメイン・データベース資産グループ *DTD ファイ*

ル に組み込む必要があります。 WebSphere Commerce スキーマをカスタマイズしていない場合は、このファイルをそのまま使えます。

3. 本書のこれまでの章で取り上げた説明に従って、ロードするグループのデータベース資産 XML ファイルを作成します。資産に関する章で取り上げたタスクを完了していれば、その XML ファイルはすでに存在しています。データベース資産ファイルの先頭には、 DTD 宣言やページ・ディレクティブを含めないでください。ファイルを連結したときに競合が生じる可能性があるからです。さらに、簡略化のために、ルート・エレメントを作成しないでおくこともできます。ルート・エレメントが必要な唯一のファイルは、メイン・データベース資産グループ XML ファイルです。

   注: 複数の言語のデータベース資産ファイルがある場合は、各ファイルの先頭に、<?xml encoding = *locale specific encoding*> を指定する必要があります。たとえば、英語のデータベース資産ファイルでは <?xml encoding = "UTF-8"?> と指定し、フランス語のファイルでは <?xml encoding = "ISO-8859-1"?> と指定します。指定するエンコードが、ファイルの実際のエンコードと一致することを確認してください。

4. ロードする各グループのメイン・データベース資産グループ XML ファイルを作成します。このファイルには参照エンティティーを指定して、 1 つ以上のデータベース資産グループのさまざまな XML ファイルを組み込むようにします。外部参照エンティティーを使用して XML ファイルを連結すると、 ID Resolve コマンドとロード・プロセスがシンプルになります。さらに、一度に複数のグループをロードするときに、各 XML ファイルで使用されている内部別名は、同じグループ内または他のグループの別の XML データに対して外部の名前になります。 XML パーサーは、外部参照を、外部参照エンティティーによって参照されているファイルの内容で置き換えます。

   次に、 1 つのストア・データベース資産グループをロードする例を示します。この例を参考にして、データベース資産グループ XML ファイルを作成してください。

```
<?xml version="1.0"?>
<!DOCTYPE import SYSTEM "store-assets.dtd">
<import>
&store.xml;
&en_US_store.xml;
&fr_FR_store.xml;
</import>
```

   ここで、

   - import は、 XML 文書のルート・エレメントです。ルート要素は、 WebSphere Commerce に用意されている wcs.dtd ファイルですでに定義されています。このルート要素には、 WebSphere Commerce データベース内のすべてのテーブルの定義が含まれています。しかし、 WebSphere Commerce スキーマをカスタマイズした場合には、別のルート要素を使用しなければならないかもしれません。カスタマイズしたスキーマに合わせて新しい DTD ファイルを生成することも、 wcs.dtd ファイルを更新することもできます。
   - store-assets.dtd は、次のステップで作成するメイン・データベース資産グループ DTD ファイルの名前です。
   - &store.xml; は、データベース資産グループ XML ファイルに対する XML エンティティー参照です。パスと位置は、データベース資産グループ DTD

ファイルで定義されています。この名前は、すでに作成されている各グループの資産ファイルに合わせて変更してください。

- *locale*_store.xml; は、ストアがサポートしている言語ごとに必要です。ストアで使用する言語が 1 つであれば、 1 つのファイルを参照するだけで十分です。ストアで複数の言語をサポートしていれば、各言語の参照が必要です。上記の抽出部分は、ストアが英語とフランス語をサポートしていることを前提としたものです。

5. 上記のエンティティーを定義したメイン・データベース資産グループ DTD ファイルと、そのグループが必要とする他の DTD ファイルを作成します。

   次に、 1 つのストア・データベース資産グループをロードする例を示します。この例を参考にして、データ・グループのメイン・データベース資産グループ DTD ファイルを作成してください。

```
<!ENTITY % wcs.dtd SYSTEM "absolute path for WebSphere Commerce wcs.dtd file">
%wcs.dtd;

<!ENTITY % ForeignKeys.dtd SYSTEM "ForeignKeys.dtd">
%ForeignKeys.dtd;
<!ENTITY store.xml SYSTEM "store.xml">
<!ENTITY en_US_store.xml SYSTEM "en_US/store.xml">
<!ENTITY fr_FR_store.xml SYSTEM "fr_FR/store.xml">
```

   ここで、

   - wcs.dtd は、このデータベース資産グループの外部で定義されているデータを含んだ DTD ファイルです。 WebSphere Commerce に用意されているこのファイルでは、データベース資産グループ XML ファイルで使用されるルート要素も定義されています。
   - ForeignKeys.dtd は、ルート要素以外の要素を定義する DTD ファイルです。このファイルには、このデータベース資産グループの外部従属関係のすべての XML エンティティー参照宣言と定義が含まれています。したがって、この XML ファイルには、このデータベース資産グループの一部としては作成されなかったものの、このグループよりも前にデータベースにロードされていなければならない外部キー値に対する参照があります。

     注: パスが正しく指定されていることを確認してください。この例では、データベース資産グループ DTD ファイルと同じディレクトリーにこのファイルが入っています。

   - store.xml、 en_US_store.xml、 fr_FR_store.xml は、データベース資産グループ XML ファイルで使用される外部参照エンティティーです。この参照を使用するには、 *&alias_name;* という形式でエンティティー参照を記述してください。
   - store.xml は、データベース資産のロード元のグループのデータ・ファイルです。この名前は、すでに作成されている各グループのデータベース資産ファイルに合わせて変更してください。ロケール固有の XML ファイルは、*WC_installdir*/samplestores ディレクトリーにあります。
   - *path*_store.xml ファイルは、ストアがサポートしている言語ごとに必要です。このファイルは、上記のディレクトリーにあります。ストアが単一の言語であれば、1 つのファイルだけを参照することになります。ストアが 2 つ

以上の言語をサポートしている場合は、各言語ごとにロケール固有のファイルが必要です。上記の抽出部分は、ストアが英語とフランス語をサポートしていることを前提としたものです。

6. それぞれのデータベース資産グループは、外部従属関係を持つ場合があるため、自分のドメイン、つまり自分のデータのセットの外側で定義された情報を必要とします。このデータを DTD ファイルに指定できます。たとえば、ストア・データベース資産グループには、以下のような外部従属関係があります。

```
  bootstrap.LANGUAGE.LANGUAGE_ID、bootstrap.MEMBER.MEMBER_ID、
bootstrap.SETCURR.SETCURR_ID、
fulfillment.FFMCENTER.FFMCENTER_ID
```

1 つのデータ・グループまたはストア・データ全体をロードする場合には、WebSphere Commerce データベースからの外部従属関係を定義する必要があります。このデータを使用するには、対応する XML エンティティー参照に従ってください。たとえば、ffmcenter_id エンティティーで定義されているデータを使用するには、 XML ファイルに &ffmcenter_id; と記述します。次に、ストア・データベース資産グループの例を示します。この例を参考にして、Non*database asset group*ForeignKeys.dtd という DTD ファイルを作成してください。

```
<!ENTITY en_US "-1">
<!ENTITY fr_FR "-2">
<!ENTITY de_DE "-3">
<!ENTITY it_IT "-4">
<!ENTITY es_ES "-5">
<!ENTITY pt_BR "-6">
<!ENTITY zh_CN "-7">
<!ENTITY zh_TW "-8">
<!ENTITY ko_KR "-9">
<!ENTITY ja_JP "-10">
<!ENTITY MEMBER_ID "-2000">
<!ENTITY ffmcenter_id "10001">
```

ここで、

- MEMBER_ID は、ストアの所有者を識別する内部参照番号です。
- ffmcenter は、ストアの配送センターの参照番号です。ストアでは複数の実行センターを使用できるため、 ForeignKeys.dtd ファイルでも複数の実行センターを定義できます。
- *locale* は、ロケールごとの WebSphere Commerce 参照番号です (国と言語または地域と言語で示されます)。この値は、LANGUAGE データベース・テーブルにあります。

注:

a. 既存のストア・アーカイブを複数のデータベース資産グループに分ける場合には、たとえば別名 @ffmcenter_id などへのすべての参照を、対応するエンティティー参照 &ffmcenter_id; で置き換えるようにしてください。

b. すでにデータベースに存在するメンバー ID を参照している場合は、サンプル・ストア・データで使用される内部別名を、&MEMBER_ID; で置き換えることができます。そうでない場合、@member_id を使用して、メンバー ID の解決に必要な XML を含めることができます。

7. すべてのデータ・ファイルを作成したら、 383 ページの『ID Resolve コマンド』で説明されているように、データベース資産グループ XML ファイルに対して IDResolve コマンドを実行してデータを解決します。

8. 393 ページの『Load コマンド』で説明されているように、解決済みのデータ・ファイルに対して Load コマンドを実行します。ロード・プロセスを検査するには、以下のログ・ファイルを参照してください。
   - DB2 idresgen.db2.log と massload.db2.log
   - Oracle idresgen.oracle.log と massload.oracle.log

   これらのログ・ファイルは、以下のディレクトリーにあります。
   - AIX Linux Solaris Windows *WC_installdir*/logs
   - 400 *WC_userdir*/instances/*instance_name*/logs
9. Business 444 ページの『ビジネス・アカウント資産の発行』の手順に従って、AccountImport コマンドを実行します。
10. 必要に応じて、 445 ページの『契約資産の発行』の手順に従って、契約を発行します。
11. 447 ページの『WebSphere Commerce Server へのコピーによるストアフロント資産およびストア構成ファイルの発行』のタスクをすべて実行します。

第 39 章 ビジネス・アカウントと契約の発行

ローダー・パッケージでは、一部のストア・データベース資産 (ビジネス・アカウント、契約) をロードできません。これらのデータベース資産を発行するには、365 ページの『第 36 章 ストア全体の発行』で説明されているストア全体の発行オプションの一部として、管理コンソールを使用するか、コマンド行を使用します。あるいは、該当するコマンドを使用して、ビジネス・アカウントと契約を発行することもできます。それらのコマンドは次のとおりです。

- AccountImport — ストア・アーカイブ内の `businessaccount.xml` ファイルからビジネス・アカウントを作成します。
- ContractImportApprovedVersion — `contract.xml` ファイルから契約を作成します。契約がアクティブな状態でこのコマンドを実行すると、その契約が作成されて展開されます。 `contract.xml` ファイルに複数の契約が含まれている場合も、このコマンドは一度呼び出すだけで十分です。

注: これらのコマンドの詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

ビジネス・アカウント資産は、WebSphere Commerce で提供されている一部のサンプル・ストア・アーカイブに、 XML ファイルの形式で含まれています。しかし、ビジネス・アカウント資産用の XML ファイルを作成するよりも、用意されているツールを使用して、ビジネス・アカウント資産を作成することをお勧めします。用意されているツールを使用して、これらの資産を作成することの詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。サンプル・ストア・アーカイブで提供されている該当する XML ファイルを発行することにした場合や、独自のビジネス・アカウントを作成することにした場合のために、ビジネス・アカウントを発行する方法の説明がこの後のセクションにあります。

注: ビジネス・アカウントや契約の発行に管理コンソールを使わない場合は、ストア資産とカタログ資産を発行してから、ビジネス・アカウントと契約を発行する必要があります。特に、ストア ID、カタログ ID、ストアを所有している組織の ID、契約にかかわる購入組織の ID が必要です。契約の条項や条件で特定のカタログを指定しない場合は、ビジネス・アカウントや契約を発行する前に、カタログを発行する必要はありません。
管理コンソールやコマンド行発行ユーティリティーを使用してこれらの資産を発行する場合、カタログ・オプションを選択しているか、または発行されているカタログがすでにストアにあることを確認してください。該当するコマンドを使用してこれらの資産を発行する場合、上記の資産をデータベースにすでにロードしていることを確認してください。

管理コンソールまたはコマンド行によるビジネス・アカウントと契約の発行

管理コンソールやコマンド行の発行ユーティリティーを使用して、ビジネス・アカウントと契約を発行できます。管理コンソールやコマンド行を使用してビジネス・アカウントと契約を発行するには、それらの資産がストア・アーカイブ形式でパッケージされていなければなりません。ストアフロント資産をストア・アーカイブとしてパッケージする処理については、355 ページの『第 9 部 ストアのパッケージ化』を参照してください。

管理コンソールやコマンド行を使用した資産発行の詳細な手順については、WebSphere Commerce オンライン・ヘルプを参照してください。

コマンドによるビジネス・アカウントと契約の発行

資産をストア・アーカイブとしてパッケージしない場合、以下の該当するコマンドを使用して、ビジネス・アカウントと契約を発行できます。

- AccountImport — ストア・アーカイブ内の `businessaccount.xml` ファイルからビジネス・アカウントを作成します。
- ContractImportApprovedVersion — XML ファイルから、承認済みの契約またはアクティブな契約を WebSphere Commerce Server にインポートします。このコマンドは、契約をインポートする前に、その契約に必要な条項や条件が含まれているかどうか、その契約が有効かどうかを確認します。

ビジネス・アカウント資産の発行

サンプル・ストアから取得したビジネス・アカウント資産を発行する手順は、次のとおりです。

1. `ForeignKeys.dtd` を以下の場所にコピーします。
   - *WC_installdir*/xml/trading/dtd
   - 400 *WC_userdir*/instances/*instance_name*/xml/trading/xml

   ForeignKeys.dtd には、*businessaccount.xml* によって参照されるエンティティー値が含まれます。
2. `businessaccount.xml` を以下の場所にコピーします。
   - *WC_installdir*/xml/trading/xml
   - 400 *WC_userdir*/instances/*instance_name*/xml/trading/xml
3. 管理コンソールをオープンします。管理者としてログインします。
4. ブラウザーで、以下の URL を入力します。
   - https://*hostname*:8002/webapp/wcs/admin/servlet/AccountImport?fileName=businessaccount.xml&URL=*The URL to redirect to upon successful completion*

注: このコマンドの構文とパラメーターの詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

契約資産の発行

サンプル・ストアから取得した契約資産を発行する手順は、次のとおりです。

1. `ForeignKeys.dtd` を以下の場所にコピーします。
   - *WC_installdir*/xml/trading/dtd
   - 400 *WC_userdir*/instances/*instance_name*/xml/trading/xml

   ForeignKeys.dtd には、*businessaccount.xml* によって参照されるエンティティー値が含まれます。
2. `contract.xml` を以下の場所にコピーします。
   - *WC_installdir*/xml/trading/xml
   - 400 *WC_userdir*/instances/*instance_name*/xml/trading/xml
3. 管理コンソールをオープンします。管理者としてログインします。
4. ブラウザーで、以下のように入力します。
   - https://*hostname*:8002/webapp/wcs/admin/servlet/ContractImportApprovedVersion?fileName=contract.xml&xsd=false&URL=ContractDisplay
5. ストアに複数の `contract.xml` ファイル (ロケール固有の契約ファイルなど) がある場合は、各 `contract.xml` ファイルについてステップ 1 から 4 までを繰り返します。

第 40 章 ストアフロント資産とストア構成ファイルの発行

ストアフロント資産、HTML および JSP ファイル、プロパティー・ファイルとリソース・バンドル、およびストア・ページを形成しているイメージやグラフィックスの発行は、ストアの機能的な面を作成するプロセスに含まれます。ストアフロントの発行は、365 ページの『第 36 章 ストア全体の発行』で説明されているストア全体の発行オプションの一環として、管理コンソールまたはコマンド行から行うこともできますし、単純に、WebSphere Commerce Server 上の指定されたロケーションに資産をコピーすることによっても行えます。

サンプル・ストアに含まれている JSP ファイルを発行して、ストア・フローを変更する予定であれば、そのストア・アーカイブに属しているストア構成ファイルも発行する必要があります。ストアの XML 構成ファイルは、ストア・アーカイブ内の以下のディレクトリーにあります。

WEB-INF/xml/tools/stores/StoreDirectory/devtools/flow

「フローの変更」ツール用のプロパティー・ファイルは、ストア・アーカイブ内の以下のディレクトリーにあります。

StoreDirectory/devtools/flow/ui

管理コンソールやコマンド行を使用した、ストアフロント資産およびストア構成ファイルの発行

ストアフロント資産とストア構成ファイルの発行は、管理コンソールを使用して、あるいはコマンド行から発行ユーティリティーを使用して行うことができます。

管理コンソールやコマンド行を使用してストアフロント資産とストア構成ファイルを発行するためには、ストアフロント資産とストア構成ファイルがストア・アーカイブ・フォーマットにパッケージされていなければなりません。ストアフロント資産をストア・アーカイブとしてパッケージする処理については、355 ページの『第 9 部 ストアのパッケージ化』を参照してください。

WebSphere Commerce Server へのコピーによるストアフロント資産およびストア構成ファイルの発行

資産をストア・アーカイブにパッケージしないほうが好ましい場合は、ストアフロント資産を直接 WebSphere Commerce Server にコピーする方法でも、資産を発行できます。 Web 資産 (HTML、JSP ファイル、イメージ、グラフィックス) は、 Web アプリケーション文書のルートにコピーしてください。リソース・バンドルやプロパティー・ファイルは、アプリケーションがプロパティーを参照するパスにコピーしてください。ストア構成ファイルは、ストアまたはツール Web モジュールの下の適切な場所にコピーする必要があります。

注: サンプル・ストア・アーカイブを (パス構造を保存しながら) ストア Web モジュールに unzip すると、ファイル資産は正しい場所に置かれます。ただし、「フローの変更」ツールによって使用されるプロパティー・ファイル (StoreDirectory/devtools/flow/ui/*.properties) を、 WebSphere Commerce 構成ファイル `instance_name.xml` で定義された ToolsStoresPropertiesPath にコピーする必要があります。

ストアフロント資産とストア構成ファイルを WebSphere Commerce Server にコピーするには、次のようにします。

1. JSP ファイル、HTML、組み込みファイル、イメージ、グラフィックスを、ストア Web アプリケーション文書のルートにある以下のストア・ディレクトリー (*storedir*) にコピーしてください。
   - *WAS_installdir*/installedApps/*cell_name*/*WC_instance_name*.ear/Stores.war/*storedir*
   - 400 *WAS_userdir*/*WAS_instance_name*/installedApps/*cell_name*/WC_*instance_name*.ear/Stores.war/*storedir*

   ここで、*storedir* は、 STORE データベース・テーブルの DIRECTORY 列の値です。
2. リソース・バンドルとプロパティー・ファイルを、アプリケーションがプロパティーを参照する以下のパスにコピーしてください。
   - *WAS_installdir*/installedApps/*cell_name*/WC_*instance_name*.ear/Stores.war/WEB-INF/classes/*storedir*
   - 400 *WAS_userdir*/*WAS_instance_name*/InstalledApps/*cell_name*/*WC_instance_name*.ear/Stores.war/WEB-INF/classes/*storedir*
3. ストア構成ファイルを、WebSphere Commerce 構成ファイル `instance_name.xml` で定義されている場所にコピーしてください。このファイルは、以下のディレクトリーにあります。
   - *WC_installdir*/instances/*instance_name*/xml
   - 400 *WC_userdir*/instances/*instance_name*/xml

   ストア構成ファイルは、以下の場所にコピーされます。
   - ストア構成 XML (WEB-INF¥xml¥tools¥stores¥storedirectory¥devtools¥flow) は、ToolsStoresXMLPath にコピーされます。このパスは、 WebSphere Commerce 構成ファイル `instance_name.xml` で定義されています。
   - ストア構成プロパティー・ファイルは、 ToolsStoresPropertiesPath にコピーされます。このパスは、WebSphere Commerce 構成ファイル `instance_name.xml` で定義されています。
4. 以下のいずれかの方法でストアを立ち上げます。
   - StoreCatalogDisplay コマンド (`StoreCatalogDisplay?storeId=`*`storeId`*`&catalogId=`*`catalogId`*`&langId=`*`langId`*) を使用します。

     ここで、

- `storeId` は、STORE データベース・テーブルの STORE_ID 列の値です。
- `catalogId` は、CATALOG データベース・テーブルの CATALOG_ID 列の値です。
- `langId` は、LANGUAGE データベース・テーブルの該当するロケールの LANGUAGE_ID 列の値です。 WebSphere Commerce のデフォルト値のリストについては、 LANGUAGE データベース・テーブルを参照してください。

- WebSphere Commerce サンプル・ストアをベースにしたストアの場合は、`index.jsp` ファイルを編集して、ストアの URL を組み立てます。そのファイルは、以下のディレクトリーにあります。
  - *WAS_installdir*/installedApps/*instance_name* /WC_*instance_name*.ear/Stores.war/*storedir*
  - 400 *WAS_userdir*/*WAS_instance_name*/ installedApps/*cell_name*/WC_*instance_name*.ear/Stores.war/*storedir*

  以下のパラメーターに正しい値を指定します。
  - `hostname` は、 WebSphere Commerce マシンの完全修飾名です。
  - `storeId` は、STORE データベース・テーブルの STORE_ID 列の値です。
  - `catalogId` は、CATALOG データベース・テーブルの CATALOG_ID 列の値です。
  - `langId` は、LANGUAGE データベース・テーブルの該当するロケールの LANGUAGE_ID 列の値です。 WebSphere Commerce のデフォルト値のリストについては、 LANGUAGE データベース・テーブルを参照してください。

  ブラウザーでストアを表示するには、 `http://`*`host name`*`/webapp/wcs/stores/servlet/`*`storedir`*`/index.jsp` を指定します。

第 11 部 ストアへの WebSphere Commerce フィーチャーの追加

WebSphere Commerce に用意されている一部のフィーチャーを追加するには、いくつかの手作業が必要になります。ここに含まれる章では、ストアに以下のフィーチャーを追加する方法について説明します。

第 41 章 ストアへのカスタマー・ケアの追加

Professional Business WebSphere Commerce のカスタマー・ケア機能は、 Lotus® Sametime™ サーバーを使用して、同期テキスト・インターフェースにより、リアルタイムの顧客サービス・サポートを提供します。ストア内でカスタマー・ケアが使用可能になっている場合、顧客はストアに入り、リンクをクリックして、顧客サービス担当者 (CSR) に接続します。そして、顧客はインターネットを介して CSR と通信を行うことができます。

注: この章では、ストア内でカスタマー・ケアを使用可能にする方法について説明します。しかし、ストア内でカスタマー・ケアを使用可能にする前に、最初に Sametime サーバーをインストールし、それが WebSphere Commerce で作動するように構成する必要があります。詳細については、「*WebSphere Commerce 追加ソフトウェア・ガイド*」を参照してください。 Sametime サーバーが WebSphere Commerce と同じ LDAP サーバーを使用しない場合、管理コンソールで CSR を登録し、 CSR を使用可能にして、カスタマー・ケアを使用できるようにする必要もあります。カスタマー・ケアの総合的な概念および CSR がカスタマー・ケアを使用する方法、さらにこの作業の詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

注:

次のサンプル・ストアのいずれかに基づいてストアを作成する場合、WebSphere Commerce アクセラレーターを使用して迅速かつ容易にストア内でカスタマー・ケアを使用可能にすることができます。

- Business B2B 向け (ToolTech)
- 消費者向け (FashionFlow)

管理コンソールを使用してストアを発行した後、WebSphere Commerce アクセラレーターを使用して、**「ストア」**メニューを選択してから、 **「フローの変更」**を選択し、カスタマー・ケア機能を使用可能にします。詳しい説明については、WebSphere Commerce オンライン・ヘルプのヘルプ・トピック『WebSphere Commerce Accelerator を使用したストア・フローの変更』を参照してください。

しかし、サンプル・ストアに基づいてストアを作成しない場合は、ストア内でカスタマー・ケアを使用可能にするために、いくらかの作業を行う必要があります。この章の残りの部分では、サンプル・ストアのいずれも使用せずにストア内でカスタマー・ケアを使用可能にするための必要な概念およびステップについて説明します。

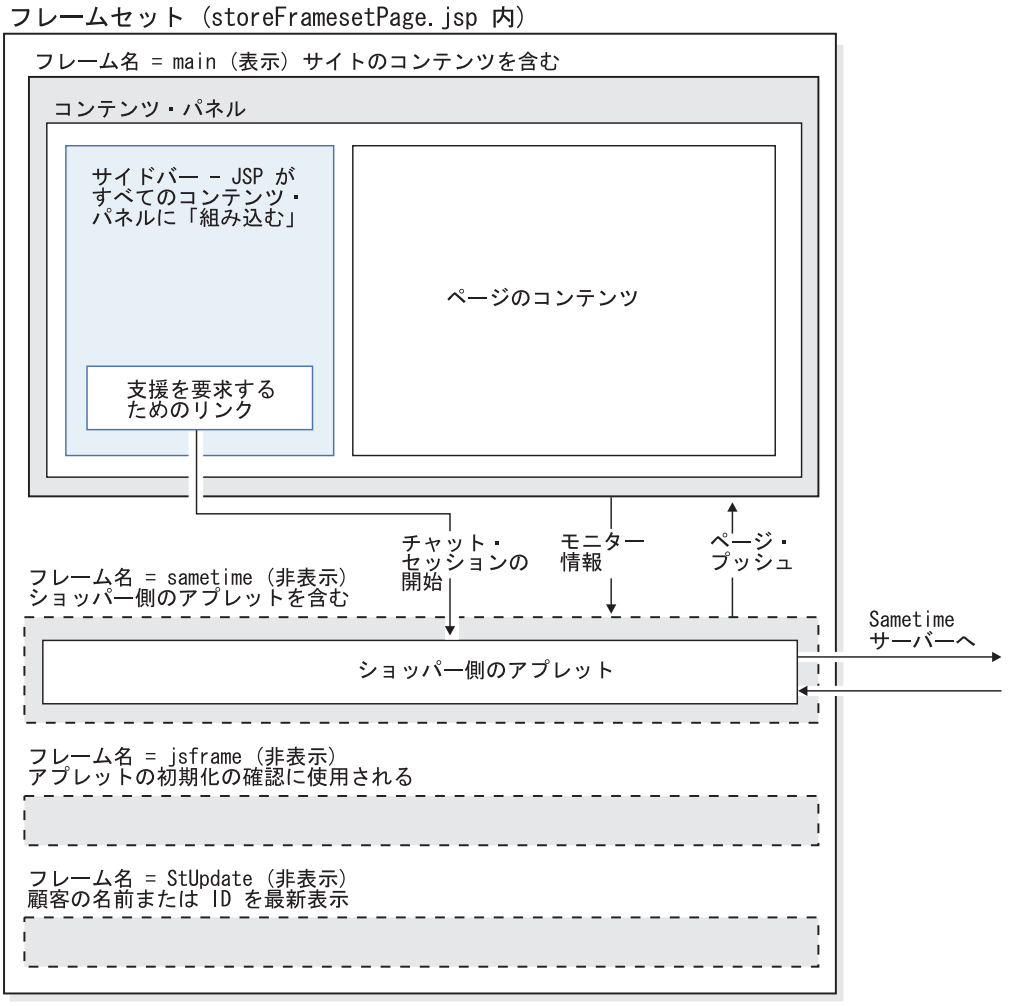
注: サンプル・ストア Business ToolTech および FashionFlow は、カスタマー・ケアをインプリメントする方法を説明し、カスタマー・ケアを使用可能にするためにストア内で使用できるコードを示します。この章では、2 つのストアの例を参照して、ストア内でカスタマー・ケアを使用可能にする方法を説明します。この章をお読みになる際には、必ずサンプル・ストアの最新版をご使用ください。

注: Internet Explorer および Netscape ブラウザーのバージョン 4.x の後方互換性および組み込み JVM をサポートするために、アプレット・コードは JDK 1.1 および AWT コンポーネントを使用して開発されます。このため、JDK 1.2 またはそれ以降のバージョンで使用可能な機能 (言語サポート、BI-direction サポート、およびアクセス可能性を含む) の中には、使用できないものと完全にサポートされるものがあります。 Sun Java プラグインおよび Netscape ブラウザー 6 および 7 はサポートされません。

ストア内のカスタマー・ケアについて

カスタマー・ケアが使用可能になっているストア内で顧客がカスタマー・ケア・リンク (たとえば、**「顧客アシスタントとのライブ・チャット (Live Chat with Customer Assistant)」**) を選択する場合、チャット・ウィンドウを含むアプレットが立ち上がります。このアプレットは隠れたフレーム・セット内で実行されるので、サイトの外観に影響を与えることはありません。

以下の図は、フレーム・セットの構成を示しています。



フレーム・セットには以下の 4 つのフレームがあります。

- メイン: ストアのコンテンツが入っているフレーム (ストア・ページを作成するファイルを含む)。コンテンツとしては、ページの本体、ヘッダーおよびフッター・

ファイル、サイドバー・ファイルを作成するファイルがあります。このフレームのコンテンツはストアのビジターが見ることができます。メインフレームには Sametime フレームへの接続 (カスタマー・ケアおよびモニターの情報) が含まれることに注意してください。モニター情報については、466 ページの『カスタマー・ケアを使用した顧客のモニター』で詳しく説明します。

- Sametime: カスタマー・ケア・アプレットが入っているフレーム。このフレームはストアのビジターには見えません。しかし、顧客がリンクをクリックしてアプレットを立ち上げると、カスタマー・ケア・ウィンドウが表示されます。また、このフレームは、ページ・プッシュ機能を使用して、情報をメインフレームにプッシュします。
- jsframe: アプレットが正しくロードされていることを確認するフレーム。このフレームのコンテンツは顧客には表示されません。
- StUpdate: 顧客の名前または ID を含む、顧客情報を最新表示するフレーム。

フレーム・セットの使用

フレーム・セット内でカスタマー・ケア・アプレットを立ち上げると、アプレット・コードがストア・ページ内のコードから分離します。上の図が示すように、ストア・ページはフレーム・セットのメインフレームに含まれており、アプレット・コードは Sametime フレームに含まれています。アプレット・コードをストア・ページから分離することにより、アプレットが一度だけダウンロードされるため、ネットワーク・トラフィックが減少します。

フレーム・セットを使用すると、Sametime サーバーとの接続を保守することもできます。アプレットがフレーム・セットではなく各ページの一部になっている場合、顧客が新規のページにアクセスするたびに新規の Sametime セッションが作成されます。カスタマー・ケア・アプレットは Sametime サーバーに匿名でログオンするため、顧客が新規のページにアクセスするたびに新規セッションを作成すると、ストアを用いた顧客の活動をトレースできません。フレーム・セットを使用すると、顧客の元の Sametime セッションが保守され、属性が変更されると顧客の活動が Sametime サーバーに返送されます。

フレーム・セットの使用について

カスタマー・ケアをストア内にインプリメントするにはフレーム・セットを使用することが勧められていますが、フレーム・セットの使用については以下の点を理解しておく必要があります。

- 単一のエントリー・ポイント: 顧客がフレームワーク内のストアをブラウズする場合、カスタマー・ケアの使用だけを行えます。同様に、CSR はフレーム・セットを介して顧客の動向のモニターだけを行えます。顧客がフレーム・セットを介してストアをブラウズしていることを確認するには、単一のエントリー・ポイントを介してサイトにアクセスする必要があります。顧客が別のページ (たとえば、カタログ・ページ) を介してストアにアクセスする場合、その顧客はフレーム・セット内にはいません。
- ブックマーク: フレーム・セットを使用する場合、顧客は個々のページではなく、フレーム・セットの URL にのみブックマークを付けることができます。
- 最新表示: 顧客がフレーム・セット内におり、「最新表示」をクリックすると、フレーム・セットでコード化されているとおり元のページに戻りますが、それはブラウザーに現在表示されているページと異なる場合があります。

- ブラウザー・ウィンドウのサイズ変更: 顧客がフレーム・セット内にいるときにブラウザー・ウィンドウのサイズを変更すると、ブラウザーは自動的にエントリー・アドレスを再ロードします。エントリー・アドレスが再ロードされる場合、Sametime サーバーへの接続が終了することがあります。この状況では、ブラウザーによって動作が異なります。
- セキュリティー: 正常に動作させるには、カスタマー・ケア・フレーム・セット内のページが互いに通信できるようにする必要があります。この通信は、JavaScript 関数呼び出しによって使用可能になります。顧客がフレーム・セットを介してサイトをブラウズする場合、フレーム・セット (ロケーション・バーの URL) と同様に個々のフレームは独自の接続を保守します。この接続は、非セキュア (http。デフォルトではポート 80) の場合と、セキュア (https。デフォルトではポート 443) の場合があります。顧客が非セキュア接続を介してストアをブラウズしている場合、フレーム・セット内のすべてのフレームは HTTP にあります。このシナリオでは、SSL については扱っていません。
  しかし、顧客がセキュア・ページ (たとえば、登録ページ) をブラウズする場合、フレーム・セット内の main フレームは HTTPS に切り替わりますが、それ以外のフレームは非セキュア (http) のままです。この状況では、顧客はカスタマー・ケア・アプレットを立ち上げることができません。ブラウザーにはアプレットを立ち上げる許可が与えられていません。なぜなら、アプレット (セキュア、ポート 443) を呼び出す JavaScript 関数は、ブラウザーのロケーション・バーの URL (HTTP、ポート 80) とは異なるサーバーから立ち上げられているように見えるからです。この問題を解決するには、ストアに入る際には常に StoreFramesetView コマンドを使用して、フレーム・セット URL 用の HTTPS 接続が使用されるようにする必要があります。
- フレーム同士で通信するには、Java アプレットおよび JavaScript 機能も通信していなければなりません。アプレット・コード・ベースは WebSphere Commerce Server ではなく、Sametime サーバーにあるため、Sun Java プラグインを使用する一部のブラウザー (例、Netscape バージョン 6 および 7) は、JavaScript が、異なるホストからロードされた Java アプレットと通信するのを妨げます。

フレーム・セットなしでカスタマー・ケアを使用する

カスタマー・ケアはフレーム・セットを使用しなくても作動しますが、CSR は顧客のアクティビティーをモニターすることはできません。つまり、顧客から集められた情報が正確であるのは、顧客がヘルプ要求を送信した時のみです。その後に顧客がページを変更したり、ショッピング・カートにさらにアイテムを追加した場合、新規情報は CSR 用にモニター・リストで更新されません。

フレーム・セットなしでカスタマー・ケアを使用するには、以下のようにします。

1. チャット・ページを起動するリンクを含む JSP ファイルに以下のコードを追加します。

```
<script>
function LaunchChat()
{
<%
String pname = request.getRequestURI();
int indpn = pname.lastIndexOf('/');
indpn = pname.lastIndexOf('/', indpn-1);
if(indpn != -1) { pname = pname.substring(indpn+1);}
String headerType = (String) request.getAttribute("liveHelpPageType");
if (headerType==null) { headerType="";}
```

```
if (headerType.equals("personal"))
{
%>
 currentPageURL='PERSONAL_URL';
<% } else { %>
 currentPageURL=escape(location.href);
<% } %>
 currentpageDesc="<%=pname%>";
 chatURL="<%=com.ibm.commerce.tools.util.UIUtil.getWebappPath(request)%>
CCChatPageView?"
  + "pageURL=" + currentPageURL
  + "&pageDesc=" + currentpageDesc;
 WindowName="";
 chatAttr="toolbars=no,location=no,directories=no,status=yes,
menubar=no,scrollbars=no,resizable=no,width=360,height=400"
 window.open(chatURL,WindowName, chatAttr);
 return true;
}
</script>
```

注: 幅および高さが CustomerCareChatSetup.jsp の APPLET タグで定義されているものと同じであることを確認してください。

2. 上のコードを追加するページがパーソナル・ページである場合、以下のステートメントを使用して、このコード・ブロックの前にパーソナル・ページを定義します。 `request.setAttribute("liveHelpPageType", "personal");`
3. 追加パラメーター値の組を、カスタマイズした属性または他の役立つ情報用に chatURL ストリングに追加することもできます。これは、要求オブジェクトを介して CustomerCareChatSetup.jsp に取り上げられます。
4. 以下のコードを追加して、リンクまたはイメージからチャット・ページを起動します。

```
<A HREF="javascript:void(LaunchChat());">
<FONT COLOR="#ffffff" STYLE="font-size : 8pt">
<%= tooltechtext.getString("LiveHelp")%></FONT></A>
```

5. (オプション) CustomerCareChatSetup.jsp ファイルをカスタマイズして追加カスタマー情報に渡します。以下のアプレット・パラメーターを使用して、CustomerCareChatSetup.jsp をカスタマイズすることができます。

表 16.

カスタマイズ可能なパラメーター名	説明	注
CHAT_FONT_SIZE	チャット領域で使用されるフォント・サイズ。	デフォルト値は 12 です。
CHAT_FONT_COLOR	チャット領域で着信メッセージ用に使用されるフォント・カラー。	デフォルトは青 (#0000FF) です。
CHAT_NAME_LENGTH	チャット領域でユーザー名を表示するのに予約済みの文字の長さ。	デフォルトは 15 です。

表 16. (続き)

WAIT_RANGE_1	整数値であり、ストアで待機中の顧客の数がこの値より小さい場合、警告メッセージ 1 が表示されます。そうでない場合、WAIT_RANGE_2 設定に応じて警告メッセージが表示されます。	メッセージ 1 のみが表示される場合は、-1 を使用します。
WAIT_RANGE_2	整数値であり、ストアで待機中の顧客の数がこの値より小さく、WAIT_RANGE_1 より大きい場合、警告メッセージ 2 が表示されます。そうでない場合、WAIT_RANGE_3 設定に応じて警告メッセージが表示されます。	WAIT_RANGE_1 が -1 である場合は無視されます。この範囲を使用不可にするには -1 を使用します。
WAIT_RANGE_3	整数値であり、ストアで待機中の顧客の数がこの値より小さく、WAIT_RANGE_2 より大きい場合、警告メッセージ 3 が表示されます。そうでない場合、警告メッセージ 4 が表示されます。	WAIT_RANGE_2 が -1 である場合は無視されます。この範囲を使用不可にするには -1 を使用します。
contentFrame	通常の WebSphere Commercestore ページに使用されるフレーム名。	デフォルトは ″_blank″ です。CSR が URL を戻すと、新規ブラウザー・ウィンドウが常に起動します。
COUNTER_UNIT_WAIT	整数値であり、どれくらいの頻度で待機カウンターが 1 ずつ増加するかを示します。	デフォルトは 30 秒です。ストアの CustomerCareMonitorList.jsp ファイルで定義されているものと同じであることを確認してください。
WIDTH	希望するチャット・フレームの幅 (ピクセル)。	デフォルトの幅は 360 ピクセルです。招待メッセージの長さは、フィニアルの幅に影響します。
HEIGHT	チャット・フレームの高さ (ピクセル)。	デフォルトは 400 ピクセルです。
QUEUE_ID	カスタマー・ケア・キュー ID	有効なキュー ID を提供することによって、ヘルプ要求がキューに直接書き込まれます。
ML_ATTRIBUTES	コンマで区切られた、カスタマイズしたモニター属性 ID のリスト。	例 : <PARAM name=″ML_ATTRIBUTES″ value=″8001,9002″>

表 16. (続き)

ML_ATTRIBUTE_xxxx	カスタマイズしたモニター属性 xxxx (xxxxx は属性 ID) の値を提供します。	例: <PARAM name="ML_ATTRIBUTE_8001" value="A value string "> 属性 ID が ML_ATTRIBUTES パラメーターですでに定義されていることを確認します。そうでない場合、このパラメーターは無視されます。

カスタマー・ケアの定義

ストア内のカスタマー・ケアを使用する前に、以下の情報を定義する必要があります。

- ストアのモニター・リストの定義
- ストアのトピック・リストの定義
- ストアの URL の定義

以下のセクションでは、これらの概念を、情報を定義する方法を説明するサンプル・ストアからの例を使用して説明します。

ストアのモニター・リストの定義

各ストアには、CSR によってモニターされるアイテムを定義するファイルが含まれます。ファイル `CustomerCareMonitorList.jsp` は以下のディレクトリーにあります。

- *WAS_installdir*/installedApps/*instance_name*/WC_*instance_name*.ear/Stores.war/*storedir*/CustomerServiceArea/CollaborationSection
- > 400 *WAS_userdir*/WAS_*instance_name*/installedApps/*cell_name*/WC_*instance_name*.ear/Stores.war/*storedir*/CustomerServiceArea/CollaborationSection

`CustomerCareMonitorList.jsp` は、WebSphere Commerce アクセラレーターが CCMonitorListView コマンドを呼び出すと、CSR アプレットにロードされます。このコマンドは、ストア用にモニター情報を定義する、以下の XML ストリングを戻す JSP ファイルを戻します。

```
<?xml version="1.0" encoding="UTF-8"?>
<WCS_CUSTOMERCARE>
      <MONITORING_LIST>
         <ATTR ID="4" LABEL="MonitoringVisitorsTableCurrentPage" ></ATTR>
         <ATTR ID="3010" LABEL="MonitoringVisitorsTableInSite" UNIT="30" ></ATTR>
         <ATTR ID="3011" LABEL="MonitoringVisitorsTableInPage" UNIT="30" ></ATTR>
         <ATTR ID="6" LABEL="MonitoringVisitorsTableCart" ></ATTR>
      </MONITORING_LIST>
</WCS_CUSTOMERCARE>
```

属性エレメント (ATTR) は、上で説明されているように、顧客のモニター属性を定義します。

表 17.

属性名	説明	注
ID	このアイテムをトラッキングするのに使用する Sametime 属性 ID。	必須
Label	プロパティー・ファイルにあるこのアイテムの説明用のラベル・キー。対応する説明は CSR アプレットに表示されます。	必須
Unit	カウンターが何秒後に 1 ずつ増えるかを示す整数値。	オプションです。事前定義カウンター属性によってのみ使用されます。

CSR が CSR アプレットを起動すると、アプレットは XML ストリングをロードして、モニター・リストを作成します。モニター・リストには、対応する Sametime 属性に関連したストリング値が表示されます。必要に応じて、JavaScript 関数は、適切なストア・ページから属性値を検索します。

注: XML ストリングにエラーが含まれる場合、または CustomerCareMonitorList.jsp ファイルを配置できない場合、CSR アプレットは、顧客の名前のみが含まれるデフォルトのモニター・リストを使用します。

例: CustomerCareMonitorList.jsp: 以下のコード例は、B2B 向けのサンプル・ストア、ToolTech からのものです。

```
<%
// Create XML string
String strCfg=ECLivehelpConstants.EC_CC_XML_HEADER
      + LiveHelpConfiguration.getOpenTagString(ECLivehelpConstants.EC_CC_XML_ROOT)
     + LiveHelpConfiguration.getOpenTagString(ECLivehelpConstants.
EC_CC_XML_MONITORING_LIST);
%>

<%  // modify this block to customize monitoring list
      strCfg=strCfg + LiveHelpConfiguration.getMonitorAttributeElementString
(ECLivehelpConstants.EC_CC_ST_ATTR_PAGE_URL, "MonitoringVisitorsTableCurrentPage")
 + LiveHelpConfiguration.getCloseTagString(ECLivehelpConstants.
EC_CC_XML_MONITORING_ATTR);
      strCfg=strCfg + LiveHelpConfiguration.getMonitorCounterAttributeElementString
(ECLivehelpConstants.EC_CC_ST_ATTR_SITE_COUNTER,
"MonitoringVisitorsTableInSite","30")
 + LiveHelpConfiguration.getCloseTagString(ECLivehelpConstants.
EC_CC_XML_MONITORING_ATTR);
     strCfg=strCfg + LiveHelpConfiguration.getMonitorCounterAttribute
ElementString
(ECLivehelpConstants.EC_CC_ST_ATTR_PAGE_COUNTER,
 "MonitoringVisitorsTableInPage", "30")
 + LiveHelpConfiguration.getCloseTagString(ECLivehelpConstants.
EC_CC_XML_MONITORING_ATTR);
     strCfg=strCfg  + LiveHelpConfiguration.getMonitorAttributeElementString
(ECLivehelpConstants.EC_CC_ST_ATTR_CART_ITEMS,
"MonitoringVisitorsTableCart")
 + LiveHelpConfiguration.getCloseTagString
(ECLivehelpConstants.EC_CC_XML_MONITORING_ATTR);
%>

<%
```

```
strCfg=strCfg
 + LiveHelpConfiguration.getCloseTagString
(ECLivehelpConstants.EC_CC_XML_MONITORING_LIST)
 + LiveHelpConfiguration.getCloseTagString
(ECLivehelpConstants.EC_CC_XML_ROOT);
%>
```

サンプルは、LiveHelpConfiguration クラスのユーティリティー・メソッドを使用して、XML の正確さを確認します。問題を構文解析するために、サンプルもエンコードされた属性値を使用します。以下のテーブルでは、このメソッドに関して詳細に説明します。

表 18.

メソッド	説明	Notes™
静的ストリング getOpenTagString(String tagName)	オープン・タグ・ストリングを戻す	LiveHelpConfiguration. getOpenTagString(″HELLO″) はストリング <HELLO> を戻す
静的ストリング getCloseTagString(String tagName)	クローズ・タグ・ストリングを戻す	LiveHelpConfiguration. getCloseTagString (″HELLO″) はストリング </HELLO> を戻す
静的ストリング getMonitorAttribute ElementString(String sAttributeId、 String sLabelKey)	モニター・リストの ATTR エレメント・ストリングを戻す	LiveHelpConfiguration. getMonitorAttribute ElementString(″1234″、″myLabel″) はストリング <ATTR ID=″1234″ LABEL=″myLabel″ > を戻す
静的ストリング getMonitorCounter AttributeElementString (String sAttributeId、String sLabelKey、String sUnit)	モニター・リストのカウンター ATTR エレメント・ストリングを戻す	LiveHelpConfiguration .getMonitorCounter AttributeElementString (″1234″、″myLabel″、″60″) はストリング <ATTR ID=″1234″ LABEL=″myLabel″ UNIT=″60″ を戻す >

以下のテーブルは、モニター可能な事前定義された Sametime 属性 ID をリストします。

表 19.

属性 ID	説明	ラベル・キー・ストリング	Notes
ECLivehelpConstants. EC_CC_ST_ATTR_ PAGE_URL	ショッパーがブラウズする Web ページ	″MonitoringVisitors TableCurrentPage″	ストア・ページからの入力が必要
ECLivehelpConstants. EC_CC_ST_ATTR_ CART_ITEMS	ショッピング・カートに追加されたアイテムの数	″MonitoringVisitors TableCart″	ストア・ページからの入力が必要

表 19. (続き)

ECLivehelpConstants. EC_CC_ST_A TTR_SITE_COUNTER	顧客がストア・サイトにいる時間	″Monitoring VisitorsTable InSite″	カウンター属性であり、UNIT を使用して更新頻度を設定する
ECLivehelpConstants .EC_CC_ST_ATTR _PAGE_COUNTER	顧客がこのページにいる時間	″Monitoring VisitorsTable InPage″	カウンター属性であり、UNIT を使用して更新頻度を設定し、カウンターをリセットするにはストア・ページからの入力が必要
ECLivehelpConstants. EC_CC_ST_ATTR _WAIT_COUNTER	顧客が CSR を待機している時間	″Monitoring Visitors TableInWait″	カウンター属性であり、UNIT を使用して更新頻度を設定します。 顧客がヘルプ要求を送信するか、顧客がチャット中に CSR によってリキューされるとリセットされます。 これは、CSR が「次の顧客を担当」ボタンを使用した場合、CSR がどの顧客を最初にサービスを提供するかを判別するのにも使用されます。
ECLivehelpConstants. EC_CC_ST _ATTR_ REQ_QUEUE	顧客の要求が含まれているキューの名前	″Monitoring VisitorsTable Queue	初期キュー名であり、顧客がストア・ページでヘルプ要求を送信すると提供される
ECLivehelpConstants. EC_CC_ ST_ATTR _REQ_CSR_NAME	顧客とチャットしている CSR の名前	″Monitoring Visitors TableCSR″	CSR がキューから要求を受け入れると更新される

注: 8000 より大きい属性 ID を使用して、カスタマイズしたモニター・アイテムを定義できます。以下は、その例です。

```
<ATTR ID="8004" LABEL="MonitoringVisitorsTableItem8004"></ATTR>
```

ストアのトピック・リストの定義

各ストアには、CSR が顧客とのチャット・セッション中に使用可能なトピックを定義するファイルが含まれます。ファイル `CustomerCareStoreQuestionList.jsp` は以下のディレクトリーにあります。

- *WAS_installdir*/installedApps/*instance_name*/WC_*instance_name*.ear/Stores.war/*storedir*/CustomerServiceArea//CollaborationSection
- > 400 *WAS_userdir*/WAS_*instance_name*/installedApps/*cell_name*/WC_*instance_name*.ear/Stores.war/*storedir*/CustomerServiceArea/CollaborationSection

この JSP ファイルは、ストア用にチャット・トピックを定義する、以下の XML ストリングを戻します。

```
<?xml version="1.0" encoding="UTF-8"?>
<WCS_CUSTOMERCARE>
     <QUESTION_LIST>
          <GROUP NAME="TopicGroupName" >
         <      <QUESTION TITLE="TopicName" TEXT="Topic+Text" ></QUESTION>
         /GROUP>
    </QUESTION_LIST>
</WCS_CUSTOMERCARE>
```

グループ・エレメント (GROUP) は、上で説明されているように、複数のサブトピックがあるトピック・グループを定義します。これには以下の属性があります。

表 20.

属性名	説明	注
名前	グループ名が重複する場合、最新のグループ定義が使用されます。	必須

質問エレメント (QUESTION) は単一トピックを定義します。これには以下の属性があります。

表 21.

属性名	説明	注
Title	表題が重複する場合、最新の表題定義が使用されます。	必須
Text	トピックの内容。	必須

例 : CustomerCareStoreQuestionList.jsp: 以下のコードは、B2B 向けのサンプル・ストア、ToolTech からのものです。 CustomerCareStoreQuestionList.jsp:

```
<% // Create XML string
String strCfg=ECLivehelpConstants.EC_CC_XML_HEADER
    + LiveHelpConfiguration.getOpenTagString(ECLivehelpConstants.EC_CC_XML_ROOT)
    + LiveHelpConfiguration.getOpenTagString(ECLivehelpConstants.
EC_CC_XML_QUESTION_LIST);
%>
<% //unmark this block to add Topic group/topics
 // start of Topic group block, repeat for more topic groups
 strCfg=strCfg + LiveHelpConfiguration.getTopicGroupElementString
("TopicGroupName");
  // start of Topic block, repeat for all topics in the same
group
  strCfg=strCfg + LiveHelpConfiguration.getTopicElementString("TopicName","
Topic Text")
           + LiveHelpConfiguration.getCloseTagString(ECLivehelpConstants.
EC_CC_XML_QUESTION_QUESTION);
```

```
  // end of Topic block
 strCfg=strCfg + LiveHelpConfiguration.getCloseTagString
(ECLivehelpConstants.EC_CC_XML_QUESTION_GROUP);
 // end of Topic group block
%>
<%
strCfg=strCfg  + LiveHelpConfiguration.getCloseTagString
(ECLivehelpConstants.EC_CC_XML_QUESTION_LIST)
         + LiveHelpConfiguration.getCloseTagString
(ECLivehelpConstants.EC_CC_XML_ROOT);
%>
```

サンプルは、LiveHelpConfiguration クラスのユーティリティー・メソッドを使用して、XML の正確さを確認します。以下のテーブルでは、このメソッドに関して詳細に説明します。

注:

1. 問題を構文解析するために、サンプルもエンコードされた属性値を使用します。Unicode ストリングも文字の破損を避けるために使用されます。
2. 複数の言語をサポートするには、言語ごとにプロパティー・ファイルを使用して、CSR のセッション内で選択された言語 ID に応じて、翻訳名またはトピックを検索することができます。

表 22.

メソッド	説明	Notes
静的ストリング getOpenTagString (String tagName)	オープン・タグ・ストリングを戻す	LiveHelpConfiguration. getOpenTagString (″HELLO″) はストリング <HELLO> を戻す
静的ストリング getCloseTagString(String tagName)	クローズ・タグ・ストリングを戻す	LiveHelpConfiguration. getCloseTagString (″HELLO″) はストリング </HELLO> を戻す
静的ストリング getTopicGroupElement String (String sGroupName)	トピック・リストの GROUP エレメント・ストリングを戻す	LiveHelpConfiguration. getTopicGroupElementString (″myGroup″) はストリング <GROUP NAME = ’myGroup’> を戻す
静的ストリング getTopic ElementString (String sTitle、String sText)	トピック・リストの QUESTION エレメント・ストリングを戻す	LiveHelpConfiguration. getTopicElementString (″myTitle″、 ″myText″) は <QUESTION TITLE =″myTitle″ TEXT =″myText″> を戻す

ストアの URL リストの定義

各ストアには、CSR がチャット・セッション中に顧客のブラウザーに送信できる URL を定義するファイルが含まれます。ファイル `CustomerCareStoreURLList.jsp` は以下のディレクトリーにあります。

- *WAS_installdir*/installedApps/*instance_name*/WC_*instance_name*.ear/Stores.war/*storedir*/CustomerServiceArea/CollaborationSection
- > 400 *WAS_userdir*/WAS_*instance_name*/installedApps/*cell_name*/WC_*instance_name*.ear/Stores.war/*storedir*/CustomerServiceArea/CollaborationSection

この JSP ファイルは、ストアの URL 情報が含まれる、以下の XML ストリングを戻します。

```
<?xml version="1.0" encoding="UTF-8"?>
<WCS_CUSTOMERCARE>
     <URL_LIST>
          <GROUP NAME="URLGroupName" >
               <PAGE NAME="IBM" URL="http%3A%2F%2Fwww.ibm.com" ></PAGE>
          </GROUP>
     </URL_LIST>
</WCS_CUSTOMERCARE>
```

グループ・エレメント (GROUP) は、上で説明されているように、複数の URL がある URL グループを定義します。これには以下の属性があります。

- Name: URL グループの名前。Name は必須の属性です。グループ名が重複する場合、最新のグループ定義が使用されます。

ページ・エレメント (PAGE) は単一 URL アドレスを定義します。これには以下の属性があります。

- Name: URL ページの名前。Name は必須の属性です。ページ名が重複する場合、最新のページ定義が使用されます。
- URL: ページの URL アドレス。URL は必須の属性です。

例 : CustomerCareStoreURLList.jsp: 以下のコードは、B2B 向けのサンプル・ストア、ToolTech からのものです。 CustomerCareStoreURLList.jsp:

```
<% // Create XML string
String strCfg=ECLivehelpConstants.EC_CC_XML_HEADER
 + LiveHelpConfiguration.getOpenTagString(ECLivehelpConstants.
EC_CC_XML_ROOT)
 + LiveHelpConfiguration.getOpenTagString(ECLivehelpConstants.
EC_CC_XML_URL_LIST);
%>

<% //unmark following block to add URL group/pages
// start of URL group block, repeat for more URL groups
strCfg=strCfg + LiveHelpConfiguration.getURLGroupElementString("URLGroupName");
 // start of URL pages block, repeat for all pages in the same group
 strCfg=strCfg + LiveHelpConfiguration.getURLPageElementString
("IBM","http://www.ibm.com")
          + LiveHelpConfiguration.getCloseTagString
(ECLivehelpConstants.EC_CC_XML_URL_PAGE);
 // end of URL pages block
strCfg=strCfg  + LiveHelpConfiguration.getCloseTagString
(ECLivehelpConstants.EC_CC_XML_URL_GROUP);
// end of URL group block
%>
<%
strCfg=strCfg  + LiveHelpConfiguration.getCloseTagString
(ECLivehelpConstants.EC_CC_XML_URL_LIST)
 + LiveHelpConfiguration.getCloseTagString(ECLivehelpConstants.EC_CC_XML_ROOT);
%>
```

サンプルは、LiveHelpConfiguration クラスのユーティリティー・メソッドを使用して、XML の正確さを確認します。以下のテーブルでは、このメソッドに関して詳細に説明します。

注: 問題を構文解析するために、サンプルもエンコードされた属性値を使用します。 Unicode ストリングも文字の破損を避けるために使用されます。

表 23.

メソッド	説明	Notes
静的ストリング getOpenTagString(String tagName)	オープン・タグ・ストリングを戻す	LiveHelpConfiguration .getOpenTagString (″HELLO″) はストリング <HELLO> を戻す
静的ストリング getCloseTagString(String tagName)	クローズ・タグ・ストリングを戻す	LiveHelpConfiguration. getCloseTagString (″HELLO″) はストリング </HELLO> を戻す
静的ストリング getTopicGroup ElementString (String sGroupName)	URL リストの GROUP エレメント・ストリングを戻す	LiveHelpConfiguration. getTopicGroupElementString (″myGroup″) はストリング <GROUP NAME = ’myGroup’> を戻す
静的ストリング getURLPage ElementString (String sName、String sURL)	URL リストの PAGE エレメント・ストリングを戻す	LiveHelpConfiguration. getURLPageElement String (″myName″、″myURL″) は <QUESTION TITLE =″myName″ TEXT =″myURL″> を戻す

カスタマー・ケアを使用した顧客のモニター

顧客がメインフレームでリンクを選択すると、改ページが戻され、動作中の以下のイベントのチェーンを設定します。

1. この改ページには、そのヘッダーに `CustomerCareHeaderSetup.jsp` ファイルが含まれます。
2. `CustomerCareHeaderSetup.jsp` は、以下の JavaScript 関数をフレーム・セットに呼び出します: parent.setPageParams()。
3. parent.setPageParams() は変数を更新して、現在のストア・ページ情報を保管し、UpdateStInfo() を呼び出します。
4. UpdateStInfo() は StUpdate フレームを再ロードして、CCShopperInfoUpdatePageView ビュー・コマンドを呼び出します。
5. CCShopperInfoUpdatePageView が戻されると、`CustomerCareInformationSetup.jsp` ファイルがロードされ、顧客の名前、ID、および顧客のショッピング・カートにあるアイテムを含む、カスタマー情報が集められます。

6. 次に、CCShopperInfoUpdatePageView は setCustomerName()、setShoppingCartItems() を呼び出して、このカスタマー情報を更新してから、changeSTAttributes() を呼び出して、アプレット内のすべての顧客属性を更新します。
7. changeSTAttributes() は、いくつかのアプレット・メソッドを呼び出して、CSR がモニターする属性値を更新します。
8. 続いて、changeSTAttributes() は、アプレットの changeWCSAttrs() メソッドを呼び出して、更新されたすべての属性値を Sametime システムに送信します。これによって、属性値が変更されたことが CSR アプレットに通知されます。

カスタマー・ケアを使用すると、以下のようにして、CSR と対応する顧客をモニターできます。

- 顧客の名前または ID の取得
- 顧客がブラウズするページの判別
- ショッピング・カート内のアイテム数の追跡
- カスタマイズしたモニター・アイテムの追跡

この情報を取得するために、カスタマイズしたコードがストア・ページに追加されます。以下のセクションでは、それぞれのモニター機能がサンプル・ストアにインプリメントされる方法について説明します。

顧客の名前または ID の取得

カスタマー・ケア・アプレットが立ち上げられ、CSR がログオンされると、 CSR はアプレットを使用する人物を名前またはショッパー ID によって識別することができます。サンプル・ストアには、カスタマー・ケア・アプレットで作動する特殊化されたコードが含まれており、これにより顧客の名前またはショッパー ID を判別します。このコードは、顧客がゲスト顧客、ショッピング・カートにアイテムを入れたゲスト顧客、または登録済み顧客のいずれかを判別し、名前または ID をその顧客に割り当てて、その名前をカスタマー・ケア・アプレットに戻します。そして、これらの名前が CSR に表示されます。たとえば、顧客がゲスト顧客であり、ショッピング・カートに何も入っていない場合、その顧客にはショッパー ID -1002 の生成済み ID が割り当てられます。顧客がゲスト顧客であり、ショッピング・カートにアイテムが入っている場合には、ショッパー ID が表示され、顧客が登録済み顧客の場合には姓名が表示されます。

サンプル・ストアは、CustomerCareInformationSetup.jsp をストア・ページのヘッダー・ファイルに組み込むことにより、顧客の名前または ID を取得します。CustomerCareInformationSetup.jsp ファイルにある以下のコードは、顧客の名前および ID を取得します。

```
<jsp:useBean id="userRDB" class="com.ibm.commerce.user.beans.
UserRegistrationDataBean" scope="page">
<% DataBeanManager.activate(userRDB, request); %>
</jsp:useBean>
<%
String customer_name="";
customer_name=userRDB.getUserId();
if (userRDB.findUser()){
   if (userRDB.getLastName() !=null && userRDB.getLastName().
length() > 0){
         if (locale.toString().equals("ja_JP")||locale.toString().
equals("ko_KR")||locale.toString().equals("zh_CN")||locale.toString().
```

```
equals("zh_TW"))
            {customer_name = userRDB.getLastName() + " " +
userRDB.getFirstName();}
          else
            {customer_name = userRDB.getFirstName() + " " +
 userRDB.getLastName();}
       }
}
if (customer_name.equals("-1002")) {
 customer_name="";
 }
else {
 // need to check order items
 ....
}
customer_name=customer_name.trim();
%>
```

注: 顧客がストア内の新規ページをブラウズするたびに、その顧客の名前または ID が最新表示されます。

CustomerCareInformationSetup.jsp ファイルにある以下のコードは、顧客の名前および ID を更新します。

```
<script language="javascript">
 ....
function changeSTAttributes()
{
 if (typeof top.setCustomerName == 'function') {
    top.setCustomerName(<%=userRDB.getUserId()%>, '<%=customer_name%>');
    top.setShoppingCartItems(<%=shoppingCartItems%>);
     top.changeSTAttributes();
    }
}
/script>
```

サンプル・ストアの「ログアウト」ページには、さらに多くのカスタム・コードが含まれています。このコードは顧客名を生成済み ID に設定し、ショッピング・カート内のアイテム数をゼロにリセットします。「ログアウト」ページは UserLogoffRouter.jsp です。カスタム・コードは以下のとおりです。

```
<HTML>
<HEAD>
<SCRIPT language="javascript">
if (typeof parent.setCustomerName == 'function')
     parent.setCustomerName (parent.WCSGUESTID, '')
   if (typeof parent.setShoppingCartItems == 'function')
     parent.setShoppingCartItems(0);
</SCRIPT>
</HEAD>
</HTML>
```

顧客がブラウズするページの判別

カスタマー・ケアを使用すると、顧客が現在ストア内のどのページをブラウズしているかを CSR が判別することができます。サンプル・ストアでは、CustomerCareHeaderSetup.jsp ファイルをヘッダー・ファイル (HeaderDisplay.jsp) に組み込むことにより、顧客が見ているページを判別します。 CustomerCareHeaderSetup.jsp ファイルにある以下のコードは、ページ URL 情報を取得して、ショッパー・アプレットを更新します。

```
<script language="javascript">
var PageName="";
var PersonalPage=false;
<%
 String pname = request.getRequestURI();
 int indpn = pname.lastIndexOf('/');
 indpn = pname.lastIndexOf('/', indpn-1);

 if(indpn != -1)
         pname = pname.substring(indpn+1);

 String headerType = (String) request.getAttribute("liveHelpPageType");
 if (headerType==null) headerType="";

 // Determine if this is a personal page or not
 if (headerType.equals("personal"))
 {
  %>
    if (typeof parent.setPageParams == 'function') {
     PersonalPage=true;
       parent.setPageParams('PERSONAL_URL', '<%=pname%>');
       }
  <%
 }
 else
 {
  %>
    if (typeof parent.setPageParams == 'function')
      parent.setPageParams(location.href, '%=pname%>');
  <%
 }
 %>
 Pagename="<%=pname%>";
</script>
```

CSR によって表示されるコンテンツは顧客によって表示されるものとは異なるため、CSR が個別設定情報が含まれる顧客ページを参照することができないようにしてください。たとえば、CSR はキャンペーン・ページ、契約で決定した価格が含まれるページ、またはユーザー ID が含まれるページ (住所録ページなど) にはアクセスできません。これらのページは、チャット・セッション中に CSR を惑わすことを避けるため、個人用としてマークする必要があります。

ページに個人用のマークを付ける (つまり、CSR が利用できないようにする) には、 CustomerCareHeaderSetup.jsp ファイルが組み込まれる直前に、サンプル・ストアは以下のコードをヘッダー・ページに組み込みます。

```
<flow:ifEnabled feature="customerCare">
<%
      request.setAttribute("liveHelpPageType", "personal");
%>
</flow:ifEnabled>
```

注: CSR は個人用というマークが付けられたページのコンテンツを見ることはできませんが、「顧客ページの表示」ボタンを使用することによって、そのページの URL は見ることができます。

ショッピング・カート内のアイテム数の追跡

カスタマー・ケアを使用すると、顧客がある時点でショッピング・カートに入れているアイテムの数を追跡することがでます。 CustomerCareInformationSetup.jsp ファイルにある以下のコードは、ショッピング・カートにあるアイテムの数を取得します。

```
<%
JSPHelper jhelper = new JSPHelper(request);
String storeId = jhelper.getParameter("storeId");
int shoppingCartItems = 0;
%>
<
jsp:useBean id="userRDB" class="com.ibm.commerce.user.beans.
UserRegistrationDataBean" scope="page">
<% DataBeanManager.activate(userRDB, request); %></jsp:useBean>

<%
....
 // need to check order items
 OrderListDataBean orderListBean = new OrderListDataBean();
 orderListBean.setStoreId(new Integer(storeId));
 orderListBean.setOrderStatus("P");
 orderListBean.setUserId(cmdcontext.getUserId());
 DataBeanManager.activate(orderListBean, request);
 Vector pendingOrders = orderListBean.getOrders();
 for (int k=0; k< pendingOrders.size(); k++) {
  OrderAccessBean next_order = (OrderAccessBean) pendingOrders.
elementAt(k);
  OrderDataBean orderBean = new OrderDataBean();
  orderBean.setOrderId(next_order.getOrderId());
  DataBeanManager.activate(orderBean, request);
  //Get items in the order
  OrderItemDataBean [] orderItems = orderBean.getOrderItemDataBeans();
  for (int i = 0; ((orderItems != null) &&
(i < orderItems.length)); i++)
  {
   OrderItemDataBean orderItem = orderItems[i];
   shoppingCartItems += orderItem.getQuantityInEJBType().intValue();
   }
  }
....
%>
```

以下の JavaScript 関数は、カート情報を Sametime アプレットに更新します。

```
<script language="javascript">
 ...
function changeSTAttributes()
{
 if (typeof top.setCustomerName == 'function') {
     top.setCustomerName(<%=userRDB.getUserId()%>,
 '<%=customer_name%>');
top.setShoppingCartItems(<%=shoppingCartItems%>);
     top.changeSTAttributes();
     }
}
</script>
```

注: CSR は**「ショッピング・カートの表示 (View Shopping Cart)」**ボタンを使用して、ショッピング・カートの中身を見ることができます。詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

サンプル・ストアでは、以下のコードを上記のページに追加することにより、ショッピング・カート内のアイテム数を判別します。

- 最初に int 変数が定義されます。

  ```
  int liveHelpShoppingCartItems = 0;
  ```

- 次に、カートに追加するオーダー・アイテムがあるときには、次のコード行を使用して数量を liveHelpShoppingCartItems に追加します。

```
liveHelpShoppingCartItems+= orderItem.getQuantityInEJBType().intValue();
```

- さらに、ページの最後で以下のコードを追加して、顧客名をゲスト・ショッパー ID に設定し、顧客のショッピング・カート内のアイテム数を取得します。

```
<script language="javascript">
 if (typeof parent.setShoppingCartItems == 'function')
  parent.setShoppingCartItems(<%=liveHelpShoppingCartItems%>);
</script>
```

「空のショッピング・カート (Empty shopping cart)」ページと「オーダーの確認」ページで次のコードを使用して、カート内のアイテム数をゼロにリセットします。

```
<script language="javascript">
 if (typeof parent.setShoppingCartItems == 'function')
  parent.setShoppingCartItems(0);
</script
```

カスタマイズしたモニター・アイテムの追跡

カスタマイズしたモニター・アイテムを追跡するには、以下のようにします。

1. モニター属性 ID を定義します (8000 より小さい属性 ID は予約済みです)。詳細については、 459 ページの『ストアのモニター・リストの定義』を参照してください。
2. JavaScript 変数を Sametime.js ファイルで定義して、属性 ID および値を保管し、すべてのフレームからのアクセスを許可します。例 : `var myTrackAttributeId=8001; var myTrackAttributeValue="anything"`。
3. JavaScript コードを追加して、この変数の値をストア・ページに割り当てるか、または更新します。例 : `top.myTrackAttributeValue="new Value"`。
4. . アプレットの setWcsAttribute() メソッドを呼び出して、changeSTAttributes() 関数の属性を Sametime.js ファイルで更新します。

```
function changeSTAttributes()
{

     if(CustomerAppletIsUp) {
       sametime.document.applets["InteractivePresenceApplet"].
setWcsAttribute(myTrackAttributeId,myTrackAttributeValue);
   sametime.document.applets["InteractivePresenceApplet"].
changeWCSAttrs();
 }
     else
         setTimeout("changeSTAttributes()",3000);
// wait for 3 sec and try again
}
```

注: changeWCSAttrs() メソッドの前にコードを挿入してください。

要求をカスタマー・ケア・キューへ直接送信する

デフォルトでは、顧客が CSR とのライブ・チャットを要求した場合、顧客の要求はデフォルトのキュー (queueId = 0) に置かれます。そして、さらに具体的なキューへの要求の発送は CSR の責任で行われます。ただし、ユーザーもストア・ページをカスタマイズして、定義されたカスタマー・ケア・キューへ要求を直接送信できます。ストア・ページをカスタマイズして定義されたキューへ要求を送信するには、以下のようにします。

1. WebSphere Commerce アクセラレーターを使用して、ストアのカスタマー・ケア・キューを作成します。詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。
2. これらのキューを CSR に割り当てます。詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。
3. 作成したキューのキュー ID を記録します。たとえば、キュー ID は 10020 です。
4. JavaScript を使用して、reqQueue 変数をストア JSP で更新します。例 : `top.reqQueue="10020";`。

キュー情報は、CSR が CSR アプレットを起動した後、自動的に更新されるわけではありません。新規キューがシステムに追加されるか、既存のキューが変更される場合、その変更は CSR が CSR アプレットを再度起動するまで有効ではありません。

キュー ID が CSR アプレットによって認識されない場合、そのキュー ID は CSR のデフォルトのキューに戻されます。この動作は、顧客に関連付けられているキュー属性値に影響しません。

カスタマー・ケアのカスタマイズ

カスタマー・ケアをカスタマイズするための以下のオプションがあります。

- アプレット・パラメーターのカスタマイズ
- ストア・メッセージのカスタマイズ

アプレット・パラメーターのカスタマイズ

以下のテーブルには、カスタマイズ可能なアプレット・パラメーターがリストされており、`CustomerCareFrameSetup.jsp` ファイルを変更することによって、カスタマー・ケアをカスタマイズすることができます。

表 24.

カスタマイズ可能なパラメーター名	説明	注
CHAT_FONT_SIZE	CharArea で使用されるフォント・サイズ	デフォルト値は 12 です
CHAT_FONT_COLOR	チャット領域で着信メッセージ用に使用されるフォント・カラー。	デフォルトは青 (#0000FF) です
CHAT_NAME_LENGTH	チャット領域でユーザー名を表示するのに予約済みの文字の長さ。	デフォルトは 15 です
WAIT_RANGE_1	整数値であり、ストアで待機中の顧客の数がこの値より小さい場合、警告メッセージ 1 が表示されます。そうでない場合、WAIT_RANGE_2 設定に応じて警告メッセージが表示されます。	メッセージ 1 のみが表示される場合は、-1 を使用します。

表 24. (続き)

WAIT_RANGE_2	整数値であり、ストアで待機中の顧客の数がこの値より小さく、WAIT_RANGE_1 より大きい場合、警告メッセージ 2 が表示されます。そうでない場合、WAIT_RANGE_3 設定に応じて警告メッセージが表示されます。	WAIT_RANGE_1 が -1 である場合は無視されます。この範囲を使用不可にするには -1 を使用します。
WAIT_RANGE_3	整数値であり、ストアで待機中の顧客の数がこの値より小さく、WAIT_RANGE_2 より大きい場合、警告メッセージ 3 が表示されます。そうでない場合には、警告メッセージ 4 が表示されます。	WAIT_RANGE_2 が -1 である場合は無視されます。この範囲を使用不可にするには -1 を使用します。
contentFrame	通常の WebSphere Commercestore ページに使用されるフレーム名	デフォルトは ″main″ です
COUNTER_UNIT_PAGE	どれくらいの頻度でページ・カウンターが 1 ずつ増加するかを示す整数値。	デフォルトは 30 秒です。ストアの CustomerCare MonitorList.jsp ファイルで定義されているものと同じ値であることを確認してください。
COUNTER_UNIT_SITE	どれくらいの頻度でサイト・カウンターが 1 ずつ増加するかを示す整数値。	デフォルトは 30 秒です。ストアの CustomerCareMonitorList.jsp ファイルで定義されるものと同じ値であることを確認してください。
COUNTER_UNIT_WAIT	どれくらいの頻度で待機カウンターが 1 ずつ増加するかを示す整数値。	デフォルトは 30 秒です。ストアの CustomerCareMonitorList.jsp ファイルで定義されるものと同じ値であることを確認してください。
WIDTH	希望するチャット・フレームの幅 (ピクセル)	デフォルトは 360 ピクセルです。招待メッセージの長さは、フィニアルの幅に影響します。
HEIGHT	チャット・フレームの高さ (ピクセル)	デフォルトは 400 ピクセルです。

ストア・メッセージのカスタマイズ

CSR に初めて接続したときに顧客に表示されるメッセージ (たとえば、 ″Hello, how can I help you?″ や ″Our office hours are from 9 a.m. to 9 p.m. ″) は、Sametime サーバーのプロパティー・ファイルに保管されます。プロパティー・ファイルは 2 つのファイル・タイプ、Customer.properties と Agent.properties に分

けられます。 Customer.properties ファイルには、顧客に表示されるメッセージが含まれており、 Agent.properties ファイルには、CSR に表示される情報が含まれています。どちらのファイルにも、WebSphere Commerce のインスタンスにインストールされたロケールごとに、対応するロケール固有のファイル (たとえば、Customer_de_DE.properties と Agent_de_DE.properties) があります。

これらのファイル内のメッセージを表示するには、以下のようにします。

1. Sametime サーバー上でプロパティー・ファイルを見つけます。デフォルトでは、プロパティー・ファイルは以下のディレクトリーにあります。
   - *Customer_Care_installdir*¥properties
2. 必要な変更を加えます。以下のテーブルには、それぞれのメッセージごとにメッセージ・キーがリストされています。

表 25.

メッセージ・キー	説明	Notes
WelcomeMessage	CSR がチャット要求を受け入れた後、顧客のアプレットに表示される最初のメッセージ。	
GoodbyeMessage	CSR または顧客がチャット・セッションを終了した直後に、顧客アプレットに表示されるメッセージ。	
PushPageMessage	CSR が顧客のブラウザーに URL を送信した後に、顧客のアプレットに表示されるメッセージ。	
CallCSRMessage	顧客がヘルプ要求を送信して、アプレットが Sametime サーバーに接続中に表示されるメッセージ。	
NoConnectionMessage	顧客アプレットが Sametime サーバーに接続できないか、または Sametime サーバーとの接続が切断された際に表示されるメッセージ。	
StoreCloseMessage	CSR が顧客にサービスを提供できない場合に表示されるメッセージ。	常に StoreWorkingHour メッセージと一緒に表示されます。
StoreWorkingHour	CSR が顧客にサービスを提供できない場合に表示されるメッセージ。ストアの操作時間について説明します。	常に StoreCloseMessage メッセージと一緒に表示されます。
WaitingMessage	顧客が要求を送信して、待機中の顧客の合計人数が WAIT_RANGE_1 より小さい場合に表示されるメッセージ。	

表 25. (続き)

WaitingMessage1	顧客が要求を送信して、待機中の顧客の合計人数が WAIT_RANGE_2 より小さく WAIT_RANGE_1 より大きい場合に表示されるメッセージ。	
WaitingMessage2	顧客が要求を送信して、待機中の顧客の合計人数が WAIT_RANGE_3 より小さく WAIT_RANGE_2 より大きい場合に表示されるメッセージ。	
WaitingMessage3	顧客が要求を送信して、待機中の顧客の合計人数が WAIT_RANGE_3 より大きい場合に表示されるメッセージ。	
CIWarningLabelLineOne	CSR が顧客へのチャット要求を開始した場合に表示される、招待メッセージの最初の行。	メッセージが表示される長さはチャット・フレームの幅に影響します。行の区切り文字を使用しないでください。
CIWarningLabelLineTwo	CSR が顧客へのチャット要求を開始した場合に表示される、招待メッセージの 2 番目の行。	メッセージが表示される長さはチャット・フレームの幅に影響します。行の区切り文字を使用しないでください。

注: 行の区切り文字 ″¥n″ をメッセージに挿入して、新規の行を開始します。そうしないと、メッセージが表示域の境界を超えてしまう可能性があります。

注: ロケールごとにプロパティーを変更して、メッセージが正しく変換されるようにする必要があります。

3. ファイルをクローズして保管します。

ストアへのカスタマー・ケアの追加

サンプルを使用しないストアにカスタマー・ケアを追加するには、以下のようにします。

パート 1: 前提条件をインストールする

カスタマー・ケアがストア内で作動するようにするには、以下を行う必要があります。

- Sametime サーバーをインストールする。詳細については、「*WebSphere Commerce 追加ソフトウェア・ガイド*」を参照してください。
- WebSphere Commerce Sametime 統合パッケージをインストールする。詳細については、「*WebSphere Commerce 追加ソフトウェア・ガイド*」を参照してください。

- WebSphere Commerce インスタンスを停止し、構成マネージャーで Sametime を使用可能にしてから、インスタンスを再始動する。詳細については、「*WebSphere Commerce 追加ソフトウェア・ガイド*」を参照してください。
- Sametime サーバーが WebSphere Commerce と同じ LDAP サーバーを使用しない場合、CSR を作成し、管理コンソールを使用して CSR をカスタマー・ケアに登録します。詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

パート 2: カスタマー・ケア統合ファイルをサンプル・ストアからコピーする

サンプル・ストア FashionFlow と ToolTech には、カスタマー・ケアをストアに統合するために使用する以下のファイルが含まれています。

- `Sametime.js`: すべてのフレーム用に組み込まれている JavaScript 関数が入っています。このファイルの関数は、メインフレーム内のページにある親接頭部を付けて呼び出されます (たとえば、parent.setCustomerName)。
- `CustomerCareBlankSetup.jsp`: 空の JSP ファイル。これは、フレームのプレースホルダーとして、CCShopperBlankPageView コマンドによって呼び出されます。
- `CustomerCareFrameSetup.jsp`: JavaScript 関数が入っており、ストアフロント用のカスタマー・ケア・アプレットを組み込みます。これは CCShopperFramePageView コマンドによって呼び出され、フレーム・セットに顧客アプレットをロードします。
- `CustomerCareAppletReadySetup.jsp`: アプレットが正しくロードされることを示します。これは CCShopperReadyPageView コマンドによって呼び出され、顧客アプレットが Javascript 関数に対して準備が整ったことを示しています。
- `CustomerCareHeaderSetup.jsp`: パラメーターをアプレットに渡すヘッダー・ファイルで、このヘッダーを含むページが個人用ページかどうかを示します。これは、顧客のページ URL を更新するためにすべてのページに組み込まれる必要があります。
- `CustomerCareInformationSetup.jsp`: 顧客の名前情報および ID を更新します。これは CCShopperInfoUpdatePageView コマンドによって呼び出されます。
- `CustomerCareMonitorList.jsp`: モニター・リスト構成ファイル。これは CSR によって呼び出され、モニター・リストをロードします。
- `CustomerCareStoreQuestionList.jsp`: トピック・リスト構成ファイル。これは CSR によって呼び出され、ストア・トピック・リストをロードします。
- `CustomerCareStoreURLList.jsp`: ストア・レベル URL リスト構成ファイル。これは CSR によって呼び出され、ストア URL リストをロードします。
- `CustomerCareChatSetup.jsp`: フレーム・セットなしでカスタマー・ケアを使用するためのセットアップ・ファイル。これは CCChatPageView コマンドによって呼び出され、フレーム・セット以外の構成で顧客アプレットを起動します。
- `EnvironmentSetup.jsp`: すべてのカスタマー・ケアによって使用される、ストア・レベル構成 (例、リソース・バンドル) 用の JSP ファイル。
- `index.jsp`: カスタマー・ケアのストア・フロー設定に応じて、顧客のブラウザーをフレーム・セット・ページまたは通常のカタログ・ページにリダイレクトするエントリー・ページ。

- StoreFramesetPage.jsp: これは StoreFramesetView コマンドによって呼び出され、カスタマー・ケア用にフレーム・セットを構成します。

Sametime 統合ファイルをサンプル・ストアからストアにコピーするには、以下のようにします。

1. 消費者向け (FashionFlow store) または B2B 向け (ToolTech) ストア用のストア・アーカイブ・ファイルを配置します。ストア・アーカイブ・ファイルは以下のディレクトリーにあります。
   - *WC_installdir*/samplestores
2. ConsumerDirect または B2BDirect フォルダーのいずれかを開いて、consumerdirect.sar または B2Bdirect.sar のいずれかを選択します。
3. WinZip または同様のツールを使用して、ストア・アーカイブ・ファイルをオープンします。
4. 上にリストされているファイルを選択します。
5. ストアの Web 資産が入っているディレクトリーにファイルを抽出します。サンプル・ストアと同じディレクトリー構造を維持するために、以下のファイル用のサブディレクトリーを作成することもできます。
   - .../CustomerServiceArea/CollaborationSection
     - CustomerCareAppletReadySetup.jsp
     - CustomerCareBlankSetup.jsp
     - CustomerCareBlankSetup.jsp
     - CustomerCareFrameSetup.jsp
     - CustomerCareInformationSetup.jsp
     - CustomerCareMonitorList.jsp
     - CustomerCareStoreQuestionList.jsp
     - CustomerCareStoreURLList.jsp
     - CustomerCareChatSetup.jsp
   - /include
     - CustomerCareHeaderSetup.jsp
     - EnvironmentSetup.jsp
   - ..¥
     - index.jsp
     - Sametime.js
     - StoreFramesetPage.jsp

パート 3: 顧客がブラウズするページを判別するためのコードを追加する

顧客がブラウズしているページを判別するには、以下のようにします。

1. CustomerCareHeaderSetup.jsp ファイルをストアのヘッダー・ファイルに組み込みます。たとえば、次のようにします。

```
<%@ include file="include¥CustomerCareHeaderSetup.jsp" %>
```

2. 以下のコードを、個人用のマークが付けられた (つまり、CSR がアクセスできない) ページに追加します。 CustomerCareHeaderSetup.jsp ファイルを組み込む前に、以下のコードを追加するようにしてください。

```
<flow:ifEnabled feature="customerCare">
<%
      request.setAttribute("liveHelpPageType", "personal");
%>
</flow:ifEnabled>=incfile%>"flush="true"/>
```

パート 4: カスタマー・ケアにリンクを追加する

顧客がストア内のカスタマー・ケアにアクセスできるようにするには、以下のようにします。

1. カスタマー・ケアへのリンクを置きたい場所を決定します。たとえば、リンクをナビゲーション・バーに置いて、顧客がいつでも利用できるようにしたり、ストア内の特定のページに置くことができます。
2. 以下のコードを、リンクを置くページにコピーします。

   注: infashiontext をそのストアに使用されるオブジェクト名と置換する必要がある場合があります。

```
<a href="javascript:if((parent.sametime != null))
top.interact();"><%=infashiontext.getString("LiveHelp")%> </a>
```

パート 5: カスタマー・ケア・フレーム・セットにリダイレクトするエントリー・ページを作成する

フレーム・セットは、ほとんどのカスタマー・ケア・フィーチャーが正常に機能するのに必要であり、顧客は StoreFramesetView コマンドを呼び出して、フレーム・セットをアクティブにする必要があります。例として、消費者向けまたは B2B 向けサンプル・ストアの index.jsp を参照してください。

第 42 章 ストアへの e-マーケティング・スポットの追加

e-マーケティング・スポットとは、ストア・ページ上に、キャンペーン・イニシアチブの特定顧客向けマーケティング・コンテンツを表示するためのスペースを確保した部分です。顧客がページを要求すると、ページの e-マーケティング・スポットは、ルール・サーバーと通信して、そのスポットに関連付けられているルール・ベースのコードを処理します。各 e-マーケティング・スポットには、1 つ以上のキャンペーン・イニシアチブが関連付けられています。キャンペーンとキャンペーン・イニシアチブの詳細については、233 ページの『第 20 章 キャンペーン資産』、および WebSphere Commerce オンライン・ヘルプを参照してください。

ストア・ページにキャンペーン・イニシアチブを正しく表示するには、JSP ファイルに e-マーケティング・スポットを追加してから、 WebSphere Commerce アクセラレーターを使用して、 e-マーケティング・スポットをデータベース内に登録する必要があります。この章では、e-マーケティング・スポットをストアの JSP ファイルに追加する方法を説明します。 WebSphere Commerce アクセラレーターを使用して、e-マーケティング・スポットをデータベース内に登録するための詳細については、 WebSphere Commerce のオンライン・ヘルプを参照してください。

e-マーケティング・スポット

以下に、e-マーケティング・スポットの例を挙げます。

```
<!-- ==========================================================================================
//*------------------------------------------------------------------------------------------
//* The sample contained herein is provided to you "AS IS".
//*
//* It is furnished by IBM as a simple example and has not been thoroughly tested
//* under all conditions.  IBM, therefore, cannot guarantee its reliability,
//* serviceability or functionality.
//*
//* This sample may include the names of individuals, companies, brands and products
//* in order to illustrate concepts as completely as possible.  All of these names
//* are fictitious and any similarity to the names and addresses used by actual persons
//* or business enterprises is entirely coincidental.
//*------------------------------------------------------------------------------------------
//*
========================================================================================== -->
<%
   /**
    * START - the following code should exist only once in a page, it initialize the
    *         command context and store data bean.
    */

   // create the store bean to get the store directory
   String collateralPath = "/webapp/wcs/stores/";
   com.ibm.commerce.command.CommandContext emsCommandContext =
      (com.ibm.commerce.command.CommandContext)
      request.getAttribute(com.ibm.commerce.server.ECConstants.EC_COMMANDCONTEXT);
   com.ibm.commerce.common.beans.StoreDataBean storeDataBean =
      new com.ibm.commerce.common.beans.StoreDataBean();
   storeDataBean.setStoreId(emsCommandContext.getStoreId().toString());
   com.ibm.commerce.beans.DataBeanManager.activate(storeDataBean, request);
   if (storeDataBean.getDirectory() != null) {
  collateralPath += storeDataBean.getDirectory() + "/";
 }

   /**
    * END - the following code should exist only once in a page, it initialize the
    *       command context and store data bean.
    */
%>
```

```
<!-- ========================================================================================
// The following HTML form submits the request on the e-Marketing Spot to the ClickInfo
// command which captures the campaign statistics, and redirects to the location specified
// by the URL parameter.
========================================================================================= -->
<form name="storeEmsForm" method="POST" action="/webapp/wcs/stores/servlet/ClickInfo">
   <input type="hidden" name="evtype">
   <input type="hidden" name="mpe_id">
   <input type="hidden" name="intv_id">
   <input type="hidden" name="URL">
</form>

<%
   /**
    * START - the following code can be used to drop multiple e-Marketing Spots onto the page.
    *         Customize the appropriate EMarketingSpot instance name and the e-Marketing Spot
    *         name before use. Duplicate this code if more than 1 spot is needed, but do not
    *         use the same spot name.
    */

 // create the e-Marketing Spot
 com.ibm.commerce.marketing.beans.EMarketingSpot eMarketingSpot =
      new com.ibm.commerce.marketing.beans.EMarketingSpot();

   // IMPORTANT - set the correct name here
   String emsName = request.getParameter("emsName");
   if (emsName == null) {
      emsName = "defaultEMSName";
   }
   eMarketingSpot.setName(emsName);

   // Set the catalog entry ID that is currently displaying on the page.  This is required if the
   // up-sell/cross-sell initiative is based on the content of the current page.
   String sourceCatentryId = request.getParameter("sourceCatentryId");
   if (sourceCatentryId != null) {
      eMarketingSpot.setSourceCatalogEntryId(sourceCatentryId); // use this method to set single ID
      //eMarketingSpot.setMultipleSourceCatalogEntryId(sourceCatentryId2);
         // use this method to set multiple IDs
   }

   // the maximum number of products/categories/ad copies that display through this
   // e-Marketing Spot can be set here
   eMarketingSpot.setMaximumNumberOfCatalogEntries(20);
   eMarketingSpot.setMaximumNumberOfCategories(20);
   eMarketingSpot.setMaximumNumberOfCollateral(20);
   eMarketingSpot.setMaximumNumberOfAssociateCatalogEntries(20);

   // instantiate the bean
   com.ibm.commerce.beans.DataBeanManager.activate(eMarketingSpot, request);
%>

<%
   // The following block is used to display the up-sell/cross-sell products associated with
   // this e-marketing spot.  The product display page which shows the selected product in the
   // campaign will be referenced through the submittion of the HTML form attached above.

   if (eMarketingSpot.getAssociateCatalogEntries() != null
      && eMarketingSpot.getAssociateCatalogEntries().length < 0) {
%>
<TABLE>
   <%
      for (int i=0; i<eMarketingSpot.getAssociateCatalogEntries().length; i++) {
         String associateCatalogEntryThumbNail = null;
         String associateCatalogEntryShortDescription = null;
         try {
            associateCatalogEntryThumbNail =
               eMarketingSpot.getAssociateCatalogEntries()[i].getDescription(emsCommandContext.
               getLanguageId()).getThumbNail();
            associateCatalogEntryShortDescription =
               eMarketingSpot.getAssociateCatalogEntries()[i].getDescription(emsCommandContext.
getLanguageId()).getShortDescription();
         }
         catch (Exception e) {
            // no description defined for the current language
         }
   %>
   <TR>
      <TD>
         <A HREF="/webapp/wcs/stores/servlet/ClickInfo?evtype=CpgnClick&mpe_id=<%=
```

```
            eMarketingSpot.getId() %>&intv_id=<%=
            eMarketingSpot.getAssociateCatalogEntries()[i].getInitiativeId() %>
            &URL=/webapp/wcs/stores/servlet/ProductDisplay&<%=
            com.ibm.commerce.server.ECConstants.EC_STORE_ID %>=<%=
            emsCommandContext.getStoreId().toString() %>&<%=
            com.ibm.commerce.server.ECConstants.EC_PRODUCT_ID %>=<%=
            eMarketingSpot.getAssociateCatalogEntries()[i].getCatalogEntryID() %>&<%=
            com.ibm.commerce.server.ECConstants.EC_LANGUAGE_ID %>=<%=
            emsCommandContext.getLanguageId().toString() %>">
         <IMG SRC="<%= collateralPath + associateCatalogEntryThumbNail %>"
            ALT="<%= associateCatalogEntryShortDescription %>" BORDER=0 WIDTH=60>
         </A>
      </TD>
      <TD><%= associateCatalogEntryShortDescription %></TD>
   </TR>
<%  } %>
</TABLE>
<% } %>

<%
 // The following block is used to display the advertisements associated with this
 // e-marketing spot.  The URL link defined with an advertisement can be referenced through
 // the submittion of the HTML form attached above.

 if (eMarketingSpot.getCollateral() != null && eMarketingSpot.getCollateral().length > 0) {
%>
<TABLE>
<%  for (int i=0; i<eMarketingSpot.getCollateral().length; i++) { %>
 <TR>
<%   if (eMarketingSpot.getCollateral()[i].getTypeName().equals("Image")) { %>
      <TD>
         <A HREF="javascript:document.storeEmsForm.evtype.value='CpgnClick';
            document.storeEmsForm.mpe_id.value='<%= eMarketingSpot.getId()
            %>';document.storeEmsForm.intv_id.value='<%=
            eMarketingSpot.getCollateral()[i].getInitiativeId()
            %>';document.storeEmsForm.URL.value='<%=
            eMarketingSpot.getCollateral()[i].getUrlLink()
            %>';document.storeEmsForm.submit();">
   <IMG SRC="<%= collateralPath + eMarketingSpot.getCollateral()[i].getLocation() %>">
   </A>
  </TD>
  <TD>
   <%= eMarketingSpot.getCollateral()[i].getMarketingText() %>
  </TD>
<%   } else if (eMarketingSpot.getCollateral()[i].getTypeName().equals("Flash")) { %>
  <TD>
         <EMBED src="<%= collateralPath + eMarketingSpot.getCollateral()[i].getLocation()
            %>" quality=high bgcolor=#FFFFFF WIDTH=120 HEIGHT=90
            TYPE="application/x-shockwave-flash"></EMBED>
  </TD>
  <TD>
         <A HREF="javascript:document.storeEmsForm.evtype.value='CpgnClick';
            document.storeEmsForm.mpe_id.value='<%= eMarketingSpot.getId()
            %>';document.storeEmsForm.intv_id.value='<%=
            eMarketingSpot.getCollateral()[i].getInitiativeId()
            %>';document.storeEmsForm.URL.value='<%=
            eMarketingSpot.getCollateral()[i].getUrlLink()
            %>';document.storeEmsForm.submit();">
   <%= eMarketingSpot.getCollateral()[i].getMarketingText() %>
   </A>
  </TD>
<%   } %>
 </TR>
<%  } %>
</TABLE>
<% } %>

<%
 // The following block is used to display the categories associated with this e-marketing
 // spot.  The category display page which shows the selected category in the campaign will
 // be referenced through the submittion of the HTML form attached above.

 if (eMarketingSpot.getCategories() != null && eMarketingSpot.getCategories().length > 0) {
%>
<TABLE>
<%
  for (int i=0; i<eMarketingSpot.getCategories().length; i++) {
   String catalogGroupName = null;
   String catalogGroupLongDescription = null;
         try {
```

```
            catalogGroupName = eMarketingSpot.getCategories()[i].getDescription
               (emsCommandContext.getLanguageId()).getName();
            catalogGroupLongDescription = eMarketingSpot.getCategories()[i].getDescription
               (emsCommandContext.getLanguageId()).getLongDescription();
         }
         catch (Exception e) {
            // no description defined for the current language
         }
%>
   <TR>
      <TD>
         <A HREF="/webapp/wcs/stores/servlet/ClickInfo?evtype=CpgnClick&mpe_id=<%=
            eMarketingSpot.getId() %>&intv_id=<%=
            eMarketingSpot.getCategories()[i].getInitiativeId()
            %>&URL=/webapp/wcs/stores/servlet/CategoryDisplay&<%=
            com.ibm.commerce.server.ECConstants.EC_STORE_ID %>=<%=
            emsCommandContext.getStoreId().toString() %>&<%=
            com.ibm.commerce.server.ECConstants.EC_CATEGORY_ID %>=<%=
            eMarketingSpot.getCategories()[i].getCategoryId() %>&<%=
            com.ibm.commerce.server.ECConstants.EC_CATALOG_ID %>=<%=
            eMarketingSpot.getCategories()[i].getCatalogId() %>&<%=
            com.ibm.commerce.server.ECConstants.EC_LANGUAGE_ID %>=<%=
            emsCommandContext.getLanguageId().toString() %>">
   <%= catalogGroupName %>
   </A>
  </TD>
  <TD><%= catalogGroupLongDescription %></TD>
 </TR>
<%  } %>
</TABLE>
<% } %>

<%
 // The following block is used to display the products associated with this e-Marketing
 // Spot.  The product display page which shows the selected product in the campaign will
 // be referenced through the submittion of the HTML form attached above.

 if (eMarketingSpot.getCatalogEntries() != null && eMarketingSpot.getCatalogEntries().length > 0) {
%>
<TABLE>
<%
   for (int i=0; i<eMarketingSpot.getCatalogEntries().length; i++) {
    String catalogEntryThumbNail = null;
      String catalogEntryShortDescription = null;
      try {
         catalogEntryThumbNail =
            eMarketingSpot.getCatalogEntries()[i].getDescription(emsCommandContext.
            getLanguageId()).getThumbNail();
         catalogEntryShortDescription =
            eMarketingSpot.getCatalogEntries()[i].getDescription(emsCommandContext.
            getLanguageId()).getShortDescription();
      }
      catch (Exception e) {
      // no description defined for the current language
      }
%>
   <TR>
      <TD>
         <A HREF="/webapp/wcs/stores/servlet/ClickInfo?evtype=CpgnClick&mpe_id=<%=
            eMarketingSpot.getId() %>&intv_id=<%=
            eMarketingSpot.getCatalogEntries()[i].getInitiativeId()
            %>&URL=/webapp/wcs/stores/servlet/ProductDisplay&<%=
            com.ibm.commerce.server.ECConstants.EC_STORE_ID %>=<%=
            emsCommandContext.getStoreId().toString() %>&<%=
            com.ibm.commerce.server.ECConstants.EC_PRODUCT_ID %>=<%=
            eMarketingSpot.getCatalogEntries()[i].getCatalogEntryID()
            %>&<%= com.ibm.commerce.server.ECConstants.EC_LANGUAGE_ID %>=<%=
            emsCommandContext.getLanguageId().toString() %>">
         <IMG SRC="<%= collateralPath + catalogEntryThumbNail %>"
            ALT="<%= catalogEntryShortDescription %>" BORDER=0 WIDTH=60>
   </A>
  </TD>
  <TD><%= catalogEntryShortDescription %></TD>
 </TR>
<%  } %>
</TABLE>
<% } %>

<%
 /**
```

```
  * END - the following code is used to drop multiple e-Marketing Spots onto the page.
  *       Customize the appropriate e-Marketing Spot name before use.
  *       Duplicate this code if more than 1 spot is needed, but do not use the same spot name.
  */
%>
```

上記の e-マーケティング・スポットでは、 4 種類のキャンペーン・イニシアチブがサポートされています。

- 商品推奨
- カテゴリー推奨
- 顧客キャッチ広告
- 取引管理アソシエーション

注: 各イニシアチブの詳細については、 233 ページの『第 20 章 キャンペーン資産』を参照してください。

e-MarketingSpot Bean

e-マーケティング・スポットは、e-MarketingSpot Bean を使用して、そのスポットに現在設定されているキャンペーン・イニシアチブの結果を戻します。この Bean の各種プロパティーを使えば、 e-マーケティング・スポットとそれに関連付けられているキャンペーン・イニシアチブをカスタマイズできます。 e-MarketingSpot Bean とそのプロパティーの詳細については、WebSphere Commerce オンライン・ヘルプを参照してください。

ストア・ページへの e-マーケティング・スポットの追加

ストア・ページに e-マーケティング・スポットを追加する手順は、次のとおりです。

1. どの JSP ファイルにスポットを表示するかを決めます。複数の JSP ファイルにスポットを追加することもできます。
2. JSP ファイルのどこにスポットを入れるかを決めます。
3. 479 ページの『e-マーケティング・スポット』中のサンプル e-マーケティング・スポットを、ストア Web アプリケーションの内部の新しい JPS ファイルにコピーします。
4. JSP ファイルのレイアウトに合わせて、サンプル e-マーケティング・スポットをカスタマイズします。
5. e-マーケティング・スポットのコードの中で、 e-マーケティング・スポットに名前を付けます。

   注: e-マーケティング・スポットには、その場所を含んだ分かりやすい名前を付けるようにしてください。たとえば、HomePageAd あるいは CheckOutPageRecommendation といった名前にします。そのようにすれば、そのスポットがどのページに表示されるのか、どのようなコンテンツが表示されるのかが分かりやすくなります。 1 つのページに複数の e-マーケティング・スポットを入れるのであれば、区別するために番号を付けることもできます。 e-マーケティング・スポットの名前は、有効な Java ID でなけれ

ばなりません。 WebSphere Commerce アクセラレーターを使用して、 e-マーケティング・スポットをデータベース内に登録するときにも、この同じ名前を使用する必要があります。

6. <jsp:include> タグを使用して動的に組み込む方法で、e-マーケティング・スポットを JSP ファイルに追加します。
7. JSP ファイル当たり複数の e-マーケティング・スポットが必要な場合は、ステップ 2 - 6 を繰り返します。
8. WebSphere Commerce アクセラレーターを使用して、 e-マーケティング・スポットをデータベース内に登録します。詳しい説明については、 WebSphere Commerce のオンライン・ヘルプを参照してください。

注:

a. 「langId=<%= languageId %>」という書式で、ストア ID、カタログ ID、言語 ID を URL に追加するときは、 e-マーケティング・スポットを組み込む JSP がその ID を公開する必要があります。その ID は、getCommandContext().getLanguageId()?) のようなコマンド・コンテキストで検索できるようになります。.

b. このコードはコマンドを直接参照していないので、 URL パラメーター CatalogDisplay の先頭は、 ″?″ ではなく ″&″ にしてください。

c. Java 動的組み込みタグを使用して e-マーケティング・スポットをストア JSP に追加し、新しい Dynacache フィーチャーを使用可能にして、 e-マーケティング・スポットを除外して、JSP のコンテンツをキャッシングできるようにします。アクセスのたびに e-マーケティング・スポットが更新されるので、適切な動的コンテンツが表示されます。動的組み込みを使用して e-マーケティング・スポットを ESpot.jsp という JSP ファイルに追加する方法の例を、以下に示します。

```
<jsp:include page="ESpot.jsp" flush="true">
<jsp:param name="emsName" value="StoreHomePage" />
</jsp:include>
```

d. 取引管理アソシエーション・イニシアチブを e-マーケティング・スポットにスケジューリングする際に、このアソシエーションのソースがページのコンテンツに基づいている場合は、com.ibm.commerce.marketing.beans.EMarketingSpot に備えられている setSourceCatalogEntryId(String source) メソッドと setMultipleCatalogEntryId(String source) メソッドを使用して設定できます。たとえば、「商品表示」ページで、このページに表示されている商品をアソシエーションのソースとして使用する場合は、以下のメソッドが呼び出されます。

eSpot.setSourceCatalogEntryId(*productId*)

eSpot は com.ibm.commerce.marketing.beans.EMarketingSpot クラスのインスタンスで、 *productId* はソース商品の ID です。

第 12 部 付録

付録 A. UML の凡例

統一モデリング言語 (Unified Modeling Language, UML) は、ソフトウェア設計でそれぞれのエレメントを表示するための標準図形言語です。以下は最も一般的な UML エレメントの例です。正式な仕様の詳細については、 http://www.rational.com および http://www.omg.org を参照してください。

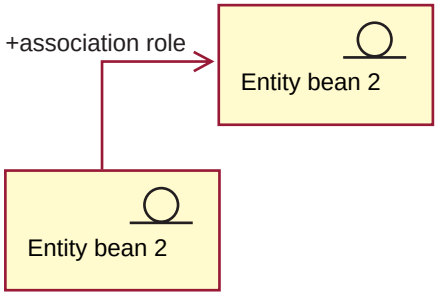
UML のダイアグラムは以下のアイテムで構成されています。

- ボックス: ボックスはオブジェクトのクラスを示します。クラス名がそのボックスの先頭に示されます。属性 (表示される場合) はクラス名の下に示されます。クラス名と属性は間の 1 行で区切られています。
- 線: 線は 2 つのクラスのオブジェクト間で可能な関連を示します。その線の終端の 1 つにあるクラスのオブジェクトは、その線のもう一方の終端にあるクラスと「関連付ける」ことができます。
- 塗りつぶされたひし形: 線の終端にある塗りつぶされたひし形は、値による埋め込みを表します。その線の反対側の終端にあるクラスのオブジェクトは、そのひし形が接点をもつクラスの唯一のオブジェクトの部分です。
- 空のひし形: 線の終端にある空のひし形は、参照による埋め込みを表します。その線のひし形の終端にあるオブジェクトは、その線の反対側の終端にあるクラスのグループ化オブジェクトのものと見なすことができます。
- 基数: これらは、関連線の終端に示されて、基数の制限を示します。以下のテーブルには、基数の制限が要約されています。

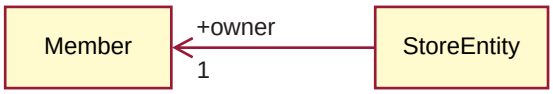
基数	関係タイプ
1	1 のみ
0..1	0 または 1
0..n	0 以上
1..n	1 以上

  基数の制限が表示されない場合で、関連線の終端に塗りつぶされたひし形が示されないかぎり、基数は 0..n と見なされます。この場合、基数は 1 でなければなりません。
- 正符号: 関連線の終端に示される正符号は、その線の終端のクラスのオブジェクトがその関連で 1 つの役割を果たすことを示します。正符号の後のテキストは、関連でのそのオブジェクトの役割を示します。
- 矢印: 関連線の終端の矢印は、 2 つのオブジェクト間の関係の方向が矢印の方向であることを示します。関連線に矢印がないのは、オブジェクト間の関係の方向が通常双方向であることを示します。

以下のダイアグラムも上記の概念を図示しています。

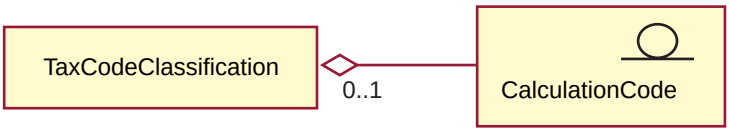


このダイアグラムは、 Enterprise Java Bean (EJB) を示すデコレーション・ステレオタイプ・シンボルで、 2 つのエンティティーを表示します。最初の Bean から 2 番目の Entity Bean への単一方向関連です。正符号の後にテキストが続いていて、Entity Bean 2 がこの関連でどのような役割を果たすかを記述しています。



このダイアグラムでは、StoreEntity は唯一の所有者を持っています。これは 1 つの Member です。 Member は 0 以上の StoreEntities を所有します。正符号は、その Member がその関連で 1 つの役割を果たすことを示します。この場合、この Member は StoreEntity の所有者です。矢印は、StoreEntity にその所有者を尋ねることによって StoreEntity の所有者を通常どおり調べることができることを示し、その Member が所有するすべての StoreEntity を尋ねる必要はありません。

このダイアグラムでは、OrderItem は常に、唯一の Order の部分です。 Order には 0 個以上の OrderItem があります。



このダイアグラムは、 CalculationCode が 0 以上の TaxCodeClassifications によってグループ化され、 TaxCodeClassification は 0 以上の CalculationCodes をグループ化します。

付録 B. データの作成

XML ファイルの形式でストア・データを作成する前に、以下に示すことを実行してください。

- 379 ページの『第 37 章 ストア・データのロードの概要』に記載されている情報を確認します。
- 作成する情報の順序を決定します。ストア・データに関する各章に載せられている情報は、データを作成する順序について示唆していますが、 XML ファイルの作成においては、親テーブルの情報が子テーブルの情報より前になければならない点に注意してください。資産をロードする順序の詳細については、 429 ページの『第 38 章 WebSphere Commerce データベース資産グループのロード』を参照してください。

サンプル・ストアのためのデータの作成

サンプル・ストア・アーカイブにあるデータは、ローダー・パッケージにとって妥当な、整形式の XML ファイルを使用しています。ストア・アーカイブ XML ファイルは移植可能であることが意図されており、データベースの特定のインスタンスに固有な、生成された基本キーを含めるべきではありません。その代わりに、発行時に ID リゾルバーによって解決される内部用の別名が使用されます。これらの規則を使用すると、サンプル・ストア・アーカイブを何度もコピーして発行することができます。

複数のストアを生成するために使用するサンプル・ストア・アーカイブを作成するのではない場合、または移植可能で、別の WebSphere Commerce インスタンスに発行できるストア・アーカイブを作成するのではない場合、ストアのストア・データを XML ファイル形式で作成する際に、上記の規則を使用する必要はありません。

結果として、サンプル・ストア・アーカイブでは以下の規則が使用されます。

- `ffmcenter_id="@ffmcenter_id_1"` などの中の @。 @ 記号の使用は、内部別名解決法と呼ばれます。ローダー・パッケージ・ユーティリティーの 1 つである ID リゾルバーは、 ID を必要とする XML エレメントのために、ID を生成します。 ID リゾルバーで使用されているテクニックの 1 つは、内部別名解決です。内部別名解決法を使用する場合は、 XML 文書内で基本キー (ID) の代わりに別名が用いられます。これで別名は、そのエレメントを参照するために、XML ファイル内の他の場所で使用できます。したがって、XML ファイルを構築するのに必要な固有索引を知っている必要はありません。管理コンソールでの発行中、またはローダー・パッケージの使用中に、 ID リゾルバーは @ 記号を固有な値に置き換えます。 XML ファイルの以下の例を参照してください。
  - ID リゾルバー実行前

```
<catalog
catalog_id="@catalog_id_1"
member_id="&MEMBER_ID;"
identifer=FashionFlow"
description="FashionFlow Catalog"
tpclevel="0"/>
```

– ID リゾルバー実行後

```
<catalog
catalog_id="10001"
member_id="-2000"
identifer=FashionFlow"
description="FashionFlow Catalog"
tpclevel="0"/>
```

ここで、10001 は ID リゾルバーによって割り当てられる固有な ID で、 -2000 は管理コンソールにおいてユーザーによって選択されたメンバー ID です。その結果、XML ファイルはローダー・パッケージを使用してロードされます。 ID リゾルバーからファイルを実行すると、単一の XML ファイルのセットから、多くのストアを確実に作成できます。

付録 C. データベース資産グループ

すべての WebSphere Commerce データベース資産は、作成とロードのために、グループに分けられています。これらのグループは、論理的に関連のあるテーブルのセットです。オブジェクト間の関係をロードするには、関連するオブジェクトが存在していなければならないため、これらのデータベース資産を編成する順序は、データのロードを実行する上で重要です。

ストア用の XML 形式のデータベース資産をロードする際、選択したグループだけをロードすることもできます。これらのグループは、以前の章で作成したデータベース資産 (たとえば、カタログやフルフィルメント) で構成されます。 437 ページの『データベース資産グループのロード』の説明に従ってデータ・グループをロードする前に、次の事柄を行います。

- ロードするデータベース資産グループを決定します。各グループには、資産をロードする前に満たしていなければならない従属関係があります。 『データベース資産グループの従属関係』に記載されている情報を確認します。
- 選択したデータベース資産グループ用の XML ファイルを作成または更新したことを確認します。資産に関する各章に載せられている情報は、データベース資産を作成する順序を示していますが、 XML ファイルの作成や更新においては、親テーブルの情報が子テーブルの情報より前になければならない点に注意してください。

データベース資産グループの従属関係

各データベース資産グループは、 WebSphere Commerce のデータベース・テーブルから情報を引き出します。ただし、データベース資産グループには、内部に従属関係が存在します。したがって、データベース資産グループは、別のデータ・グループから他の XML ファイルのデータを引き出すことはできませんし、各グループは、外部キーに依存していません。データベース資産グループが、別のグループに定義されている外部データを参照する必要があるときは、そのデータをユーザーが手作業で提供する必要があります。その場合は、1 つのグループから得られるデータに、そのドメインの外部 (別のデータベース資産グループ) で定義されているデータに対する*外部従属関係* が存在することになります。外部従属関係は、データベース資産グループに、別のグループ内のテーブルの基本キーに対する外部キー関係がある場合に発生します。データベース資産グループをロードするには、そのグループの外部従属関係を満たす必要があります。たとえば、以下の表では、ストア・データベース資産グループの外部従属関係の 1 つが、
`fulfillment.FFMCENTER.FFMCENTER_ID` になっています。つまり、ストア・データベース資産グループをロードするには、 WebSphere Commerce データベース内にフルフィルメント・データベース資産が存在している必要があるということです。

ロード・プロセスを開始する前に、以下の表を検討してください。各データベース資産グループは、データのロード元になる他のデータベース・テーブルに依存しています。

以下の点を覚えておいてください。

- 1 つのグループで満たせない外部従属関係もあります。すべてのストアが使用するサイト全体のデータベース資産 (汎用のデータベース資産) は、ブートストラップ によるインスタンス作成時にデータが取り込まれるため、すぐにアクセスできます。データベース資産グループに含まれているテーブルには、この種のデータに対する外部キー参照があります。このブートストラップ・データは、共通データとロケール固有データの 2 種類に分けられます。マルチリンガル・ストアの場合は、共通およびロケール固有のブートストラップ・データを両方とも選択する必要があります。たとえば、言語とメンバーのブートストラップ・データが必要です。インスタンス作成プロセスでは、 LANGUAGE テーブルに WebSphere Commerce でサポートされているストアの言語データが取り込まれ、ルート組織 (MEMBER.MEMBER_ID=-2001) とデフォルト組織 (MEMBER.MEMBER_ID=-2000) が作成されます。必要な場合は、ルート組織を使用しなければなりませんが、デフォルト組織については、代わりにストア所有者組織を作成するようにしてください。組織と組織の階層の詳細については、 WebSphere Commerce オンライン・ヘルプを参照してください。
- **外部従属関係**列に挙げられているファイルは、「*database asset group.database table.database column*」という形式の名前になっています。たとえば、store.STOREENT.STOREENT_ID ファイルであれば、 *store* データベース資産グループの *STOREENT* テーブルの *STOREENT_ID* 列からデータが取り込まれる、というわけです。名前の先頭が *bootstrap* になっているファイルは、 WebSphere Commerce インスタンスの作成時にデータが取り込まれます。
- **外部従属関係**列に挙げられているファイルには、**関連テーブル**への外部キー参照が含まれています。関連テーブルへのデータ取り込みを最初に実行する必要があります。
- 表を見やすくするために、商品説明などのマルチリンガル情報を含んだロケール固有テーブルは、別の列にまとめてあります。
- 以下の表に挙げられているそれぞれのテーブルは、WebSphere Commerce のサンプル・ストアのデータベース資産です。ストアのサイズ、機能、ニーズによって、テーブルが変わる場合もあります。以下の表に挙げられていないとしても、それぞれのストアの要件に応じて、ストアの資産を含んだデータベース・テーブルをすべて含めるようにしてください。

アクセス制御データベース資産		
外部従属関係	**データベース資産 XML ファイルの関連テーブル**	**データベース資産 XML ファイルの関連ロケール固有テーブル**
bootstrap.LANGUAGE.LANGUAGE_ID (ルート組織とストア所有者組織)、 bootstrap.MEMBER.MEMBER_ID (ルート組織とストア所有者組織)	**accesscontrol.xml** ACACTACTGP、 ACACTGRP、 ACACTION、 ACPOLICY、 ACRESCGRY、 ACRESGPRES、 ACRESGRP	**accesscontrol.xml** ACACGPDESC、 ACACTDESC、 ACPOLDESC、 ACRSCGDES、 ACRESGPDES

ビジネス・ポリシー・データベース資産		
外部従属関係	データベース資産 **XML** ファイルの関連テーブル	データベース資産 **XML** ファイルの関連ロケール固有テーブル
bootstrap.LANGUAGE.LANGUAGE_ID、boostrap.MEMBER.MEMBER_ID、store.STOREENT.STOREENT_ID (ストア所有者組織)	**businesspolicy.xml** POLICY、POLICYCMD	**businesspolicy.xml** POLICYDESC
キャンペーン・データベース資産		
外部従属関係	データベース資産 **XML** ファイルの関連テーブル	データベース資産 **XML** ファイルの関連ロケール固有テーブル
store.STOREENT.STOREENT_ID	**campaign.xml** CAMPAIGN、COLLATERAL、EMSPOT	**campaign.xml** COLLDESC

カタログ・データベース資産		
外部従属関係	**データベース資産 XML ファイルの関連テーブル**	**データベース資産 XML ファイルの関連ロケール固有テーブル**
bootstrap.LANGUAGE.LANGUAGE_ID、 bootstrap.MEMBER.MEMBER_ID (ストア所有者組織)、 store.STOREENT.STOREENT_ID、 shipping.CALCODE.CALCODE_ID、 tax.CALCODE.CALCODE_ID	**catalog.xml** BASEITEM、 CATALOG、 CATENTREL、 CATENTRY、 CATGROUP、 CATGRPREL、 CATTOGRP、 ITEMSPC、 ITEMVERSN、 RA、RADETAIL、 STOREITEM、 STORITMFFC、 VERSIONSPC **offering.xml** CATGRPTPC、 MGPTRDPSCN、 OFFER、 OFFERPRICE、 TRADEPOSCN **storefulfill.xml** INVENTORY **store-catalog.xml** DISPCGPREL、 DISPENTREL、 STORECAT、 STORECENT、 STORECGRP **store-catalog-shipping.xml** CATENTCALCD、 CATENTSHIP **store-catalog-tax.xml** CATENTCALD	**catalog.xml** ATTRIBUTE、 ATTRVALUE、 BASEITMDSC、 CATALOGDSC、 CATENTDESC、 CATGRPDESC、 PKGATTR、 PKGATTRVAL

コマンド・データベース資産		
外部従属関係	**データベース資産 XML ファイルの関連テーブル**	**データベース資産 XML ファイルの関連ロケール固有テーブル**
store.STOREENT.STOREENT_ID	**command.xml** CMDREG、 VIEWREG	なし
Business **契約データベース資産**		
外部従属関係		**データベース資産 XML ファイルの関連テーブル**
store.STOREENT.STOREENT_ID		契約データベース・テーブルは直接ロードされません。また、その他の WebSphere Commerce データ・グループとは異なるプロセスに従います。詳細については、445 ページの『契約資産の発行』を参照してください。
通貨データベース資産		
外部従属関係	**データベース資産 XML ファイルの関連テーブル**	**データベース資産 XML ファイルの関連ロケール固有テーブル**
store.STOREENT.STOREENT_ID	**currency.xml** CURCVLIST	**currency.xml** CURCONVERT、 CURLIST
フルフィルメント・データベース資産		
外部従属関係	**データベース資産 XML ファイルの関連テーブル**	**データベース資産 XML ファイルの関連ロケール固有テーブル**
bootstrap.LANGUAGE.LANGUAGE_ID、 boostrap.MEMBER.MEMBER_ID (ストア所有者組織)	**fulfillment.xml** FFMCENTER、 STADDRESS	**fulfillment.xml** FFMCENTDS
組織データベース資産		
外部従属関係	**データベース資産 XML ファイルの関連テーブル**	**データベース資産 XML ファイルの関連ロケール固有テーブル**
bootstrap.LANGUAGE.LANGUAGE_ID (ルート組織とストア所有者組織)、 boostrap.MEMBER.MEMBER_ID (ルート組織とストア所有者組織)	**organization.xml** ADDRBOOK、 ADDRESS、 MBRREL、 MEMBER、 ORGENTITY	なし

配送データベース資産		
外部従属関係	**データベース資産 XML ファイルの関連テーブル**	**データベース資産 XML ファイルの関連ロケール固有テーブル**
bootstrap.LANGUAGE.LANGUAGE_ID、 bootstrap.MEMBER.MEMBER_ID (ストア所有者組織)、 fulfillment.FFMCENTER.FFMCENTER_ID、 store.STOREENT.STOREENT_ID	**shipping.xml** CALCODE、 CALRULE、 CRULESCALE、 JURST、 JURSTGPREL、 JURSTGROUP、 SHIPMODE、 STENCALUSG **shipping.xml** SHPJCRULE、 SHPARRANGE	**shipping.xml** CALCODEDSC、 CALRANGE、 CALRLOOKUP、 CALSCALE、 SHPMODEDSC
ストア・データベース資産		
外部従属関係	**データベース資産 XML ファイルの関連テーブル**	**データベース資産 XML ファイルの関連ロケール固有テーブル**
bootstrap.LANGUAGE.LANGUAGE_ID、 bootstrap.MEMBER.MEMBER_ID (ストア所有者組織)、 bootstrap.SETCURR.SETCURR_ID、 fulfillment.FFMCENTER.FFMCENTER_ID	**store.xml** INVADJCODE、 RTNREASON、 STORE、 STORENT、 STORELANG、 VENDOR	**store.xml** FFMCENTDS、 INVADJDESC、 RTNRSNDESC、 STADDRESS、 STOREENTDS、 STORLANGDS、 VENDORDESC
ストア・デフォルト・データベース資産		
外部従属関係	**データベース資産 XML ファイルの関連テーブル**	**データベース資産 XML ファイルの関連ロケール固有テーブル**
shipping.SHIPMODE.SHIPMODE_ID (このファイルを最初にロードする)、 contract.CONTRACT.CONTRACT_ID、 store.STOREENT.STOREENT_ID	**store-default.xml** STOREDEF	なし

税データベース資産		
外部従属関係	**データベース資産 XML ファイルの関連テーブル**	**データベース資産 XML ファイルの関連ロケール固有テーブル**
bootstrap.LANGUAGE.LANGUAGE_ID、boostrap.MEMBER.MEMBER_ID (ストア所有者組織)、 store.STOREENT.STOREENT_ID、fulfillment.FFMCENTER.FFMCENTER_ID、store.STOREENT.STOREENT_ID	**tax.xml** CALCODE、CALRANGE、CALRLOOKUP、CALRULE、CALSCALE、CRULESCALE、JURST、JURSTGROUP、JURSTGPREL、STENCALUSG、TAXCGRY、TAXJCRULE **taxfulfill.xml** TAXJCRULE	**tax.xml** CALCODEDSC、CALSCALEDS、TAXCGRYDS

特記事項

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で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

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

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

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

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

IBM Canada Ltd.
Office of the Lab Director
8200 Warden Avenue Markham, Ontario L6G 1C7
Canada

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

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

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

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

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

表示されている IBM の価格は IBM が小売り価格として提示しているもので、現行価格であり、通知なしに変更されるものです。卸価格は、異なる場合があります。

本書はプランニング目的としてのみ記述されています。記述内容は製品が使用可能になる前に変更になる場合があります。

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

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。

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。

この製品で使用されているクレジット・カードのイメージ、商標、商号は、そのクレジット・カードを利用して支払うことを、それら商標等の所有者によって許可された人のみが、使用することができます。

商標

以下は、IBM Corporation の商標です。

AIX	IBM	WebSphere
AS/400®	IBM Payment Manager	
DB2	iSeries	
DB2 Universal Database	OS/400	
eServer	VisualAge®	

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

SET、SET ロゴ および SET Secure Electronic Transaction は、SET Secure Electronic Transaction LLC の商標です。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

UNIX は、The Open Group の米国およびその他の国における登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

Printed in Japan

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