Oracle® Application Server Wireless

開発者ガイド

10*g*（9.0.4）

部品番号 : B12350-02

2004 年 4 月

ORACLE®

Oracle Application Server Wireless 開発者ガイド , 10*g*（9.0.4）

部品番号 : B12350-02

原本名：Oracle Application Server Wireless Developer's Guide, 10*g* (9.0.4)

原本部品番号：B10948-01

目次

3 OracleAS Wireless Developer Kit

4 JDeveloper Wireless Extension

5 サービスの開発

6 Mobile Studio

7 Wireless Customization Portal

第 III 部　Wireless アプリケーションの開発

8　モバイル・ブラウザおよび音声アプリケーションの作成

9 マルチチャネル・サーバーの使用

10 メッセージ・アプリケーションの作成

11 通知エンジン

12 J2ME の開発とプロビジョニング

13 Web スクレイピング

14 ロケーション・サービスの使用

15 ユーザー・カスタマイズの有効化

16 請求

A サポートされる XHTML モジュール

B メディア・タイプとその機能

C XForms 仕様のサポート

D OracleAS Wireless による CSS のサポート

E CSS のレイアウト・プロパティの使用

F Oracle XML Grammar Subset

G JSP タグ・ライブラリ

索引

はじめに

このマニュアルでは、OracleAS Wireless を使用してモバイル・サービスを開発し、モバイル・デバイスに配信する方法について説明します。

対象読者

このマニュアルは、開発者が主要テクノロジと製品の機能をすばやく理解して、開発グループのためにこの製品を評価できるように構成されています。

このマニュアルの構成

このマニュアルは、次の章と付録で構成されています。

章または付録	内容
第 1 章「Oracle Application Server Wireless の概要」	OracleAS Wireless の概要
第 2 章「Oracle Application Server Wireless 開発者用のツールの概要」	OracleAS Wireless を使用したアプリケーションの作成
第 3 章「OracleAS Wireless Developer Kit」	OracleAS を完全にインストールせずに OracleAS Wireless アプリケーションを開発およびテストする方法
第 4 章「JDeveloper Wireless Extension」	JDeveloper の使用方法と拡張方法
第 5 章「サービスの開発」	サービス・マネージャを使用した Oracle Application Server Wireless リポジトリのサービス関連オブジェクトの作成および管理
第 6 章「Mobile Studio」	Oracle Mobile Studio は、OracleAS Wireless プラットフォーム用のモバイル・アプリケーションを開発、テストおよびデプロイするためのオンライン・ホスティング環境です。
第 7 章「Wireless Customization Portal」	ブラウザからポータルをカスタマイズする方法
第 8 章「モバイル・ブラウザおよび音声アプリケーションの作成」	XHTML および XFORMS を使用した、デバイスに依存しない文書の作成
第 9 章「マルチチャネル・サーバーの使用」	マルチチャネル・アプリケーションの開発
第 10 章「メッセージ・アプリケーションの作成」	モバイル・ユーザー間でのメッセージの送受信をサポートするメッセージ・アプリケーション
第 11 章「通知エンジン」	対象ユーザーと関連コンテンツを一致させる通知およびデータ・フィード
第 12 章「J2ME の開発とプロビジョニング」	J2ME アプリケーションを開発して、社内のバックエンド・アプリケーションにアクセスする方法
第 13 章「Web スクレイピング」	Web 対応のデバイスで使用するためのデバイスおよびマークアップ言語の再フォーマット
第 14 章「ロケーション・サービスの使用」	ロケーション・ベースのアプリケーション開発用の専用サービス

章または付録	内容
第 15 章「ユーザー・カスタマイズの有効化」	モバイル・アプリケーションの効率を改善するためのアプリケーションの調整
第 16 章「請求」	OracleAS Wireless の Billing Integration Framework には、拡張可能で柔軟なフレームワークが用意されており、請求可能なサービスのモデル化、請求可能なアクションの取得およびビリング・エンジンとの統合に使用されます。
付録 A「サポートされる XHTML モジュール」	XHTML 文書のサポートに関する参考資料
付録 B「メディア・タイプとその機能」	メディア・タイプとその機能に関する参考資料
付録 C「XForms 仕様のサポート」	サポートされている XForms プロパティのリスト
付録 D「OracleAS Wireless による CSS のサポート」	サポートされている CSS プロパティのリスト
付録 E「CSS のレイアウト・プロパティの使用」	CSS 機能の使用方法に関する参考資料
付録 F「Oracle XML Grammar Subset」	Oracle XML Grammar Subset の説明
付録 G「JSP タグ・ライブラリ」	JSP タグのリスト
「索引」	索引

関連ドキュメント

OracleAS Wireless および関連する製品やコンポーネントに関する重要情報が記載されている関連ドキュメントの一部を次に示します。

- 『Oracle Application Server Wireless 管理者ガイド』: 短時間でセットアップして実行するために必要なすべての情報。
- OracleAS Wireless リリース・ノート: マニュアルとヘルプの制作後に製品について判明した最新の注意事項。
- OracleAS Wireless オンライン・ヘルプ（製品に付属）。
- 製品のディレクトリ構造に含まれる Javadoc とサンプル・コード。
- OracleAS マニュアル（HTML および PDF ライブラリ）
- OTN-J（Oracle Technology Network Japan）

  http://otn.oracle.co.jp/

  OTN-J は、製品の情報、サンプル、更新および他のダウンロード用の主なリソースです。OTN-J を通じて、スタイルシート、ドライバ、マニュアルの更新版、サンプル・コード、デモ用ソフトウェアおよび他の役に立つリソースを入手できます。OTN-J に登録して、Oracle 製品と運用に関する最新情報へのアクセス権を取得し、ダウンロードしてください。未登録の場合は、無償で登録できます。

第 I 部

概要

第 I 部では、Oracle Application Server Wireless の概要について説明します。

1

Oracle Application Server Wireless の概要

項ごとに様々なトピックを記載しています。各項の内容は、次のとおりです。

- OracleAS Wireless の概要
- OracleAS Wireless の新機能
- ネットワーク内の OracleAS Wireless のデプロイ

OracleAS Wireless の概要

OracleAS Wireless は、Oracle Application Server の Wireless と音声に関するコンポーネントです。これを使用すると、企業およびサービス・プロバイダは、次のタイプのアプリケーションを効率的に作成、デプロイおよび管理できます。

- ブラウザ・ベースのアプリケーション
- 音声アプリケーション
- 非同期アプリケーション
- 通知
- J2ME アプリケーション

サービス・プロバイダにとって、OracleAS Wireless は、Wireless サービスを迅速に作成およびデプロイするためのサービス配信プラットフォームです。標準のインタフェースを使用して、モバイル・ネットワークをサード・パーティの開発者に公開します。この結果、ユーザー 1 人あたりの平均収益が向上します。OracleAS Wireless は、WAP、SMS、MMS、ロケーションおよび音声など、すべての種類のモバイル・サービスを開発するための統一プラットフォームです。これを使用すると、システム総保有コスト（TCO）の低減につながります。

サービス・プロバイダが OracleAS Wireless を使用すると、モバイル・ポータル、SMS サービス、J2ME プロビジョニング・サーバー、コンテンツ配信プラットフォーム、サード・パーティ統合プラットフォーム、メッセージ・ゲートウェイおよびロケーション・ゲートウェイを実行できます。

OracleAS Wireless コンポーネントは、5 つのグループに分けることができます。各グループは、優れた Wireless アプリケーションの迅速な作成や、既存アプリケーションの管理とデプロイに役立ちます。

- デバイス・ポータル : 開発したアプリケーションとコンテンツにアクセスするためのエンド・ユーザーの Wireless ポータル。
- モバイル・アプリケーション : 企業を迅速にモバイル対応にするための設定が容易なアプリケーション。
- マルチチャネル・サーバー : デバイスを検出し、コンテンツとアプリケーションをそのデバイスに変換するサーバー。
- 基本管理サービス : アプリケーションを拡張し、開発時間を短縮するサービス。Java API または Web サービスの形式です。
- 開発ツール : Wireless および音声アプリケーションのコーディング、テストおよびデバッグに開発者が利用するツール。

Wireless アプリケーションを構築するときは、Wireless Developer Kit（WDK）を活用してアプリケーションを作成し、JDeveloper や他の統合開発環境（IDE）でアプリケーションをテストできます。Wireless アプリケーションは、OracleAS Wireless の完全インストールをシ

ミュレートする開発用の PC またはラップトップで実行できます。モバイル・アプリケーションを作成するときは、このアプリケーションを OracleAS Wireless の基本管理サービスを使用してさらに拡張できます。基本管理サービスを使用すると、ロケーション認識、アラート、パーソナライズなどの優れた機能を Wireless アプリケーションに簡単に追加できます。これらのサービスは、オープンな規格の簡単な Web サービスまたは Java API で使用できます。Web クリッピングなどのその他のツールでは、既存の PC ブラウザ・アプリケーションを Wireless アプリケーションに変換できます。

OracleAS Wireless では、Wireless および音声アプリケーションのデプロイも簡略化されます。Wireless アプリケーションを、Oracle JDeveloper から Oracle Application Server 環境に自動的にデプロイできます。Oracle Application Server 環境にあるアプリケーションは、Web ベースのアプリケーション開発者ツールを使用して、デプロイ用に登録できます。別の Web サーバーにあるアプリケーションも、OracleAS Wireless に登録（その URL とともに）して、すべてのユーザーにモバイル・アクセスを提供できます。無線環境（OTA）配信、ブラウザ・アクセス、ダウンロードなど、いくつかの柔軟な方法でアプリケーションをエンド・ユーザーに配信できます。

アプリケーションを作成してデプロイした後は、アクセス権限、ユーザーとグループまたはシステム全体の管理が必要となる場合があります。OracleAS Wireless には、これらの各タスク用の完全なツールが、直感的な Web ベース・ツールのセットで用意されています。

OracleAS Wireless の新機能

OracleAS Wireless には、多数の新機能と拡張機能が追加されています。

マルチチャネル・サーバー

OracleAS Wireless の中心機能はマルチチャネル・サーバーです。この機能を使用すると、SMS、音声アクセス、WAP、Pocket PC などの複数の配信方法を使用してアプリケーションにアクセスできます。マルチチャネル・サーバーは、モバイル・アプリケーションに対するインテリジェントな Wireless プロキシとして機能することによって、開発を簡略化し、開発にかかる時間とコストを大幅に削減します。開発者は、多岐にわたるモバイル・デバイスやネットワークについて考慮する必要はなく、すべてのチャネルに対するモバイル・アプリケーションを、将来も使用可能な 1 つのオープンな規格の言語で作成することに専念できるようになりました。新しいマルチチャネル・サーバーでは、以前の OracleAS Wireless リリースの既存マルチチャネル機能が拡張されています。

XHTML で作成されたアプリケーションは、マルチチャネル・サーバーを経由して、任意のデバイスおよびネットワーク用に変換されます。たとえば、マルチチャネル・サーバーを経由した XHTML アプリケーションは、音声ダイアログを使用して電話でコールした場合は VoiceXML に変換され、WAP 電話でアクセスした場合は WML に変換されます。

J2ME サポート

Oracle J2ME Developer's Kit には、モバイル・デバイス用に最適化された方法で、Web サービスを J2ME デバイスに拡張する機能があります。

Java 2 Micro Edition（J2ME）では、オープンな規格のクライアント側の開発を可能にする、モバイル・デバイス用の軽量オペレーティング・システムが提供されます。大量の J2ME 対応電話が発売されているため、ベンダーは目的のモバイル・デバイスに対して J2ME アプリケーションを効率的に作成、管理および配信する方法を必要としています。OracleAS Wireless には、J2ME アプリケーションを作成し、モバイル・デバイスに配信するためのエンド・トゥ・エンドの完全なサポートが用意されています。J2ME サポートには、J2ME Developer's Kit と J2ME プロビジョニング・システムが含まれます。

ただし、モバイル・デバイスは処理能力に限界があるため、J2ME アプリケーションの難易度は制限されます。J2ME アプリケーションが複雑になると、デバイスでの使用に適さなくなります。優れた J2ME アプリケーションを作成する 1 つの方法として、Web サービスを使用する方法があります。アプリケーションは、CPU に負荷のかかるロジックの一部をサーバー側の Web サービスにプッシュできます。ただし、J2ME デバイスからの Web サービスへのコールも CPU に非常に負荷がかかります。J2ME Developer's Kit を使用すると、J2ME アプリケーション開発者は、クライアント・スタブを使用し、Oracle Application Server の J2ME プロキシ・サーバーを介して Web サービスをコールできます。また、MIDlet 開発者は、ネットワークが使用できない場合、リクエストとレスポンスのキャッシュなど、通信を最適化する組込み機能を利用できます。この場合は、ネットワーク接続が回復したときに、コールが自動的に再開されます。

J2ME アプリケーションをデプロイするために、OracleAS Wireless は、そのプロビジョニング・システムを使用して、J2ME アプリケーションのデプロイ、管理および配信を効率化します。Web ベースのアプリケーション管理によって、ユーザーは J2ME アプリケーションを管理と安全な格納のためにアップロードできます。バイトコード・インスペクタは、不当なコンテンツについてアプリケーションを検証します。OracleAS Wireless は、無線環境（OTA）をサポートし、ターゲットのユーザーやデバイスにアプリケーションを効率的に配信します。デジタル著作権管理（DRM）によって、J2ME アプリケーションの周囲にデジタル・レイヤーが追加され、アプリケーションを完全に制御するビジネス・ロジックがサポートされます。デジタル・ラッパーによって、請求方法とアプリケーションの存続期間の制御がサポートされます。

通知とマルチメディア・メッセージ

OracleAS Wireless では、アクション可能アラート、メッセージ調整およびフェイルオーバー配信制御に関する新機能が提供され、メッセージ機能がさらに強化されています。また、新機能である MMS 機能では、さらに豊富なメッセージ機能を使用できます。既存のメッセージ機能が拡張され、より柔軟なメッセージ・テンプレート、メッセージ偽装を防止するセキュリティ、メッセージ優先順位のサポート、およびさらに大量の通知を処理する機能が導入されています。

OracleAS Wireless では、グラフィックス、ビデオ、オーディオなどの豊富なモバイル・メッセージ用のマルチメディア・メッセージ（MMS）がサポートされます。MMS メッセージは、SMIL で固有に作成するか、またはオープンな規格の XHTML で作成できます。XHTML で作成されたメッセージは、OracleAS Wireless によって MMS と互換性のあるデバイス用に自動的に調整されます。この調整機能によって、メッセージは一度で記述され、すべてのターゲット・デバイスに対して自動的に最適化されます。

アクション可能アラートは、モバイル・デバイスとの間で応答ができる通知です。たとえば、株価アラートの場合は、目標価格に達した時点で、ユーザーに処置を促し株を売却できます。

ロケーションもアラートをトリガーできます。ロケーション・ベースのアラートでは、モバイル・ユーザーの現在のロケーションに基づいてアラート・メッセージが生成され、配信されます。たとえば、フィールド・サービス管理者は、緊急サービスを要求している顧客の 2 マイル以内にサービス技術者がいる場合にアラートを受信します。

マルチメディア適応サービスもマルチチャネル・サーバーの新機能です。OracleAS Wireless のマルチメディア適応サービスでは、イメージ、着信音、音声構文およびオーディオ / ビデオの各配信に関するデバイス固有の調整が提供されます。デバイスによって、サポートされるイメージ・フォーマット、および画面サイズや色の深度が異なります。リクエストに応答して OracleAS Wireless が実行するコンテンツ調整の一部には、デバイスにあわせたイメージの動的な調整があります。着信音調整では、着信音データを、RTTTL、iMelody、MIDI などの最も一般的な電話でサポートされているフォーマットに変換できます。着信音調整の柔軟なフレームワークによって、開発者は、着信音の新規フォーマット用のサポートを簡単に追加できます。

Wireless Developer Kit

Oracle Wireless Developer Kit は、Wireless および音声アプリケーションを開発するためのフットプリントの小さい OracleAS Wireless 開発環境です。この環境では、IDE、開発ツール、Web サービスまたはデバイス・シミュレータ（あるいはその組合せ）の使用に特別な柔軟性が提供され、開発プロセスが短縮されます。Wireless Developer Kit は、Wireless および音声アプリケーションを作成およびテストするために、PC またはラップトップ（接続または非接続）で使用できます。Wireless アプリケーションの作成とテストのために、Oracle Application Server を完全にインストールする必要がなくなりました。Wireless Developer Kit は、音声、モバイル・ブラウザ、J2ME およびメッセージの各アプリケーションの開発をサポートします。

オラクル社では、JDeveloper Wireless Extension という JDeveloper 用バージョンの Wireless Developer Kit を特別に用意しています。JDeveloper ユーザーは、この JDeveloper Wireless Extension を、コード・テンプレート、ウィザード、コード・インサイトおよび Oracle Application Server への自動デプロイを備えた完全な Wireless 開発に使用できます。

Web クリッピング

Wireless Web クリッピング・サーバーを使用すると、既存の Web コンテンツをクリップおよびスクレイプして、既存の PC ブラウザ・ベースのアプリケーションを再利用する Wireless アプリケーションを作成できます。Wireless Web クリッピング・サーバーは、多数のアプリケーションの作成に使用されます。それぞれのアプリケーションは、大規模な組織全体に散在する 1 つ以上の Web サイトからクリップおよびスクレイプされた Web コンテンツを表します。

Wireless Web クリッピング・サーバーには、次の機能があります。

- フォーム・ベースと JavaScript ベースの送信、および Cookie ベースのセッション管理付きの HTTP Basic 認証と Digest 認証など、様々な形式のログイン・メカニズムによるナビゲーション。
- クリッピングのファジー・マッチング。Web クリッピングの順序がソース・ページ内で変更された場合やその文字フォント、サイズまたはスタイルが変更された場合、その変更は、Wireless Web クリッピング・サーバーによって正しく識別され、Wireless Web クリッピング・アプリケーションのコンテンツとして配信されます。
- 広範囲にわたる Web コンテンツの再利用。このコンテンツには、HTML 4.0.1、JavaScript、アプレットで作成されたページの基本サポート、および HTTP GET および POST（フォーム送信）によって取得したプラグイン対応コンテンツが含まれます。

Wireless Web クリッピング・アプリケーションの定義はすべて、Oracle Application Server Infrastructure データベースに永続的に格納されます。パスワードなどの保護情報は、データ暗号化規格（DES）に従い、Oracle 暗号化テクノロジを使用して暗号化された形式で格納されます。

ロケーション・サービス

OracleAS Wireless ロケーション・サービスによって、すべてのロケーション・ベース・サービス（LBS）機能へのアクセスが提供されます。この機能には、オープンな規格の方法による、ユーザー・ポジショニング、ジオコーディング、マッピング、運転方向、ビジネス・ディレクトリの参照などがあります。アプリケーションまたは一般的なクライアントは、インクルード WSDL を使用して、LBS Web サービスを起動できます。また、OracleAS Wireless インスタンスは、サービス・プロバイダ・プロキシを使用することによって、LBS 機能をさらに便利に使用できます。この結果、LBS 機能を使用してアプリケーションを変更することなく、LBS プロバイダを切り替えることができます。

LBS 機能は、API のみでなく、OracleAS Wireless Tools を介しても使用可能にできます。LBS 機能は、（ユーザーの現在のロケーション、およびプライバシ管理を提供する）モバイル・ポジショニングを可能にし、モバイル・ユーザーのロケーションをいつ誰に対して使用可能にするかを制御できます。モバイル・ポジショニングとロケーション情報のキャッシュは、システムまたは個々のユーザーによって使用可能または使用禁止にできます。ユーザーは、モバイル・ポジショニングへのアクセス権を、特定の日付範囲と指定した時間枠で他のユーザーまたはユーザー・グループ（コミュニティ）に付与できます。

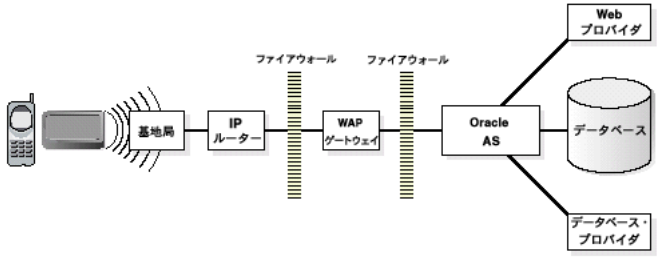
OracleAS Wireless ロケーション・サービスでは、モバイル・ユーザー / デバイスが現在のロケーションを OracleAS Wireless に送信することもできます。現在のロケーションは通常、GPS レシーバによって提供されます。この現在のロケーションは、その後、既存のモバイル・ポジショニングおよびプライバシ管理フレームワークを使用して問い合せることができます。ユーザーは、ロケーション・マーク機能を使用して、自分の位置を手動で選択することもできます。ロケーション・マークは、住所で指定される地点または都市、州、国で指定されるリージョンのいずれかです。

以前のリリースでは、ジオコーディング、マッピング、運転方向およびビジネス・ディレクトリの各サービスに対して、ユーザーは複数のコンテンツ・プロバイダを構成でき、プロバイダは、静的な順序付けまたはその可用性リージョンに基づいて選択されていました。このリリースでは、プロバイダのパフォーマンスと信頼性を監視し、選択基準を動的に調整する機能が追加されています。また、このリリースでは、管理者がシステム管理に利用できるパフォーマンス統計のログも記録されます。

ネットワーク内の OracleAS Wireless のデプロイ

OracleAS Wireless は、企業ネットワークに簡単に適合し、既存のバックエンド・データや従来型システムと容易に統合できます。アプリケーションは、OracleAS Wireless をモバイル・イネーブラとして使用しているサーバー上で実行できます。次の使用例では、OracleAS Wireless のデプロイ方法を示します。この例は、モバイル・デバイスがアプリケーションにコンテンツをリクエストする場合のネットワーク・コンポーネントを説明しています。

図 1-1 処理のリクエスト



この図は、WAP を使用した Wireless ネットワークを示しています。WAP は、複数ある Wireless 標準の 1 つです。目的のターゲット・デバイスに応じて、WAP ゲートウェイを他のゲートウェイと切り替えることができます。異なるプロトコルのサポートには、異なるゲートウェイが必要な場合があります。OracleAS Wireless は、Wireless サービスに対するリクエストを次のように処理します。

1. ユーザーが Wireless アプリケーションにリクエストを送信します。

   ユーザーがデバイスのマイクロブラウザを使用して、モバイル・デバイスからアプリケーションをリクエストすると、デバイスによって、Wireless ネットワークの基地局にそのリクエストが送信されます。リクエストは、使用中のデバイスの種類に応じて、様々なプロトコルを介して送信できます。これらのプロトコルは、帯域幅の制限と断続的な接続性を伴う Wireless ネットワークで動作するように最適化されています。そのため、これらは、既存の Wireless ネットワークでは標準的なインターネット HTTP プロトコルよりも効率的です。

2. Wireless リクエストがインターネット・リクエストに変換されます。

   リクエストは、Wireless ネットワークから従来型のインターネットに渡される前に、ゲートウェイによってネットワーク・プロトコルから標準的なインターネット HTTP プロトコルに変換されます。WAP 対応のデバイスの場合は、WAP ゲートウェイにより WTP が HTTP に変換されます。多数のゲートウェイがあり、通常は、デバイスのタイプごとに 1 つのゲートウェイが存在します。ゲートウェイでは、リクエストがプロトコル間でマップされるのみでなく、メッセージを Wireless ネットワークから従来型のインターネット・インフラストラクチャである HTTP に渡すこともできます。

3. OracleAS Wireless によって Wireless セッションが確立されます。

   ゲートウェイによって、Wireless リクエストが HTTP URL に変換されると、メッセージは標準的なインターネット・リクエストとして OracleAS Wireless に送信されます。OracleAS Wireless とゲートウェイは、相互を認証してから、セッションを確立します。デプロイ作業環境に従って、アプリケーションは、（セキュリティ作業環境に応じた）ユーザー認証用のユーザー名とパスワードを要求します。

4. Wireless アプリケーションが最適化され、ユーザーに送信されます。

   リクエストを受信した OracleAS Wireless は、そのリクエストを次の 3 つのステップで処理します。

   a. アプリケーションのコンテンツがそのソースから取得されます。アプリケーションは Web サーバーに存在している場合があるため、OracleAS Wireless は、アプリケーションのコンテンツに対して HTTP リクエストを送信します。

   b. OracleAS Wireless によってアプリケーションのコンテンツが取得されると、アプリケーションはユーザー作業環境に基づいてカスタマイズされます。ユーザーは、ローカライズされた情報、表示オプションなどを使用できます。

   c. 使用される特定のプロトコル用にコンテンツが変換されます。ユーザーは様々な WAP デバイスを使用しているため、OracleAS Wireless は、デバイスのタイプ、画面サイズ、色オプションを検出し、最も効率的なフォーマットで提供するようにコンテンツを最適化します。

第 II 部

Oracle Application Server Wireless 開発者用のツール

第 II 部では、Oracle Application Server Wireless の開発者用ツールについて説明します。

2

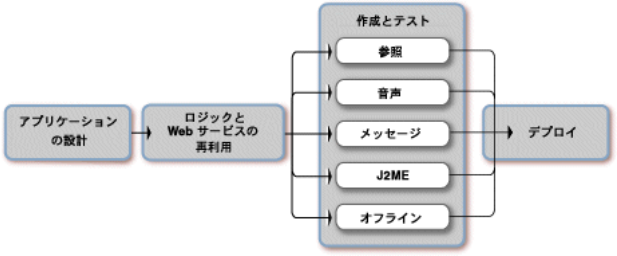
Oracle Application Server Wireless 開発者用のツールの概要

項ごとに様々なトピックを記載しています。各項の内容は、次のとおりです。

OracleAS Wireless の開発パス

この章では、Wireless 開発パスの高度な手順を紹介し、モバイル・アプリケーションの作成に使用するいくつかのツールについて説明します。アプリケーションの作成には様々なパスが考えられます。この章は、PC ベースのアプリケーションを作成するための現在の方法を、Wireless 固有の概念の追加によって補完することを目的としています。モバイル・アプリケーション作成の最初の手順は、その設計です。設計フェーズでは、モバイル対応の対象が、ビジネス・プロセスなのか、生産性ツールなのか、またはエンタテインメント・アプリケーションなのかを見極めます。

図 2-1 Wireless の開発パス



モバイル・アプリケーション設計プロセスへの最も効率的なアプローチ方法は、そのプロセスを 4 つの手順に分けることです。最初に、アプリケーションまたはビジネス・プロセスをモバイルにするためのビジネス・ケースを作成します。これには、モバイル・ソリューションの具体的な利点の判断が含まれます。モバイル・アプリケーションのビジネス・ケースに関する詳細は、OTN: http://www.otn.oracle.com/mobile の Mobile Center を参照してください。

2 番目に、使用状況、必要なビジネス・フローおよび最適なアクセス・チャネル（モバイル・ブラウザ、音声アクセス、メッセージまたはネットワーク・アクセスなしのアプリケーション（オフライン））を判別し、アプリケーションの範囲を決定します。3 番目の手順では、そのチャネルを使用可能にするための実際の開発モデル（XHTML/J2EE や J2ME）を選択します。4 番目の重要な手順は、アプリケーションのデプロイに関する考慮です。デプロイによって、アプリケーションをエンド・ユーザーに配信するプロセスとそのアプリケーションのライフ・サイクルの管理に使用する方法がカプセル化されます。このプロセスを通して、開発者は次のことを確認できます。

- 適切なビジネス・プロセスのモバイル化
- 適切なアクセス・チャネルを使用したビジネス・プロセスの拡張
- 適切な量の機能の開発
- 適切なデプロイ・モデルの決定
- ソリューションに具体的な投資利益（ROI）があることの確認

Web サービスの活用とビジネス・ロジックの再利用

Oracle Application Server Wireless には、ロケーション・ベース・サービス、メッセージ・サービスおよびパーソナライズ・サービスなど、使用中のモバイル・アプリケーションを大幅に拡張する Web サービスが用意されています。たとえば、社内の Web サービスを Oracle Application Server Wireless のメッセージ Web サービスと結合すると、在庫レベルが危機的な状態まで低下したときに管理者に自動的に警告できます。Oracle Application Server Wireless のメッセージ Web サービスは、アドレス、メッセージ・テキストおよび配信チャネル（音声、FAX、SMS または電子メール）を入力として受け入れます。次に、Web サービスはメッセージ・コンテンツを取得し、それぞれのアドレス、SMS、電子メールまたは電話番号（テキスト音声合成を使用）に送信します。開発者の作業は、メッセージ送信先の基礎となるインフラストラクチャやビジネス上の関係を考慮する必要がないため、大幅に簡略化されます。メッセージ Web サービスの詳細は、第 10 章「メッセージ・アプリケーションの作成」または OTN の Mobile Center を参照してください。

Wireless アプリケーションを作成するとき、すでに存在するビジネス・ロジックを複製する必要はありません。様々なバックエンド・システムとアプリケーションを Web サービスとして公開し、Oracle Application Server Wireless を使用して任意のデバイスに配信できます。たとえば、PC 専用の従来型フィールド・サービス・システムを再利用する場合があります。既存のフィールド・サービス・システムを使用し、XHTML または J2ME を使用してモバイル・ビューを作成するだけで再利用できます。

アプリケーションの作成とテスト

Oracle Application Server Wireless には、J2EE/XHTML モデルまたは J2ME モデルのいずれかを使用しているかによって、モバイル・アプリケーションの作成とテストに使用できるツールが 2 つ用意されています。J2EE/XHTML 開発モデルを使用すると、サーバー側のリアルタイムの Wireless アプリケーションと音声アプリケーションを作成できます。これには、メッセージ、音声インタフェース、PDA および携帯電話のブラウザなどのチャネルが含まれます。これらすべてのチャネル用に、1 つのオープンな規格の開発モデルでアプリケーションを作成できます。J2ME は、制限付きまたは断続的なネットワーク接続性を伴う小型画面付きのデバイスに最適です。J2ME を使用すると、ビジネス・ロジックおよび UI ロジックはデバイス上で処理でき、CPU に負荷のかかる処理についてはサーバーに対して Web サービスをコールできます。このコールは、ネットワーク接続が使用不可の場合はバッファに格納されます。

Oracle Application Server WirelessTools を使用すると、デバイスのメーカー（Nokia や Openwave など）が提供するモバイル・デバイス・シミュレータを利用できます。デバイス・シミュレータを使用すると、開発用の PC またはラップトップを使用して、電話上にアプリケーションを表示できます。パーソナル・コンピュータでは、標準的な Web ブラウザを使用してモバイル・アプリケーションをテストできますが、テストには様々なフォーム要素を持つ各種デバイス・エミュレータを使用することをお薦めします。これにより、様々なデバイスのフォーム要素のレンダリングに関連する Oracle Application Server Wireless の XML の構成メンバーを理解できます。Mobile Center には、最も一般的なデバイス用のシミュレータの一部がリストされています。

JDeveloper Wireless Extension（JWE）を使用すると、XHTML または J2ME のモバイル・アプリケーションを JDeveloper でコード作成、デバッグおよびテストできます。開発サイクル全体にわたり、次のすべての段階で、JWE によるサポートが提供されます。

- ウィザードとテンプレートを使用したアプリケーションの作成
- GUI ベースのツールを使用したアプリケーションの編集
- 統合されたエミュレータまたは実際のデバイス上でのアプリケーションのテスト
- 高度なデバッグ・システムとログ・ファイルを使用したアプリケーションのデバッグ
- 直感的な UI を使用したアプリケーションのデプロイ

JWE は、OTN: http://www.otn.oracle.com/mobile の Mobile Center からダウンロードできます。

Wireless Developer Kit（WDK）は、Oracle Application Server Wireless 用のフットプリントの小さい開発環境です。WDK を使用すると、Oracle Application Server Wireless 用の Wireless アプリケーションの開発に、すべての開発 IDE とデバイス・シミュレータを使用できます。WDK には、XHTML と J2ME のサンプル、デバイス検出、エラー・ロギングおよび Web サービスが付属しています。開発者は、開発用の PC またはラップトップで Oracle Application Server Wireless の完全インストールをシミュレートできます。WDK は、http://www.otn.oracle.com/mobile の Mobile Center からも入手できます。

アプリケーションのデプロイ

アプリケーションは、デプロイする前に実際のデバイスでテストする必要があります。JWE または WDK を使用して作成したアプリケーションをターゲット・デバイスのシミュレータでテストすると、実際のデバイスでのテストが可能な状態になります。

Oracle Application Server Wireless には、デプロイするためにアプリケーションを登録し、デプロイしたアプリケーションを管理できるように、Web ベースのツールがいくつか用意されています。アプリケーション登録用の 2 つのツールは、サービス・マネージャと Mobile Studio です。

Mobile Studio は、アプリケーションを実際のデバイスでテストするための開発者用のテスト環境です。Mobile Studio は Oracle Application Server Wireless とともにインストールされます。また、OTN: http://www.otn.oracle.com/mobile の Mobile Center で入手可能なホスティング・バージョンもあります。アプリケーションを Mobile Studio に登録すると、実際のモバイル・デバイスでアプリケーションを確認できます。また、ある電話番号に電話をかけ、音声を使用してアクセスしたり、SMS アプリケーションの場合は、SMS デバイス上でアプリケーションをテストすることもできます。

サービス・マネージャを使用すると、XHTML または J2ME アプリケーションを登録し、必要なユーザーにアクセス権限を付与できます。サービス・マネージャは、アプリケーションを本番環境にデプロイするために使用されます。サービス・マネージャを使用するには、Oracle Application Server Wireless をインストールする必要があります。

アプリケーションの配信

OracleAS Wireless には、モバイル・サービスを作成、管理および配信できるように、ロール固有の Web ベース・ツールが用意されています。OracleAS Wireless Tools には、リポジトリ・オブジェクトの管理、サーバーの管理およびアプリケーションの配信を実行するためのウィザードが含まれています。

Web ベース・ツールの詳細は、『Oracle Application Server Wireless 管理者ガイド』を参照してください。

3

OracleAS Wireless Developer Kit

項ごとに様々なトピックを記載しています。各項の内容は、次のとおりです。

- Wireless Developer Kit の概要
- WDK のインストールと構成
- WDK ログ・ファイル
- WDK チュートリアルを使用した Wireless アプリケーションの実行

Wireless Developer Kit の概要

OracleAS Wireless には、Oracle Application Server を完全にインストールせずに、Wireless アプリケーションを開発およびテストできる開発者キットが含まれています。このキットでは、次の領域のアプリケーション開発がサポートされます。

- J2ME: OracleAS Wireless Developer Kit（WDK）には、J2ME Web Services Software Development Kit（J2ME SDK）および J2ME Web サービス・プロキシ・サーバー（J2ME プロキシ・サーバー）が含まれています。MIDlet をテストするには、デバイスに J2ME SDK をインストールする必要があります。
- メッセージ : WDK には、OracleAS Wireless の Java メッセージ API の実装が含まれています。開発者は、これらの API を使用するアプリケーションを作成して、そのアプリケーションを変更することなく OracleAS Wireless サーバーにデプロイできます。また、開発者が開発アクティビティを迅速に開始できるようにいくつかの例が含まれています。
- ロケーション・サービス : ロケーション・サービスの作成に使用できる Web サービス WSDL ファイルがいくつかあります。
- Wireless クライアント : WDK には、クライアント・アプリケーションの作成方法に関するドキュメントおよび例とともに、OracleAS Wireless クライアントのインストーラが含まれています。
- マルチチャネル・サーバー : WDK には、マルチチャネル・サーバーの軽量バージョンである MCSLite が含まれています。MCSLite では、完全なマルチチャネル・サーバー製品と同じ調整機能が提供されます。Wireless および音声アプリケーションの開発者は、このメモリー・フットプリントの小さいサーバーを利用して、アプリケーションを OracleAS Wireless にデプロイする前にテストできます。

OracleAS Wireless Developer Kit は、Oracle JDeveloper Wireless Edition と緊密に統合されており、ウィザード、コード・テンプレートおよびデバイス・シミュレータなどの機能が提供されます。OracleAS Wireless Developer Kit はスタンドアロン・モードで提供されるため、IDE または開発ツールで OracleAS Wireless の開発機能を活用できます。

WDK のインストールと構成

Oracle Application Server WDK は、Oracle Application Server Developer CD に格納されています。また、Oracle JDeveloper Wireless Edition にも埋め込まれています。WDK は、非常に簡単にインストールおよび使用でき、通常インストール後の構成は必要ありません。詳細な構成については、このマニュアルの該当する項で WDK コンポーネント固有の構成を参照してください。

Oracle Application Server Wireless Developer Kit の構造

WDK をインストールした後のディレクトリ構造は次のようになります。

- [ORACLE_HOME]
  - wireless
    - bin
    - dtd
      - simpleresult
      - xhtml+xforms
    - examples
      - messaging
      - wclient
  - j2ee
    - applications/wdk/wdk-web
      - logs
      - repository
      - webservice
  - j2me
  - lib
  - server
    - classes
    - messages
  - wclient
  - wsdl

マルチチャネル・サーバー Lite（MCSLite）

この項では、開発者が WDK の MCSLite コンポーネントを使用して、マルチチャネル・アプリケーションを開発およびテストする方法を説明します。

MCSLite は、サーブレットとサーブレット・フィルタが含まれた J2EE Web アプリケーションです。メモリー・フットプリントは小さいですが、完全な MCS サーバーのコンテンツ調整機能をすべて備えています。その目的は、開発者がアプリケーションを OracleAS Wireless サーバーにデプロイする前に、完全にテストできるようにすることです。

MCSLite の主目標はテストです。このコンポーネントは、開発者がアプリケーションの実行内容をより深く理解できるように情報を提供します。MCSLite のログ・ファイルには、次の重要な情報が含まれます。

- デバイス / シミュレータから受信した HTTP ヘッダーとリクエスト・パラメータ
- バックエンド・アプリケーションに送信した HTTP ヘッダーとリクエスト・パラメータ
- バックエンド・アプリケーションから受信した HTTP ヘッダーとコンテンツ
- デバイス / シミュレータに送信した HTTP ヘッダーとコンテンツ
- リクエスト処理中のエラー

MCSLite は次の 2 つの方法でデプロイできます。

- ローカル : MCSLite は、調整対象の Web アプリケーションの前にサーブレット・フィルタとしてデプロイされます。このデプロイの利点は、MCSLite とコンテンツ生成 Web アプリケーションが同じ Java VM で実行されることです。この結果、アプリケーションの全体的なパフォーマンスが向上します。不利な点は、各アプリケーションとともに MCSLite をデプロイする必要があることです。
- リモート : MCSLite は、デバイスと Web アプリケーション間の HTTP プロキシとして機能します。このデプロイの利点は、MCSLite と関係なく Web アプリケーションを開発およびデプロイでき、その場合でもアプリケーションをあらゆるデバイスでテストできることです。不利な点は、デバイスからコンテンツまでの間に、デバイスから MCSLite サーバーへのホップと MCSLite サーバーから Web アプリケーションへのホップの 2 つの HTTP ホップがあることです。これはパフォーマンスに影響を与えますが、使いやすさの点を考慮するとそれほど問題ではありません。開発者にはこのデプロイ方法をお薦めします。

主な機能

MCSLite の主な機能は次のとおりです。

- 完全な調整機能（マルチチャネル・サーバーと同一）

  次に、1 回の調整サイクルの処理内容を示します。

  1. デバイスの検出 : 詳細は以降の説明を参照してください。

2. コンテンツの取出し : データ・ソースへの接続とバックエンド・アプリケーションで生成されたコンテンツのフェッチ。
3. コンテンツ・タイプの検出 : バックエンド・アプリケーションから戻されたコンテンツと HTTP ヘッダーが、適切なコンテンツ・タイプを取得するために検証されます。この検証は、適切なトランスフォーマを現在のデバイスとコンテンツ・タイプに基づいて選択するために必要です。
4. 変換 : デバイス不特定のマークアップ言語からデバイス固有のマークアップ言語への変換。

- 小さいメモリー・フットプリント : 高度な技術（遅延オブジェクト・インスタンス化など）によって、小さいメモリー・フットプリントを実現しています。この機能は、WDK でのアプリケーションの作成やテストに、あまり性能の高くないマシンを使用できるという点で役に立ちます。
- 柔軟なログ・ファイル・システム : コンテンツ調整処理に関するログ・ファイルが生成されます。ログ・ファイルの情報量は、WDK の `web.xml` から構成可能なログ・レベルによって決まります。Wireless アプリケーションをデバッグするときは、ログ・レベルを、最も多くの情報が生成される debug に設定してください。デフォルトの MCSLite 構成の変更方法は、次の項を参照してください。
- トランスフォーマとデバイス記述の自動再ロード : 自動再ロード機能によって、デバイスとトランスフォーマのメタデータに対する変更が、サーバーを再起動せずに自動的に取得されます。この機能は、マルチチャネル・サーバーに新規デバイスまたはトランスフォーマを追加するときに特に便利です。MCSLite では同じ XML 表現のデバイス・メタデータが使用されるため、MCSLite で新規デバイス記述を簡単に作成およびテストでき、その内容を MCS XML プロビジョニング・ツールを使用して MCS にアップロードできます。

MCSLite の使用方法

MCSLite は、非常に使いやすく設計されています。これを使用するには、最初に Web アプリケーションを作成し、デプロイする必要があります。アプリケーションの開発には、あらゆる Web テクノロジを使用できます。静的ページまたは動的ページのいずれも使用できます。また、MobileXML、あるいは XHTML+XForms または XHTML MP マークアップ言語のいずれかを使用できます。唯一の要件は、アプリケーションへのアクセスに HTTP プロトコルを使用することです。

MCSLite を使用したアプリケーションへのアクセス 使用方法は、現在のデプロイがローカルであるかリモートであるかによって異なります。

- ローカル・デプロイ : MCSLite Web アプリケーションとともにアプリケーションをデプロイします。最も簡単な方法は、JSP/ サーブレットを *`[ORACLE_HOME]`*`/wireless/wdk/server/applications/wdk/wdk-web` ディレクトリにコピーすることです。この方法は、テストする簡単なアプリケーションがあり、独自の Web アプリケーションを作成およびデプロイしない場合に使用してください。次に、このデプロイ例で MCSLite を使用する方法の例を示します。

MCSLite およびユーザー・アプリケーションの URL は
`http://apphost:port/myApp.jsp` です。

デバイスのブラウザ（シミュレータ）を起動し、アプリケーションの URL を入力します（つまり、アドレス・フィールドに
「`http://apphost:port/myApp.jsp`」と入力します）。

- リモート・デプロイ（推奨）: MCSLite では、この使用方法をお薦めします。その理由は、テストするアプリケーションが、（MCSLite でのテスト成功後に）その内容を変更せずに稼働中の OracleAS Wireless サーバーにデプロイできる独立した Web アプリケーションであるためです。次に、このデプロイ例で MCSLite を使用する方法の例を示します。

  テストするアプリケーション、およびその URL は
  `http://apphost:port/myApp.jsp` です。

  MCSLite がホスト名 MCSLitehost というマシンにデプロイされる場合、MCSLite コンテンツ取出しサーブレットの URL は
  `http://MCSLitehost:port/wdk/MCSLite` です。

  2 つの Web アプリケーションは、同じマシンに配置しても配置しなくてもかまいません。このデプロイ例を使用すると、複数の開発者が 1 つの MCSLite インスタンスを共有できます。アプリケーションにデバイスまたはシミュレータからアクセスするには、ブラウザに特別な URL を入力します。その特別な URL は、（前述の例を使用すると）`http://MCSLitehost:port/wdk/MCSLite/http/apphost/port/myApp.jsp` です。

  この特別な URL を生成する手順は、次のとおりです。

  * 始めに MCSLite の URL を記述します。
    `http://MCSLitehost:port/wdk/MCSLite`
  * この URL にスラッシュ（/）を追加し、その後にバックエンド・アプリケーションの URL（`http://apphost:port/myApp.jsp`）を追加します。結果は、`http://MCSLitehost:port/wdk/MCSLite/http://apphost:port/myApp.jsp` となります。
  * この新しい URL は無効です。修正するには、アプリケーションの URL のコロン、スラッシュ、スラッシュ（://）とコロン（:）をスラッシュ（/）に変更します。この結果が、特別な URL:
    `http://MCSLitehost:port/wdk/MCSLite/http/apphost/port/myApp.jsp` になります。

注意： 特別な URL には簡略フォームがあります。アプリケーションがデプロイされる HTTP ポートがポート 80（デフォルトの HTTP ポート）の場合は、バックエンド・アプリケーションの URL の port の部分を省略できます。プロトコルの部分（http）も省略できます。簡略 URL は、`http://MCSLitehost:port/wdk/MCSLite/apphost/myApp.jsp` となります。

バックエンド・アプリケーションへのパラメータの送信

MCSLite のデプロイ例に関係なく、バックエンド・アプリケーションへのパラメータの送信方法は同じです。通常のブラウザからのパラメータの送信と異なる点はありません。URL の問合せ部分にパラメータを追加するだけで送信できます。たとえば、次の 2 つのパラメータをアプリケーションに送信するとします。

`fname=John および lname=Doe`

この場合は、次のように URL に追加します。

```
http:// ... /myApp.jsp?fname=John&lname=Doe
```

注意：

- パラメータ名と値を URL にエンコードすることに注意してください。
- OracleAS Wireless の予約パラメータ名のリストがあります。独自のパラメータには別の名前を選択してください。予約済のパラメータはすべて、MCSLite によって排除されます。

MCSLite の URL リライティングとキャッシュ

MCSLite では、MCS と同じ URL リライティングとキャッシュ・メカニズムが使用されます。ただし、URL リライティングでは詳細フォームまたは簡略フォームのいずれを使用するかを示す構成パラメータが使用される点が異なります。

関連項目： URL リライティングとキャッシュの詳細は、第 9 章「マルチチャネル・サーバーの使用」を参照してください。

National Language Support（NLS）

関連項目： National Language Support の詳細は、9-47 ページの「グローバリゼーション（NLS）サポート」を参照してください。

MCSLite のログ・ファイル

ログ・ファイルには、開発者にとって重要な情報が含まれており、アプリケーション・テスト時にデバッグ情報を得るための優れた情報源です。ログ・ファイルに記録されるメッセージには、次の 4 つのタイプがあります。

- ERROR: MCSLite でのリクエスト処理中に重度（リカバリ不可能）の問題が発生した場合。最も一般的な問題は、バックエンド・アプリケーションの URL が無効、またはバックエンド・アプリケーションから戻されたコンテンツが無効というエラーです。これらエラーおよび他のすべてのエラーで、ログ・ファイルにはその問題の識別に必要なすべての情報が記録されます。
- WARNING: リクエスト処理時に問題が発生したが、MCSLite によってリカバリされ、リクエストが処理された場合。開発者は、このような警告の原因をすべて取り除く必要があります。
- INFO: リクエスト処理フローに関する情報メッセージ。
- DEBUG: バックエンド・アプリケーションではなく、MCSLite 自体の問題に関連する低レベルのメッセージ。MCSLite に関するバグと問題の報告時には、ログ・ファイルの情報を使用してください。

ログ・ファイルは、次のいずれかの方法で表示できます。

- `[ORACLE_HOME]/wireless/wdk/server/applications/wdk/wdk-web/logs` ディレクトリ内のファイルを直接オープンする方法。この場合は、MCSLite が稼働しているマシン（通常は開発者自身のマシン）に直接アクセスする必要があります。
- 使用中の PC の HTML Web ブラウザを使用して、Web ログ・ビューアにアクセスする方法。MCSLite ログ・ビューア（サーブレット）の URL は `http://MCSLitehost:port/wdk/log` です。Web ベースのログ・ビューアは、（リモート MCSLite デプロイなどで）1 つの MCSLite サーバーを開発者グループで共有している場合に便利です。

ログ・ファイルの各メッセージについて、デバイス・リクエストの IP アドレスとセッション情報が提供されます。この情報は、ユーザーが自分のデバイスから送信したリクエストを、別のユーザーのデバイスから送信されたリクエストと区別して見つける場合に役立ちます。

注意： log.xml ファイルの XFM-xxxx という書式のエラー・メッセージは、XForms プロセッサによって生成されます。

MCSLite の詳細な構成

MCSLite の利点の 1 つは、デフォルトの状態で動作し、構成する必要がないことです。オプションで、詳細な構成処理を実行できます。MCSLite では、ログ・ファイルの位置と名前、ロギング・レベル、XML 妥当性チェック・モード、デバイスおよびトランスフォーマの自動再ロードの有効化または無効化などの構成が可能です。これらの構成処理を実行するには、MCSLite の `web.xml` ファイルを編集します。`web.xml` は `[ORACLE_HOME]/wireless/wdk/server/applications/wdk/wdk-web/WEB-INF` にあります。

次に、MCSLite の `web.xml` 内の構成プロパティを示します。

- `wdk.log.file`: ログ・ファイルへの絶対パス（ディレクトリとファイル名）。たとえば、*`D:¥wdk¥logs¥wdk.log`* のように指定します。また、*`System.out`* または *`System.err`* を指定すると、標準出力または標準エラーを使用できます。値を指定しない場合は、デフォルトのログ・ファイル位置である `[ORACLE_HOME]/wireless/wdk/server/applications/wdk/wdk-web/logs/wdk.log` が使用されます。
- `wdk.log.level`: ログ・ファイルに記録する情報の量を指定します。有効な値は、debug、info、warning および error です。debug を指定すると、最も多くの情報が生成され、info を指定すると、その次に多くの情報が生成され、以下情報量が減少していきます。このプロパティのデフォルト値は info です。
- `xml.validation.mode`: XML パーサーの妥当性チェック・モードを設定します。有効な値は、true または false です。デフォルト値は false です。
- `autoreload.transformers`: トランスフォーマの変更を自動的に検出および再ロードするかどうかを指定します。有効な値は、true または false です。デフォルト値は true です。
- `autoreload.devices`: デバイスの変更を自動的に検出および再ロードするかどうかを指定します。有効な値は、true または false です。デフォルト値は true です。
- `long.url.format`: 埋込み URL のリライトに、URL の詳細フォーマットまたは簡略フォーマットのいずれを使用するかを指定します。有効な値は、true または false です。デフォルト値は true です。

> **関連項目：** URL の詳細フォーマットと簡略フォーマットの説明と比較は、3-7 ページの「MCSLite の URL リライティングとキャッシュ」を参照してください。

デバイス記述

OracleAS Wireless サーバーでは、すべてのデバイス記述がデータベースに格納されます。ただし、MCSLite を簡素化するために、データベース接続は不要で、各デバイス記述は XML ファイルに格納されます。デバイスの XML ファイルは、`[ORACLE_HOME]/wireless/wdk/server/applications/wdk/wdk-web/repository` にあります。各 XML ファイルには、単一のデバイスの属性と特性を記述するメタデータが含まれます。

次に、MCSLite が必要とする重要なデバイス・プロパティの一部を示します。

- Name: デバイスの一意の名前。
- UserAgents: デバイスのメタデータには、複数のユーザー・エージェントの値が含まれている場合があります。これは、デバイスのメタデータが複数の物理デバイスと一致する可能性があることを意味します。ユーザー・エージェントはデバイスの検出に使用されます。
- Transformers: 特定のデバイスに使用する必要があるトランスフォーマの名前が含まれます。マークアップ言語ごとに 1 つずつ、複数のトランスフォーマ値があります。
- DefaultMarkupLanguage: デバイスの MIME タイプ。Accept-Charset とともに、デバイスに送信されるレスポンスのコンテンツ・タイプを構成します。
- Accept-Charset: デバイスのエンコーディング。DefaultMarkupLanguage とともに、デバイスに送信されるレスポンスのコンテンツ・タイプを構成します。
- DeviceClass: デバイスのクラス（microbrowser、pdabrowser、pcbrowser、voice、micromessenger など）。
- DeviceHeight と DeviceWidth: デバイス画面の高さと幅。

> **関連項目：** デバイスのプロパティの詳細は、第 9 章「マルチチャネル・サーバーの使用」を参照してください。

デバイスの検出

MCSLite では、OracleAS Wireless サーバーと同じ高度なデバイス検出アルゴリズムが使用されます。

> **関連項目：** デバイスのプロパティの詳細は、第 9 章「マルチチャネル・サーバーの使用」を参照してください。

マルチメディア調整

MCSLite でサポートされるマルチメディア調整は、OracleAS Wireless サーバーの場合と同じです。ただし、MCSLite のマルチメディア調整では、マルチメディア調整インタフェースの独自の実装をプラグインするための拡張可能なフレームワークは提供されません。

> **関連項目：** マルチメディア調整の詳細は、第 9 章「マルチチャネル・サーバーの使用」を参照してください。

ロケーション・サービス

アプリケーション開発者は、優れたロケーション・サービスを新たに作成するか、またはロケーション情報を使用して既存のアプリケーションを拡張できます。このためには、次の専用サービスが必要です。

- モバイル・ポジショニング : アプリケーションで、ユーザーの現在のロケーションを取得および設定できます。
- 次の種類のロケーション・サービスがあります。
  - ジオコーディング : 住所または固定回線の電話番号の地理的位置を検索します。また、地理的位置を住所または電話番号に関連付ける逆の機能もあります。
  - マッピング : ロケーション周辺地域のマップ・イメージ、一連のロケーションをカバーするマップ、ルートのマップなどを取得します。
  - 運転方向 : 2 つの住所またはロケーションの間の運転方向を取得します。
  - ビジネス・ディレクトリ : 住所やロケーション周辺のビジネス、都市、州または国などのビジネスを検出します。

OracleAS Wireless WDK には、これらのロケーション・サービスのコンポーネントにアクセスするための Application Program Interface（API）が用意されています。開発者は、これらの API から OracleAS Wireless Web サービスにアクセスして、関連するコンテンツ・サービス・プロバイダとの完全な OracleAS Wireless 環境を設定することなく、アプリケーションを開発、デバッグおよびテストできます。

次の各項では、重要な API コールの一部を、いくつかの例に沿って簡単に説明します。API の詳細は、WDK とともに提供される Javadoc を参照してください。

モバイル・ポジショニング

モバイル・ポジショニング・サービスを使用すると、モバイル・ユーザーの現在のロケーションを取得および設定できます。このサービスは、OracleAS Wireless WDK で Web サービスとして実装されています。アプリケーションでは、インターネット上のどこからでも任意のプログラミング・モデルを使用して、モバイル・ユーザーの現在のロケーションを取得および設定できます。

oracle.panama.mp.soap.MPSoapClient クラスによって、クライアントの SOAP コールがラップされ、Java プログラミング・インタフェースでサービスが公開されます。クライアントの Java プログラムでは、モバイル・ユーザーのロケーションを取得および設定する前に、Web サービスの URL とサービス ID を使用してこのクラスのオブジェクトを最初に作成しておく必要があります。Web サービスの URL は、OracleAS Wireless サーバー上のロケーション Web サービスの SOAP ルーターで、たとえば、*http://myaswserver.oracle.com:7777/location/web services* です。サービス ID は SOAP のサービス ID で、たとえば *urn:MobilePositionServer* です。

- ロケーションの取得

  モバイル・ユーザーのロケーションを取得するには、次の 2 つのメソッドを使用できます。

  - `getPositionSimple(String username, String password, String msid)`
  - `getPosition(String username, String password, String msid, boolean getLatestLocationOnly)`

  getPositionSimple メソッドのパラメータ *username* と *password* は、認証目的で使用されます。3 番目のパラメータ *MSID* は、ロケーション取得がリクエストされているユーザーのモバイル・ステーション ID です。MSID は通常、ユーザーの携帯電話の番号です。リクエストが成功すると、このファンクションによって、緯度と経度を表す 2 つの倍精度数の配列が戻されます。

  `getPosition` メソッドの最初の 3 つのパラメータは `getPositionSimple` メソッドの場合と同じです。最後のパラメータは、コール元がモバイル・ステーションの認識済最新ロケーションの取得を要求しているかどうかを示すブール値です。true に設定されている場合、ポジショニング操作は実行されずに、OracleAS Wireless サーバーからモバイル・ステーションの認識済最新ロケーションが戻されます。キャッシュされているロケーションが OracleAS Wireless サーバーに存在しない場合は例外が発生します。false に設定されている場合は、経過期間がシステムのデフォルトのサービス品質の範囲内であるかぎり、キャッシュされているロケーションが OracleAS Wireless サーバーから戻されます。それ以外の場合は、OracleAS Wireless サーバーによってモバイル・ポジショニング操作が実行され、モバイル・ステーションの現在のロケーションが取得されます。`getPosition` メソッドの戻り値はユーザーのロケーションが（XML フォーマットで）記述された文字列で、次のようなスキーマです。

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
elementFormDefault="qualified">
<xsd:element name="RESPONSE ">
<xsd:complexType>
<xsd:sequence>
<xsd:element ref="TIMESTAMP"/>
<xsd:element ref="POS"/>
<xsd:element ref="VELOCITY" minOccurs="0"/>
<xsd:element ref="BEARING" minOccurs="0"/>
```

```
<xsd:element ref="ALTITUDE" minOccurs="0"/>
<xsd:element ref="ALT_UNCERTAINTY" minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="ALTITUDE" type="xsd:string"/>
<xsd:element name="ALT_UNCERTAINTY" type="xsd:string"/>
<xsd:element name="BEARING" type="xsd:string"/>
<xsd:element name="LAT" type="xsd:string"/>
<xsd:element name="LONG" type="xsd:string"/>
<xsd:element name="POS">
<xsd:complexType>
<xsd:sequence>
<xsd:element ref="LONG"/>
<xsd:element ref="LAT"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="TIMESTAMP" type="xsd:string"/>
<xsd:element name="VELOCITY" type="xsd:string"/>
</xsd:schema>
```

- <POS> 要素には、ロケーションの緯度と経度が含まれます。
- <TIMESTAMP> 要素には、ロケーション取得時のタイムスタンプが含まれます。時間には、常にグリニッジ標準時が使用されます。たとえば、2003-03-12 20:01:06 GMT のように記述されます。
- オプションの <VELOCITY> 要素では、モバイル・デバイスの速度（m/ 秒）が指定されます。
- オプションの <BEARING> 要素では、北からの時計回りで方位角（緯度）が指定されます。
- オプションの <ALTITUDE> 要素では、モバイル・デバイスの海抜高度（m）が指定されます。

username パラメータと password パラメータで識別されるコール元は、有効な OracleAS Wireless ユーザーであり、MSID に関連付けられているユーザーのロケーションにアクセスするためのロケーション認可が付与されている必要があります。username と password によってコール元を認証できない場合、またはコール元がロケーション情報へのアクセスを許可されていない場合は、例外が発生します。

- ロケーションの設定

  モバイル・デバイスは、通常、汎地球測位システム（GPS）によって提供される現在のロケーションを OracleAS Wireless サーバーに送信できます。送信された現在のロケーションはサーバーにキャッシュされ、モバイル・ポジショニング API とプライバシ API を使用して問い合せることができます。デバイスの現在のロケーションを OracleAS

Wireless サーバーに転送するクライアント・アプリケーション・プログラムを作成する必要があります。Java クライアントは、oracle.panama.mp.soap.MPSoapClient クラスの setPosition(String xmlReq) メソッドをコールできます。このファンクションは、位置データを表す 1 つの String パラメータを取ります。データは XML フォーマットで、次のスキーマに準拠している必要があります。

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
elementFormDefault="qualified">
<xsd:element name="MP_GPS">
<xsd:complexType>
<xsd:sequence>
<xsd:element ref="USERNAME"/>
<xsd:element ref="PASSWORD"/>
<xsd:element ref="MSID"/>
<xsd:element ref="TIME" minOccurs="0"/>
<xsd:element ref="GMT" minOccurs="0"/>
<xsd:element ref="POS"/>
<xsd:element ref="ALTITUDE" minOccurs="0"/>
<xsd:element ref="ALT_UNCERTAINTY" minOccurs="0"/>
<xsd:element ref="VELOCITY" minOccurs="0"/>
<xsd:element ref="BEARING" minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="ALTITUDE" type="xsd:string"/>
<xsd:element name="ALT_UNCERTAINTY" type="xsd:string"/>
<xsd:element name="BEARING" type="xsd:string"/>
<xsd:element name="GMT" type="xsd:string"/>
<xsd:element name="LAT" type="xsd:string"/>
<xsd:element name="LONG" type="xsd:string"/>
<xsd:element name="MSID" type="xsd:string"/>
<xsd:element name="PASSWORD" type="xsd:string"/>
<xsd:element name="POS">
<xsd:complexType>
<xsd:sequence>
<xsd:element ref="LAT"/>
<xsd:element ref="LONG"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="TIME" type="xsd:string"/>
<xsd:element name="USERNAME" type="xsd:string"/>
<xsd:element name="VELOCITY" type="xsd:string"/>
</xsd:schema>
```

- <USERNAME> および <PASSWORD> 要素は、リクエストを許可するために OracleAS Wireless サーバーによって使用されます。
- <MSID> 要素は、モバイル・デバイスまたはユーザーのモバイル・ステーション ID です。
- オプションの <TIME> 要素は、このロケーションが GPS によって生成された時間を示します。この値が不明な場合は、OracleAS Wireless サーバーがデータを受信した時間が使用されます。
- オプションの <VELOCITY> 要素では、モバイル・デバイスの速度（m/ 秒）が指定されます。
- オプションの <BEARING> 要素では、北からの時計回りで方位角（緯度）が指定されます。
- オプションの <ALTITUDE> 要素では、モバイル・デバイスの海抜高度（m）が指定されます。

■ モバイル・ポジショニングの例

次に、SOAP クライアントを使用して OracleAS Wireless サーバーに対する位置を設定および取得する方法の例を示します。

```
MPSoapClient mpsc = new
MPSoapClient("http://usunnab16.us.oracle.com:5555/location/webservices",
"urn:MobilePositionServer");

String xmlReq = "<?xml version= '1.0' encoding='ISO-8859-1'
standalone='yes'?>¥n" +
                "<MP_GPS>¥n" +
                "<MSID>6038973096</MSID>¥n" +
                "<POS>¥n" +
                "<LAT>42.1576</LAT>¥n" +
                "<LONG>-122.34</LONG>¥n" +
                "</POS>¥n"+
                "</MP_GPS>";
    System.out.println(mpsc.setPosition(xmlReq));

// NOTE: Need to change getPosition call//
    double[] ret = mpsc.getPositionSimple("", "", "6038973096");
    System.out.println(ret[0] + "," + ret[1]);
```

ロケーション・サービス クライアント側の LBS Web サービス API は、実質的には OracleAS Wireless サーバー内の API と同じです。次のいくつかの例で、これについて説明します。最も問題とされる違いは、set() ファンクションではサーバーの指定が必要なことです。OracleAS Wireless サーバー内のアプリケーションでは、このようなターゲット・サーバーの指定は必要ありません。

ジオコーディング

ジオコーディングでは次の機能がサポートされます。

- ジオコーディング : 地理上の座標（緯度と経度）を住所に割り当てます。
- 一括ジオコーディング : 広範囲にわたる住所の集合に対するジオコーディングを 1 回の操作で実行します。
- 逆ジオコーディング : 住所を地理上の座標（緯度と経度）に割り当てます。
- 一括逆ジオコーディング : 一連の住所を地理上の座標（緯度と経度）に割り当てます。
- 電話番号ジオコーディング : 住所を固定電話番号に割り当てます。
- 一括電話番号ジオコーディング
- 逆電話番号ジオコーディング : 一連の電話番号を住所に割り当てます。
- 一括逆電話番号ジオコーディング

次に、ジオコーディング操作の例を示します。

```
...
  public static void main(String args[])
  {
    SpatialManager.set(
      "http://mhorhamm-us.us.oracle.com:9000/studio/soap/servlet/soaprouter",
      "your username",
      "your password");

    Location locs[] =
      SpatialManager.getGeocoder().geocodeAddress(
        new LocationImpl(
          new PointImpl(0, 0),
          "Oracle",
          "1",
          new String[] { "Oracle Dr" },
          "",
          "Nashua",
          "NH",
          "03062",
          "",
          "US"),
        "");
    if(locs != null)
    {
      for(int i = 0; i < locs.length; i++)
        System.out.println(i + ": " + locs[i]);
    }
```

```
    else
      System.out.println("null");


  }
```

マッピング

マッピングでは次の機能がサポートされます。

- ラベル付き地点とマーク付き地点を含むまたは含まない地図を表示します。
- 全運転方向または 1 つの経路の地図を表示します（運転方向サービス内の同じ機能）。

次に、マーク付き地点が 1 つあるマッピングの例を示します。

```
...
  public static void main(String args[])
  {
    SpatialManager.set(
      "http://mhorhamm-us.us.oracle.com:9000/studio/soap/servlet/soaprouter",
      "your username",
      "your password");

    Location
      adr1[] =
        SpatialManager.getGeocoder().geocodeAddress(
          new LocationImpl(
            new PointImpl(0, 0),
            "NEDC",
            "1",
            new String[] { "Oracle Dr" },
            "",
            "Nashua",
            "NH",
            "03062",
            "",
            "US"),
          ""),
      adr2[] =
        SpatialManager.getGeocoder().geocodeAddress(
          new LocationImpl(
            new PointImpl(0, 0),
            "HQ",
            "500",
            new String[] { "Oracle Parkway" },
            "",
```

```
            "Redwood City",
            "CA",
            "94065",
            "",
            "US"),
          "");


    System.out.println(
      SpatialManager.getMapper().getMapURL(
        adr1[0],
        ImageFormats.GIF,
        800,
        600,
        false));

    ...
  }
...
```

運転方向

運転方向のコンポーネントでは、次の機能がサポートされます。

- 住所または地理的地点の座標間の運転方向を決定します。
- 全方向または 1 つの経路の地図を表示します
- 運転方向の簡易ジオメトリを提供します。
- 要求されたルートの方向、距離、所要時間をテキストで提供します。

次の例は、2 つの住所間の運転方向を示しています。

```
...
  public static void main(String args[])
  {
    SpatialManager.set(
      "http://mhorhamm-us.us.oracle.com:9000/studio/soap/servlet/soaprouter",
      "your username",
      "your password");

    Location
      adr1[] =
        SpatialManager.getGeocoder().geocodeAddress(
          new LocationImpl(
            new PointImpl(0, 0),
            "NEDC",
            "1",
            new String[] { "Oracle Dr" },
```

```
          "",
          "Nashua",
          "NH",
          "03062",
          "",
          "US"),
        "");
    adr2[] =
      SpatialManager.getGeocoder().geocodeAddress(
        new LocationImpl(
          new PointImpl(0, 0),
          "HQ",
          "500",
          new String[] { "Oracle Parkway" },
          "",
          "Redwood City",
          "CA",
          "94065",
          "",
          "US"),
        "");

  ////////////////////////////////////////////////////////////////

  RoutingSettings settings = new RoutingSettings(true, false);
  settings.setSecondaryOption(RoutingOption.maneuverMapWidth,   "800");
  settings.setSecondaryOption(RoutingOption.maneuverMapHeight,  "600");
  settings.setSecondaryOption(RoutingOption.overviewMapWidth,   "800");
  settings.setSecondaryOption(RoutingOption.overviewMapHeight,  "600");

  System.out.println(
    SpatialManager.getRouter().computeRoute(
      adr1[0],
      adr2[0],
      null,
      settings,
      Locale.US));

}
```

ビジネス・ディレクトリ

ビジネス・ディレクトリのコンポーネントでは、次の機能がサポートされます。

- 市、州、郵便番号内のビジネスの検索、住所または地理的位置から離れた場所、運転方向ルートから離れた場所のビジネスの検索、住所または地理的位置の周辺にある直近の *n* ビジネスのセット内のビジネスの検索
- 指定した名前によるビジネスの検索、指定したカテゴリ内のビジネスの検索、指定したキーワード（名前やカテゴリ）に一致するビジネスの検索、指定した名前とカテゴリに一致するビジネスの検索
- カテゴリ階層の検索とナビゲート

次に、指定した名前を持つ複数のカテゴリとビジネスの例を示します。

```
  public static void main(String args[])
  {
    SpatialManager.set(
      "http://mhorhamm-us.us.oracle.com:9000/studio/soap/servlet/soaprouter",
      "your username",
      "your password");

    ////////////////////////////////////////////////////////////////

    System.out.println(SpatialManager.getYPFinder().getCategoryAtRoot());
    System.out.println(SpatialManager.getYPFinder().getCategoryAtPath(new
String[0]));
    YPCategory cats[] =
      SpatialManager.getYPFinder().getCategoryAtRoot().getSubCategories();
    if(cats != null)
      for(int i = 0; i < cats.length; i++)
        System.out.println(i + ": " + cats[i]);

    ////////////////////////////////////////////////////////////////

    YPBusiness b[] =
      SpatialManager.getYPFinder().getBusinessesAnywhere(
        "Border",
        Locale.US);
    for(int i = 0; i < b.length; i++)
      System.out.println(i + ": " + b[i]);


  }
```

WDK ログ・ファイル

OracleAS Wireless WDK には、Wireless アプリケーションの開発に役立つログ・ファイルが用意されています。この項では、WDK ログ・ファイルに記録される情報を詳細に説明します。

ログ・ファイルに記録される情報の量は、WEB-INF/web.xml 内の wdk.log.level パラメータを変更することによって構成できます。このパラメータの有効な値は、情報量の最も多いレベルから最も少ないレベルまで順に、debug、info、warning および error です。このパラメータのデフォルト値は info です。このレベルでは、次の種類の情報がログ・ファイルに記録されます。

- リクエスト URL: 現在処理中のリクエストに関連付けられている URL 文字列および問合せパラメータです。たとえば、http://myhost:80/myapp/foo.jsp?param1=value1 などです。
- クライアントから受信したリクエスト HTTP ヘッダー : クライアント（デバイス）から受信したすべてのリクエスト HTTP ヘッダーがリストされます。このリストの重要なヘッダーの 1 つは user-agent ヘッダーで、その値はデバイス検出に使用されます。
- 検出されたデバイス : このデバイスが、user-agent ヘッダーに基づいて検出されたデバイスであることをユーザーに通知します。この情報は、問題が生じた場合に確認する必要があります。
- バックエンド・アプリケーションに送信したリクエスト HTTP ヘッダー : バックエンド・アプリケーションに送信したすべてのリクエスト HTTP ヘッダーがリストされます。仮想ブラウザとして、WDK は元のヘッダーの一部を変更し、別のヘッダーを追加します。
- バックエンド・アプリケーションから受信したレスポンス HTTP ヘッダー : バックエンド・アプリケーションから受信したすべてのレスポンス HTTP ヘッダーがリストされます。これらのヘッダーは WDK によってレスポンスの処理に使用され、クライアント（デバイス）には送信されないことに注意してください。
- バックエンド・アプリケーションからの XML: バックエンド・アプリケーションからの正確な XML 文字列のレスポンスを示します。
- XML のタイプとバージョン : WDK は、XML のタイプ（Simple Result、XHTML+XFORMS または XHTML-MP）とバージョンを検出しようとします。検出結果が、ログのこのセクションに示されます。この情報に基づいて、WDK は使用する適切なトランスフォーマを選択します。
- WDK レスポンス : クライアント（デバイス）に送信した最終的なマークアップ言語と HTTP ヘッダーを示します。この最終的な結果は、バックエンド・アプリケーションから受信した XML レスポンスにトランスフォーマ（XSLT または Java トランスフォーマ）を適用することによって取得されます。

debug ログ・レベルが使用されると、さらに詳細なログ・メッセージが生成されます。デバッグ・メッセージは次のとおりです。

- トランスフォーマのロード : 成功したトランスフォーマのロードを示します。
- デバイスのロード : 成功したデバイスのロードを示します。
- デバイスとトランスフォーマのマッピング : 各デバイスに複数のトランスフォーマが関連付けられています（各マークアップ・タイプに対して 1 つずつ）。このデバッグ・メッセージでは、デバイスとトランスフォーマ間のマッピングが示されます。
- バックエンド URL: このリクエストのバックエンド URL を示します。
- バックエンド・レスポンス・コードとレスポンス・メッセージ : このリクエストのバックエンド・レスポンス・コードとレスポンス・メッセージを示します。
- トランスフォーマに渡された XML: トランスフォーマに渡された XML 文字列を示します。この XML 文字列は、バックエンド・アプリケーションから受信した元の XML 文字列とは異なります。この文字列は注釈からの結果である中間の形式で、トランスフォーマに対する入力として使用されます。

WDK ログのサンプル

次の例は、debug ログ・レベルの WDK ログ・ファイルからの抜粋で、UP Simulator 4.1.1 を使用した 1 回のリクエスト / レスポンス型サイクルの成功例です。

ヘッダーには、次の情報が含まれます。

- クライアントの IP アドレス : `127.0.0.1`
- セッション ID: `fe88712959d4470794599b62102e61df`
- ログ・レベル : `INFO`
- タイムスタンプ : `[Fri, 23 May 2003 10:41:11 PDT]`

```
127.0.0.1 - - fe88712959d4470794599b62102e61df - - INFO : [Fri, 23 May 2003 10:41:11 PDT]
*************** Start of serving request ***************

127.0.0.1 - - fe88712959d4470794599b62102e61df - - INFO : [Fri, 23 May 2003 10:41:11 PDT]
Request URL:
http://localhost:9010/wdk/mcslite/

127.0.0.1 - - fe88712959d4470794599b62102e61df - - INFO : [Fri, 23 May 2003 10:41:11 PDT]
Request HTTP headers received from client:
user-agent: OWG1 UP/4.1.20a UP.Browser/4.1.20a-XXXX UP.Link/4.1.HTTP-DIRECT
x-upfax-accepts: none
x-up-devcap-max-pdu: 2984
x-up-devcap-iscolor: 0
x-up-devcap-numsoftkeys: 2
accept: application/x-hdmlc, application/x-up-alert, application/x-up-cacheop, application/x-up-device,
application/x-up-digestentry, application/vnd.wap.wml, text/x-wap.wml, text/vnd.wap.wml,
application/vnd.wap.wmlscript, text/vnd.wap.wmlscript, application/vnd.uplanet.channel,
application/vnd.uplanet.list, text/x-hdml, text/plain, image/vnd.wap.wbmp, image/bmp,
```

```
application/remote-printing text/x-hdml;version=3.1, text/x-hdml;version=3.0, text/x-hdml;version=2.0,
image/bmp, text/html
x-up-devcap-smartdialing: 1
x-up-devcap-msize: 8,18
accept-charset: ISO-8859-1, UTF-8, *
x-up-devcap-screenpixels: 171,108
host: localhost:9010
accept-language: en
x-up-devcap-screendepth: 1
content-type: application/x-www-form-urlencoded
x-up-devcap-charset: ISO-8859-1
x-up-subno: rhalimma_st3010pc
cookie: JSESSIONID=fe88712959d4470794599b62102e61df
x-up-devcap-immed-alert: 1


127.0.0.1 - - fe88712959d4470794599b62102e61df - - INFO : [Fri, 23 May 2003 10:41:11 PDT]
Request HTTP headers sent to back end application:
x-oracle-user.location.addresslastline: Room 200
x-oracle-service.home.url: http://localhost:9000/omsdk/rm
x-up-devcap-screendepth: 1
host: localhost:9010
x-up-devcap-numsoftkeys: 2
x-oracle-user.deviceid: 1234
x-oracle-orig-user-agent: OWG1 UP/4.1.20a UP.Browser/4.1.20a-XXXX UP.Link/4.1.HTTP-DIRECT
accept: application/vnd.oracle.xhtml+xforms, text/vnd.oracle.mobilexml, application/vnd.wap.xhtml+xml,
application/xhtml+xml;profile="http://xmlns.oracle.com/ias/dtds/xhtml+xforms",
application/xhtml+xml;profile="http://www.wapforum.org/xhtml", application/xhtml+xml, application/xml,
text/xml, application/vnd.oracle.xad, */*, */*
x-oracle-service.parent.url: http://localhost:9000/omsdk/rm
x-oracle-user.location.addressline2: Apt# 1004
x-oracle-user.location.addressline1: 1007 Broadway St
x-oracle-user.location.block: Block A
x-oracle-user.locale: US
x-oracle-user.authkind: unauthenticated
x-oracle-user.displayname: Jon Smith
x-oracle-user.location.type: profile
x-up-devcap-max-pdu: 2984
x-oracle-user.userkind: registered
x-up-devcap-iscolor: 0
x-up-devcap-screenpixels: 171,108
x-oracle-mcs.character.encoding: UTF-8
x-oracle-user.location.postalcodeext: 3158
accept-charset: ISO-8859-1, UTF-8, *
accept-charset: UTF-8, *
x-oracle-user.location.companyname: Company XYZ
x-oracle-home.url: http://localhost:9000/omsdk/rm
cookie: JSESSIONID=fe88712959d4470794599b62102e61df
x-up-devcap-immed-alert: 1
```

```
x-oracle-module.callback.url: http://localhost:9000/omsdk/rm
x-upfax-accepts: none
x-up-devcap-smartdialing: 1
user-agent: PTG/2.0 (Oracle9iAS Wireless 9.0.4.0; media="handheld"; paged="1")
x-oracle-user.location.postalcode: 94104
x-up-devcap-msize: 8,18
x-oracle-user.location.city: San Francisco
x-oracle-user.location.country: USA
content-type: application/x-www-form-urlencoded
x-oracle-user.name: jsmith
x-oracle-user.location.y: 200.8
x-oracle-user.location.x: 135.9
x-oracle-user.location.county: San Francisco
x-oracle-orig-accept: application/x-hdmlc, application/x-up-alert, application/x-up-cacheop,
application/x-up-device, application/x-up-digestentry, application/vnd.wap.wml, text/x-wap.wml,
text/vnd.wap.wml, application/vnd.wap.wmlscript, text/vnd.wap.wmlscript, application/vnd.uplanet.channel,
application/vnd.uplanet.list, text/x-hdml, text/plain, image/vnd.wap.wbmp, image/bmp,
application/remote-printing text/x-hdml;version=3.1, text/x-hdml;version=3.0, text/x-hdml;version=2.0,
image/bmp, text/html
x-up-subno: rhalimma_st3010pc
accept-language: en
x-up-devcap-charset: ISO-8859-1
x-oracle-module.callback.label: Home
x-oracle-user.location.state: CA


127.0.0.1 - - fe88712959d4470794599b62102e61df - - DEBUG : [Fri, 23 May 2003 10:41:11 PDT]
Back end URL: http://localhost:9010/mcs/examples/index.jsp

127.0.0.1 - - fe88712959d4470794599b62102e61df - - DEBUG : [Fri, 23 May 2003 10:41:11 PDT]
Back end response code: 200 ; response message: OK

127.0.0.1 - - fe88712959d4470794599b62102e61df - - INFO : [Fri, 23 May 2003 10:41:11 PDT]
Response HTTP headers received from back end application:
x-oracle-wireless.referer.url: http://localhost:9010/mcs/examples/index.jsp
content-type: application/vnd.oracle.xhtml+xforms; charset=UTF-8
x-oracle-wireless.base.url: http://localhost:9010/mcs/examples/index.jsp
connection: Close
date: Fri, 23 May 2003 17:41:10 GMT
server: Oracle9iAS (9.0.3.0.0) Containers for J2EE
content-length: 994


127.0.0.1 - - fe88712959d4470794599b62102e61df - - INFO : [Fri, 23 May 2003 10:41:11 PDT]
XML Result from backend:
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>

<html profile="http://xmlns.oracle.com/ias/dtds/xhtml+xforms/0.9.0/1.0"
      xmlns="http://www.w3.org/1999/xhtml"
      xmlns:style="urn:oracle:iasw-internal:style.1.0"
```

```
     xmlns:extra="urn:oracle:iasw-internal:mxml.1.0">
  <head>
     <title>Oracle9iAS Wireless Examples</title>
     <style type="text/css">
         .title {font-style: italic; color: blue; font-size: xx-large}
         .menu {font-style: italic; color: blue; font-size: x-large}
         li {font-weight: bold; color: blue}
     </style>
  </head>
  <body>
    <nl style="list-style-type: decimal">
      <label class="title">Oracle9iAS Wireless Examples</label>
      <li class="menu" href="xhtml%2Bxforms/index.jsp">XHTML+XFroms Examples</li>
      <li class="menu" href="xhtml%2Bmp/index.jsp">XHTML MP Examples</li>
      <li class="menu" href="mobile-xml/index.jsp">MobileXML Examples</li>
    </nl>
  </body>
</html>

127.0.0.1 - - fe88712959d4470794599b62102e61df - - INFO : [Fri, 23 May 2003 10:41:11 PDT]
XML content info - Type: XHTML, version: 0.9.0

127.0.0.1 - - fe88712959d4470794599b62102e61df - - INFO : [Fri, 23 May 2003 10:41:11 PDT]
Transformer that will be used: xforms-wml11-openwave

127.0.0.1 - - fe88712959d4470794599b62102e61df - - DEBUG : [Fri, 23 May 2003 10:41:11 PDT]
The XML passed to transformer:
<html profile="http://xmlns.oracle.com/ias/dtds/xhtml+xforms/0.9.0/1.0" xmlns="http://www.w3.org/1999/xhtml"
xmlns:style="urn:oracle:iasw-internal:style.1.0" xmlns:extra="urn:oracle:iasw-internal:mxml.1.0"
xmlns:mxml="http://xmlns.oracle.com/2002/MobileXML" style:mheight="0mm" style:mwidth="0mm"
style:word-spacing="normal" style:padding-top="0" style:text-align="justify" style:border-top-color="#000000"
style:border-right-style="none" style:font-size="medium" style:padding-bottom="0" style:margin-right="0"
style:list-style-type="disc" style:vertical-align="baseline" style:border-bottom-color="#000000"
style:pause-after="none" style:width="0px" style:speech-rate="default" style:border-left-width="medium"
style:speak="normal" style:float="none" style:text-decoration="none" style:padding-right="0"
style:border-right-color="#000000" style:list-style-image="none" style:background-attachment="scroll"
style:clear="none" style:stress="none" style:font-family="serif,san-serif" style:margin-top="0"
style:letter-spacing="normal" style:font-variant="normal" style:border-top-width="medium"
style:margin-bottom="0" style:border-left-style="none" style:speak-numeral="none"
style:background-image="none" style:pause-before="none" style:volume="default"
style:border-bottom-width="medium" style:pitch="default" style:text-transform="none"
style:list-style-position="outside" style:padding-left="0" style:margin-left="0"
style:border-right-width="medium" style:color="#000000" style:text-indent="0" style:border-top-style="none"
style:border-left-color="#000000" style:height="0px" style:font-weight="400" style:background-repeat="repeat"
style:font-style="normal" style:pitch-range="default" style:border-bottom-style="none"
style:voice-family="neutral" style:speak-header="once" style:display="inline"
extra:iaswhref="http://localhost:9010/wdk/mcslite/?PAckey=4!" extra:newdoc="true"><extra:param
extra:name="PAckey" extra:value="4!" extra:hidden="true"/>
  <head style:border-left-width="medium" style:border-left-style="none" style:border-left-color="#000000"
```

```
style:display="inline"><extra:messages/><extra:patparams/>
     <title style:border-left-width="medium" style:border-left-style="none" style:border-left-color="#000000"
style:display="inline">Oracle9iAS Wireless Examples</title>
     <style type="text/css" style:border-left-width="medium" style:border-left-style="none"
style:border-left-color="#000000" style:display="inline">
         .title {font-style: italic; color: blue; font-size: xx-large}
         .menu {font-style: italic; color: blue; font-size: x-large}
         li {font-weight: bold; color: blue}
     </style>
  <extra:displaypage page="1" deck="1"/></head>
  <body __length__="13" style:border-left-width="medium" style:border-left-style="none"
style:border-left-color="#000000" style:display="block" extra:emwidth="0" extra:pxwidth="0"
extra:emheight="0" extra:pxheight="0" extra:random="30274" extra:softkeys="2" extra:paged="true"><extra:page
page="1" pagelength="342" deck="1" extra:expand="true">
    <nl style="list-style-type: decimal" __length__="70" style:white-space="nowrap"
style:list-style-type="decimal" style:border-left-width="medium" style:margin-top="0.5em"
style:margin-bottom="0.5em" style:border-left-style="none" style:border-left-color="#000000"
style:display="block" extra:uid="XF1" extra:expand="true">
      <label class="title" __length__="53" style:border-top-color="#0000ff" style:font-size="xx-large"
style:border-bottom-color="#0000ff" style:border-left-width="medium" style:border-right-color="#0000ff"
style:margin-top="0" style:margin-bottom="0" style:border-left-style="none" style:color="#0000ff"
style:border-left-color="#0000ff" style:font-style="italic" style:display="inline">Oracle9iAS Wireless
Examples</label>
      <li class="menu" href="xhtml%2Bxforms/index.jsp" __length__="78"
extra:abshref="http://localhost:9010/mcs/examples/xhtml%2Bxforms/index.jsp"
extra:iaswhref="/wdk/mcslite?PAckey=5!" extra:iaswphref="/wdk/mcslite?PAckey=%PAckey!" extra:iaswpkhref="5"
style:border-top-color="#0000ff" style:font-size="x-large" style:border-bottom-color="#0000ff"
style:border-left-width="medium" style:text-decoration="underline" style:border-right-color="#0000ff"
style:margin-top="0" style:margin-bottom="0" style:border-left-style="none" style:color="#0000ff"
style:border-left-color="#0000ff" style:font-weight="700" style:font-style="italic"
style:display="list-item">XHTML+XFroms Examples</li>
      <li class="menu" href="xhtml%2Bmp/index.jsp" __length__="70"
extra:abshref="http://localhost:9010/mcs/examples/xhtml%2Bmp/index.jsp"
extra:iaswhref="/wdk/mcslite?PAckey=6!" extra:iaswphref="/wdk/mcslite?PAckey=%PAckey!" extra:iaswpkhref="6"
style:border-top-color="#0000ff" style:font-size="x-large" style:border-bottom-color="#0000ff"
style:border-left-width="medium" style:text-decoration="underline" style:border-right-color="#0000ff"
style:margin-top="0" style:margin-bottom="0" style:border-left-style="none" style:color="#0000ff"
style:border-left-color="#0000ff" style:font-weight="700" style:font-style="italic"
style:display="list-item">XHTML MP Examples</li>
      <li class="menu" href="mobile-xml/index.jsp" __length__="71"
extra:abshref="http://localhost:9010/mcs/examples/mobile-xml/index.jsp"
extra:iaswhref="/wdk/mcslite?PAckey=7!" extra:iaswphref="/wdk/mcslite?PAckey=%PAckey!" extra:iaswpkhref="7"
style:border-top-color="#0000ff" style:font-size="x-large" style:border-bottom-color="#0000ff"
style:border-left-width="medium" style:text-decoration="underline" style:border-right-color="#0000ff"
style:margin-top="0" style:margin-bottom="0" style:border-left-style="none" style:color="#0000ff"
style:border-left-color="#0000ff" style:font-weight="700" style:font-style="italic"
style:display="list-item">MobileXML Examples</li>
    </nl>
  </extra:page></body>
</html>
```

```
127.0.0.1 - - fe88712959d4470794599b62102e61df - - INFO : [Fri, 23 May 2003 10:41:11 PDT]
Mobile WDK Response:
Content-Type: text/vnd.wap.wml; charset=UTF-8
Content-Length: 1384

<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE wml PUBLIC "-//PHONE.COM//DTD WML 1.1//EN"
"http://www.phone.com/dtd/wml11.dtd">
<wml><head><meta http-equiv="Cache-Control" forua="true" content="max-age=10"/></head><template><do
type="accept" name="do_accept" label="Ok"><refresh/></do></template><card><onevent type="onenterforward"><go
href="#ID117"/></onevent><onevent type="onenterbackward"><prev/></onevent></card><card id="ID117"
title="Oracle9iAS Wireless Examples"><p mode="nowrap"><big><i>Oracle9iAS Wireless Examples</i></big><select
name="mID117"><option value="5">XHTML+XFroms Examples</option><option value="6">XHTML MP
Examples</option><option value="7">MobileXML Examples</option><option title="Ok">[Back]<onevent
type="onpick"><prev/></onevent></option></select><do type="accept" name="do_accept" label="Ok"><go href="#_
nav"><setvar name="_h" value="$(mID117)"/></go></do></p></card><card id="_nav"
onenterforward="/wdk/mcslite?PAckey=$(_h)!"><onevent type="onenterbackward"><prev/></onevent></card><card
id="_form"><onevent type="onenterforward"><go method="post"
href="http://localhost:9010/wdk/mcslite/?r=30274"><postfield name="PAckey" value="4!"/><postfield name="$(_
sb)" value="$(_sv)"/><postfield name="PATpage" value="$(PATpage)"/><postfield name="PATdeck"
value="$(PATdeck)"/></go></onevent><onevent type="onenterbackward"><prev/></onevent></card></wml>


127.0.0.1 - - fe88712959d4470794599b62102e61df - - INFO : [Fri, 23 May 2003 10:41:11 PDT]
*************** End of serving request ***************
```

一般的な誤り

この項では、WDK で発生する一般的な誤りの一部を説明します。

- バックエンド・アプリケーションの URL の誤り。たとえば、バックエンド URL http://somehost:8080/myapp/first.jsp に基づいた正しい URL と誤りのある URL の例は、次のとおりです。

  正しい URL: http://localhost:9010/wdk/mcslite/http/somehost/9090/myapp/first.jsp

  誤りのある URL:
  http://localhost:9010/wdk/mcslite/http/somehost/9090/myap/first.jsp

  URL に誤りがあると、エラーの原因（この場合は 404 Not Found）を詳細に説明したエラー・ページが表示されます。

バックエンド URL の値は、ログ・ファイルの「Back end URL」の下の記述で確認できます。

ログ・エントリの例 :

```
127.0.0.1 - - 3b34912b68474bc1b1defa87e74dbd1e - - DEBUG : [Fri, 23 May 2003 11:06:45 PDT]
Back end URL: http://localhost:9010/mcs/examples/mobile-xmll/index.jsp

127.0.0.1 - - 3b34912b68474bc1b1defa87e74dbd1e - - DEBUG : [Fri, 23 May 2003 11:06:45 PDT]
Back end response code: 404 ; response message: Not Found

127.0.0.1 - - 3b34912b68474bc1b1defa87e74dbd1e - - ERROR : [Fri, 23 May 2003 11:06:45 PDT]
javax.servlet.ServletException:
HTTP(S) Error: 404 : Not Found
```

- バックエンド・アプリケーションからの content-type レスポンス・ヘッダーの誤り。たとえば、XForms ページを開発するときは、コンテンツ・タイプを application/vnd.oracle.xhtml+xforms に設定する必要があります。この値を設定しないと、WDK のコンテンツ検出ロジックでエラーが発生し、最終的なマークアップ言語が誤って生成されます。

  バックエンド・アプリケーションから受信したコンテンツ・タイプの値は、ログ・ファイルの「Response HTTP headers received from back-end application」の下の記述で確認できます。

ログ・エントリの例 :

```
127.0.0.1 - - 3b34912b68474bc1b1defa87e74dbd1e - - INFO : [Fri, 23 May 2003 10:59:27 PDT]
Response HTTP headers received from back end application:
x-oracle-wireless.referer.url: http://localhost:9010/mcs/examples/index.jsp
content-type: application/vnd.oracle.xhtml+xforms; charset=UTF-8
x-oracle-wireless.base.url: http://localhost:9010/mcs/examples/index.jsp
connection: Close
date: Fri, 23 May 2003 17:59:27 GMT
server: Oracle9iAS (9.0.3.0.0) Containers for J2EE
content-length: 994
```

- バックエンド・アプリケーションからの XML コンテンツの誤り。バックエンド・アプリケーションからの XML コンテンツは整形式の XML 文書であることが必要です。つづりの間違いまたは終了タグの不足の場合は、WDK からエラー・コード 500 が戻されます。この場合は、クライアント・デバイスにエラー・メッセージ（「XML を解析できませんでした。無効な XML が含まれています。」）が表示されます。

  バックエンド・アプリケーションからの XML コンテンツは、ログ・ファイルの「XML Result from backend」の下の記述で確認できます。

ログ・エントリの例：

```
127.0.0.1 - - 3b34912b68474bc1b1defa87e74dbd1e - - INFO : [Fri, 23 May 2003 11:00:50 PDT]
XML Result from backend:
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>

<html profile="http://xmlns.oracle.com/ias/dtds/xhtml+xforms/0.9.0/1.0"
      xmlns="http://www.w3.org/1999/xhtml"
      xmlns:style="urn:oracle:iasw-internal:style.1.0"
      xmlns:extra="urn:oracle:iasw-internal:mxml.1.0">
  <head>
     <title>Oracle9iAS Wireless Examples</title>
     <style type="text/css">
         .title {font-style: italic; color: blue; font-size: xx-large}
         .menu {font-style: italic; color: blue; font-size: x-large}
         li {font-weight: bold; color: blue}
     </style>
  </head>
  <body>
    <nl style="list-style-type: decimal">
      <label class="title">Oracle9iAS Wireless Examples</label>
      <li class="menu" href="xhtml%2Bxforms/index.jsp">XHTML+XFroms Examples</li>
      <li class="menu" href="xhtml%2Bmp/index.jsp">XHTML MP Examples</li>
      <li class="menu" href="mobile-xml/index.jsp">MobileXML Examples</li>
    </nl>
  <body>
</html>

127.0.0.1 - - 3b34912b68474bc1b1defa87e74dbd1e - - ERROR : [Fri, 23 May 2003 11:00:50 PDT]
oracle.wireless.sdk.SdkException: The xml could not be parsed. It contains invalid xml.
```

WDK チュートリアルを使用した Wireless アプリケーションの実行

このチュートリアルは、Wireless Developer Kit（WDK）を使用してラップトップやデスクトップで OracleAS Wireless 開発環境を設定するプロセスを、順を追って説明します。既存のアプリケーションをいくつか使用し、それらを独自の環境で実行します。

必要なもの

- モバイル・デバイス・シミュレータ
- Wireless Developer Kit（約 50MB）
- JDK

注意： WDK とシミュレータは、http://otn.oracle.com/tech/wireless/tools/content.html からダウンロードできます。

チュートリアルの概要

WDK は、OracleAS Wireless の小規模なランタイム環境です。Wireless アプリケーションを作成およびテストできるように、デバイス調整およびマルチメディア調整用に OracleAS Wireless のマルチチャネル・サーバーが組み込まれています。

環境の設定

次に、WDK 環境の設定手順を示します。

WDK 環境の設定

1. JDK 1.4.1 をダウンロードし、コンピュータにインストールします（インストール先の例 : D:¥j2sdk1.4.1_01）。JDK 1.4.1 を所有していない場合は、http://java.sun.com からダウンロードできます。このチュートリアルでは、JDK を D:¥j2sdk1.4.1_01 にインストールした場合を想定しています。
2. Wireless Developer Kit（WDK）をダウンロードして、マシンにインストールします。
   a. WDK をコンピュータ上の任意の場所で解凍します。WDK のパスに空白が含まれていないことを確認します。
   b. コンピュータに 2 つの環境変数を設定します。
      * IAS_HOME を D:¥wdk9041（WDK を解凍した場所）に設定します。
      * JAVA_HOME を D:¥j2sdk1.4.1_01（JDK 1.4.1 をインストールした場所）に設定します。

3. デバイス・シミュレータがない場合は、ダウンロードしてインストールします。インターネットにアクセスできる場合は、OTN: http://otn.oracle.com/tech/wireless/tools/content.html の Mobile Tech Center からデバイス・シミュレータを入手できます。

WDK の構成

次の WDK ファイルを変更する必要があります。

- `WDK_INSTALL_DIR/wireless/j2ee/config/oc4j.properties`
- `WDK_INSTALL_DIR/wireless/j2ee/config/global-web-application.xml`

エントリ *__REPLACE_WITH_IASW_HOME_PATH__* を WDK_INSTALL_DIR の絶対パスに置換する必要があります。ディレクトリのセパレータには、Microsoft Windows の場合でもスラッシュ（/）を使用します。たとえば、WDK が d:¥wdk にインストールされている場合は、*__REPLACE_WITH_IASW_HOME_PATH__* を d:¥wdk に置換します。

WDK の起動

1. `wdk.bat`（wdk9041/j2ee/home/wdk.bat にあります）を実行して、Wireless 環境を起動します。最初の起動時に、いくつかのファイルが解凍されます。起動時に、Containers for J2EE が初期化されていることを示すメッセージが表示されます。
2. 環境を停止するには、[Ctrl] キーを押しながら [C] を入力し、「Yes」と応答します。前述のステップ 1 に従って環境を再起動します。

Wireless Developer Kit に関しては、2 つの主な URL である、アプリケーションにアクセスするための URL とログを表示するための URL を認識しておく必要があります。

- アプリケーションにアクセスするための URL

http://<host_name>:9010/wdk/mcslite/http/<host_name>/9010/examples/<app_name>

たとえば、
http://localhost:9010/wdk/mcslite/http/localhost/9010/examples/mobile-xml/Hello.jsp
のようになります。この URL を使用すると、PC のブラウザまたはモバイル・デバイス・シミュレータから Hello World の例にアクセスできます。これは、WDK をテストするための簡単なデモです。

Hello World アプリケーションのコードは、WDK の
D:¥wdk9041¥wireless¥j2ee¥applications¥wdk¥examples-web¥mobile-xml ディレクトリにあります。このコード例は、WDK を初めて起動し、WDK によって例が解凍されるまでは参照できません。

URL の説明 : WDK を使用すると、アプリケーションを WDK 環境で実行していない場合でも、アプリケーションへの Wireless アクセスが可能です。URL の前半部分（http://localhost:9010/wdk/mcslite/http/）は、WDK で Wireless アプリケーションにアクセスすることを示しています。URL の後半部分（localhost/9010/examples/mobile-xml/Hello.jsp）はアプリケーションの URL です。

アプリケーションが Geocities に格納された場合、完全な URL は http://localhost:9010/wdk/mcslite/http/www.geocities.com/myaccount/Hello.jsp となります。

- ログにアクセスするための URL

http://<host_name>:9010/wdk/log。たとえば、http://localhost:9010/wdk/log となります。ログの URL にアクセスすると、WDK へのリクエストに関する全エラー・ログが示されます。この URL に PC のブラウザからアクセスすると、Hello World アプリケーションのログが表示されます。スクロール・ダウンすると、最新のログ出力を表示できます。

マルチメディア調整のデモ

OracleAS Wireless には、マルチメディア調整機能があります。アプリケーションに GIF イメージが含まれている場合、そのイメージはサーバーによって自動的にサイズ変更され、適切なフォーマットに変換されます。たとえば、WAP 電話の場合、GIF は WBMP フォーマットに変換されます。次に、複数のデバイスに表示される Oracle のロゴ・イメージの例を示します。この URL を PC のブラウザ、カラー表示の電話または白黒表示の電話にコピーおよび貼付けして、イメージを確認してください。

http://localhost:9010/wdk/mcslite/http/localhost/9010/examples/mobile-xml/ImageExample.jsp

4

JDeveloper Wireless Extension

項ごとに様々なトピックを記載しています。各項の内容は、次のとおりです。

- 概要
- マルチチャネル・アプリケーションの開発
- Wireless 対応 J2EE アプリケーションの作成
- J2ME アプリケーションの作成

概要

Oracle Extension Framework 上に構築される JDeveloper Wireless Extension（JWE）は、OracleAS Wireless の機能を Oracle JDeveloper と統合することによって、JDeveloper を使用した Wireless アプリケーションの作成を可能にします。この章では、JWE の概要、および JWE の機能の例をいくつか示します。完全なドキュメント、チュートリアルおよびダウンロードについては、http://otn.oracle.com/tech/wireless/tools/content.html を参照してください。

JWE を使用すると、WAP、メッセージ、音声などの様々な配信方法でアクセスできるマルチチャネル・アプリケーション、および標準の J2ME（Java 2 Micro Edition）MIDlet アプリケーションとメソッド・コールを通じて Web サービスと通信する MIDlet アプリケーションの両方を開発、デバッグ、デプロイおよび実行できます。

エミュレータまたは実際のデバイスを使用してアプリケーションをテストし、ブレーク・ポイントを使用して MIDlet をデバッグし、不明瞭化を使用して MIDlet を保護できます。JWE を使用すると、様々なメーカーの開発ツールキットとエミュレータを使用できます。

JWE を JDeveloper にインストールすると、JDeveloper のギャラリの「カテゴリ」ペインに、「Wireless アプリケーション」という新規カテゴリが作成されます。このノードを選択すると、JWE オプション（表 4-1 を参照）が起動されます。

表 4-1 JWE オプション

オプション	説明
J2ME デフォルト MIDlet	MIDlet を作成できます。
J2ME MIDlet デプロイメント	MIDlet の実行に必要なデプロイ・プロファイルを作成できます。
J2ME プロキシ接続	J2ME プロキシ・サーバーとの接続を作成できます。この接続から、Web サービスを WSDL（Web Service Definition Language）文書の URL に登録し、J2ME スタブ・クラスを生成します。
マルチチャネル・アプリケーション作成ウィザード	このマルチチャネル・アプリケーション作成ウィザードを使用すると、アプリケーションをモバイル対応にできます。
マルチチャネル・メッセンジャ作成ウィザード	統一されたメッセージを J2EE アプリケーションに追加できます。

マルチチャネル・アプリケーションの開発

JWE を使用すると、5 ステップのマルチチャネル・アプリケーション・ウィザードを使用して、JDeveloper IDE からアプリケーションをモバイル対応にできます。このウィザードでは、選択したアプリケーション用のテンプレートを生成できます。この目的のために、ウィザードには「基本機能」と「PushLite API を使用したメッセージのサポート」の 2 つのオプションがあります。これらのオプションには、それぞれ特定のテンプレートが関連付けられています。たとえば、「基本機能」を選択すると、次のテンプレートから選択できます。

- Hello World!（MXML）
- Hello World!（XHTML JSP）
- Hello World!（MXML JSP）
- Hello <Name>!（MXML JSP）- 2 ページ
- Hello <Name>!（XHTML JSP）- 2 ページ

アプリケーション用に適切なテンプレートを選択し（複数選択可）、ウィザードを完了すると、JDeveloper によってテンプレートが作成されます。次に、JDeveloper のコンポーネント・パレットを使用してアプリケーションにタグを挿入し、デバイス・エミュレータを使用してアプリケーションをテストします。

Wireless 対応 J2EE アプリケーションの作成

JWE では、マルチチャネル・メッセンジャ作成ウィザードを使用し、JDeveloper IDE で OracleAS Wireless Messaging を J2EE アプリケーションに追加できます。このウィザードでは、MultiChannelMessenger.java というファイルが、必要な OracleAS Wireless SDK ライブラリとともに J2EE のプロジェクト・ファイルに生成されます。MultiChannelMessenger.java ファイルには、このユーティリティ・クラスを使用するための例が含まれています。J2EE アプリケーションで、`MultiChannelMessenger` ユーティリティに対する `import` 文を追加します。次に、送信者、受信者および送信情報を指定して、J2EE アプリケーションで `MultiChannelMessenger` の API をコールします。

J2ME アプリケーションの作成

JWE を使用すると、通常の MIDlet、さらに Web サービスをコールする MIDlet も作成できます。

MIDlet の作成には、デフォルト MIDlet を作成する処理、およびその後でデプロイ用に MIDlet スイートにパッケージ化する処理の 2 つの手順があります。

デフォルト MIDlet の作成

デフォルト MIDlet を作成するために、JWE には、3 ページにわたるウィザードが用意されています。ここで、MIDlet のパッケージ名とクラス名を選択します。オプションで、MIDlet が作成されるプロジェクトに特定のライブラリを追加するか、または MIDlet のクラスの生成直後にデフォルト MIDlet をデプロイ・ウィザードに連鎖できます。アプリケーション固有のコードを追加すると MIDlet をさらに拡張できます。

MIDlet アプリケーションのデプロイ

デプロイ・ウィザードを使用すると、MIDlet の実行に必要なデプロイ・プロファイル情報のパッケージ化とデプロイを指定する MIDlet スイートを作成できます。MIDlet スイートを作成するには、選択したプロジェクトに組み込む Java クラス、イメージ（.png ファイル）またはその他のリソースを選択します。次に、`javax.microedition.midlet.MIDlet` クラスを拡張し、MIDlet へのエントリ・ポイントとして機能するクラスをプロジェクトのクラスの中から選択します。その後、選択したクラスに対する MIDlet 名を入力し、その MIDlet を MIDlet スイートに公開するように選択します。MIDlet には、アイコンを関連付けることもできます。さらに、MIDlet スイートの名前設定、ネットワーク・プロキシの設定（必要な場合）、MIDlet の manifest エントリの管理、MIDlet とともにデプロイするライブラリの選択（複数選択可）を行います。MIDlet は、MIDlet スイートの完成後すぐにデプロイできます。MIDlet がデプロイされると、それをエミュレータで実行およびテストできます。

Web サービスをコールする MIDlet の作成

JWE を使用すると、Web サービスをコールする MIDlet を作成できます。MIDlet は、MIDlet と Web サービス間の通信を最適化する OracleAS Wireless J2ME プロキシ・サーバーを使用して Web サービスをコールします。

JWE を使用すると、WSDL（Web Services Definition Language）文書を使用して Web サービスを J2ME プロキシ・サーバーに登録し、サービス用の J2ME スタブ・クラスを生成できます。MIDlet は、スタブ・クラスのメソッドをコールします。これらの各メソッドは、Web サービスの操作を順に表します。JWE を使用すると、Web サービスのメソッド・コールをテストする MIDlet を迅速に作成できます。その後、MIDlet スイートを作成して、MIDlet をデプロイおよび実行できます。

5

サービスの開発

この項では、アプリケーション開発者が、サービス・マネージャを使用して Oracle Application Server Wireless リポジトリのサービス関連オブジェクトを作成および管理する方法を説明します。項ごとに様々なトピックを記載しています。各項の内容は、次のとおりです。

- サービス・マネージャの概要
- サービス・マネージャへのログイン
- アプリケーションの管理
- 通知の管理
- マスター・アラートの管理（使用中止）
- データ・フィーダの管理
- プリセット定義の管理
- J2ME Web サービスの管理

サービス・マネージャの概要

サービス・マネージャには、アプリケーション、通知、データ・フィーダ、プリセット定義、J2ME Web サービスなどのサービス関連オブジェクトを作成するための一連のウィザードが用意されています。サービス・マネージャのウィザードでは、各オブジェクトの作成方法が別個のタスクとして一連のステップに分けて提示されるため、オブジェクトを迅速に作成できます。これらのステップの完了に必要なものは、最小限の情報のみです。サービス・マネージャでは、情報を正しく入力できるように各ステップごとに順にガイドします。

これらのウィザード以外に、サービス・マネージャを使用して OracleAS Wireless リポジトリ・オブジェクトを編集することもできます。また、サービス・マネージャはアプリケーションのテストやデバッグにも使用できます。

表 5-1 で、サービス・マネージャを使用して作成、変更、テストおよび削除できるサービス関連オブジェクトを説明します。

表 5-1 Wireless リポジトリのサービス関連オブジェクト

オブジェクト・タイプ	説明
アプリケーション・フォルダ	アプリケーション・フォルダに、アプリケーションが編成されます。
アプリケーション	アプリケーションは、エンド・ユーザーによってデバイスから起動されるか、またはエンド・カスタマにメッセージを配信するために通知エンジンによって起動されます。サービス・マネージャでは、次のタイプのアプリケーションを作成できます。 ■ マルチチャネル・アプリケーション ■ 通知アプリケーション（アラート） ■ J2ME アプリケーション ■ Web クリッピング・アプリケーション 通知アプリケーションは通知に基づいて起動され、通知はデータ・フィーダに基づいて起動されます。
通知	通知は通知エンジンによって起動されます。アプリケーションの実行時には、データ・フィーダ、タイマー、ロケーション・イベント・サーバーのいずれかからイベント・データが通知にプッシュされます。次に、このイベント・データはユーザー・サブスクリプション条件と比較されます。イベント・データがサブスクリプション条件と一致した場合は、通知メッセージの生成または特定処理の実行（あるいはその両方）のために通知アプリケーションが起動されます。通知メッセージのデフォルトの配信メカニズムでは、XMS（XML メッセージング・サーバー）が使用されます。

表 5-1 Wireless リポジトリのサービス関連オブジェクト（続き）

オブジェクト・タイプ	説明
データ・フィーダ	データ・フィーダは、データ・フィーダ・プロセスによって起動されます。実行時には、データ・フィーダによって外部コンテンツ・ソースからイベント・データが取得され、通知にプッシュされます。
プリセット定義	プリセット定義では、プリセットの属性（および属性タイプ）が定義されます。通常、プリセット定義はアプリケーションに関連付けることができます。プリセット属性に値を入力することで、エンド・ユーザーはアプリケーションに対してパーソナライズ情報を提供できます。この結果、アプリケーション・コードでプリセットのパーソナライズ情報を問い合せ、パーソナライズされた Wireless アプリケーションを配信できます。
J2ME Web サービス	J2ME Web サービスは、J2ME プロキシ・サーバーでホスティングされ、デバイスで実行されている J2ME MIDlet アプリケーションから起動されます。J2ME Web サービスは、Web サービスの WSDL URL の指定、JAR ファイルの URL の指定、またはクライアント・マシンの JAR のファイル・パスの指定によって登録できます。

サービス・マネージャへのログイン

サービス・マネージャにアクセスする手順は、次のとおりです。

1. 最初に、次の URL を使用して OracleAS Wireless Tools にログインします。

   http://hostname:port/webtool/login.uix

   たとえば、次の URL を使用してログイン・ページにアクセスします。

   http://hostname:7777/webtool/login.uix

   注意： 7777 は Oracle Application Server Wireless のデフォルトのポート番号です。ポート番号の範囲は 7777 ～ 7877 です。正しいポート番号を使用していることを確認するには、[Oracle home]/install/portlist.ini に格納されている Oracle Application Server Wireless のポート番号をチェックします。ポートの使用方法の詳細は、Oracle Application Server のインストール・マニュアルおよび『Oracle Application Server 管理者ガイド』を参照してください。

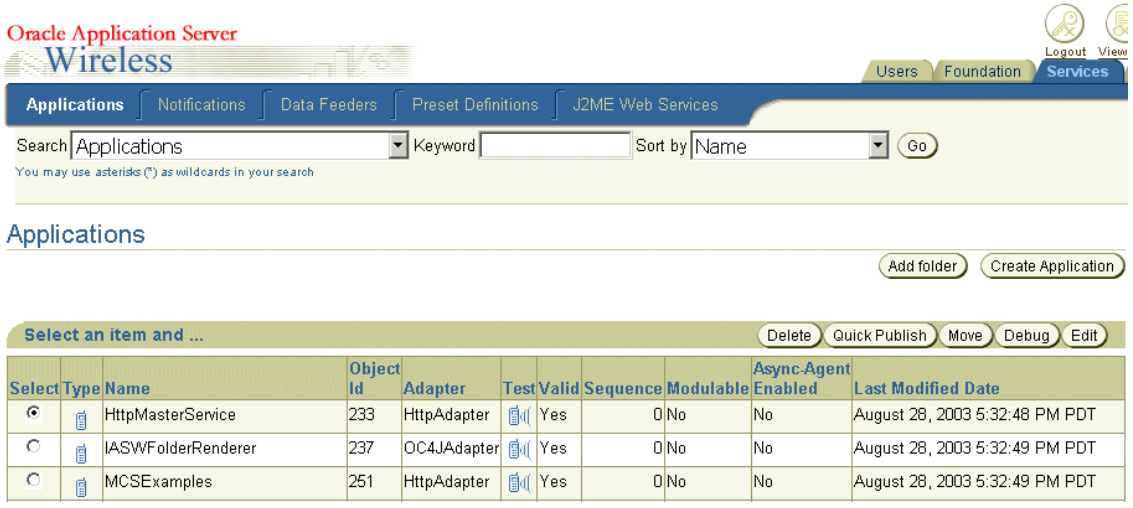
2. ユーザー名を入力し、次にパスワードを入力します。管理者の場合は、ユーザー名として orcladmin を入力します （パスワードはインストール時に設定されますが、ユーザー・マネージャを使用して変更できます)。

OracleAS Wireless Tools にログインした後、**「サービス」**タブ（図 5-1）をクリックしてサービス・マネージャにアクセスします。サービス・マネージャには次のサブタブがあります。

- アプリケーション
- 通知
- データ・フィーダ
- プリセット定義
- J2ME Web サービス

これらのサブタブのいずれかをクリックすると、オブジェクト固有の参照画面が起動されます。これらの参照画面から、作成、編集と削除、およびテスト用の機能を使用してオブジェクトを管理できます。

図 5-1 サービス・マネージャ（一部）



アプリケーションの管理

「アプリケーション」タブを選択すると、「アプリケーション」参照ページが表示されます。このページで、アプリケーションおよびフォルダを作成、編集、削除、検索および移動できます。このページでは、アプリケーションのテストやデバッグ、およびデバイス・シミュレータでのアプリケーションの表示もできます。

アプリケーションを作成し、テストした後、参照ページで**「クイック公開」**ボタンをクリックすると、アプリケーションをホーム・フォルダに公開できます。アプリケーションを公開すると、そのアプリケーションをデバイス・ポータルから実行できます。

「アプリケーション」画面では、階層内のトップレベル・フォルダを表示できます。これはサービス・マネージャでハイパーリンクとして表示されます。これらのハイパーリンク（図 5-2 を参照）を順にクリックしていくと、下位の階層に順にドリルダウン（横断）できます。トレース・パスによって階層の構造が表示されるため、現在アクセスしているレベルがわかります。

図 5-2 ナビゲーション・パス

Services > Applications > modules > pim

サービス・マネージャに初めてアクセスすると、アプリケーション・フォルダの参照画面がデフォルトで表示されます。この参照画面には、リポジトリ内の現在のフォルダとアプリケーションがリストされた表が表示されます。表 5-2 で、この表のヘッダー行を説明します。

表 5-2 サービス・マネージャの「フォルダの参照」画面の要素

要素	説明
タイプ	オブジェクトのタイプ。
名前	フォルダまたはアプリケーションの表示名。サービス・マネージャでは、フォルダがハイパーリンクとして表示されるため、フォルダの内容を参照できます。
オブジェクト ID	オブジェクトのオブジェクト ID（OID）。
アダプタ	アプリケーションが使用するアダプタ。
テスト	電話のアイコンをクリックすると、選択したアプリケーションをデバイス・シミュレータに表示できます。
有効	列に「Yes」と表示されている場合、オブジェクトは使用可能です。「No」と表示されている場合は使用不可です。
順序	アプリケーションとフォルダが出力デバイスに表示される順序。デフォルトでは、順序番号、名前の順にソートされて表示されます。

表 5-2 サービス・マネージャの「フォルダの参照」画面の要素（続き）

要素	説明
モジュール化可能	このアプリケーションを、モジュール化可能なアプリケーションとしてデプロイできるかどうかを示します。
非同期可能	サービス・マネージャを使用すると、HTTP 以外のプロトコルによるアプリケーションへのアクセスを可能にして、アプリケーションを補強できます。たとえば、Web ブラウザはないが、双方向のメッセージや電子メールをサポートしているデバイスのユーザーがアクセスするアプリケーションには、非同期エージェントを割り当てることができます。 ユーザーは非同期可能アプリケーションを使用して Web コンテンツにアクセスできます。たとえば、株価、トラフィック・レポート、星占いなどの Web コンテンツを取得する OracleMobile サービスをサブスクライブしているエンド・ユーザーが、Async@OracleMobile.com にメッセージを送信したとします。この場合は、OracleAS Wireless Server で稼働している非同期リスナーによって、このメッセージ（電子メールやショート・メッセージ）が捕捉され、リクエストが適切なサービスまたはアプリケーションにルーティングされ、リクエストされた情報がユーザーに戻されます。
最終更新日時	オブジェクトが最後に変更された時間。

マスター・アプリケーションの検索

サービス・マネージャの参照画面では、検索フィールドを検索オプションのドロップダウン・リストとともに使用してアプリケーションまたはフォルダを検索できます。この検索オプションは、検索範囲を絞り込むため、または広げるために使用できます。検索結果は、「検索結果」画面にリストで表示されます。

オブジェクトを検索するには、次の処理を 1 つ以上実行してから、**「実行」**をクリックします。

1. 次のいずれかのオプションをドロップダウン・リストから選択して、検索範囲を絞り込むか、または広げます。
   - すべてのアプリケーション
   - 非同期エージェント使用可能アプリケーション
   - モジュール化可能なアプリケーション
   - フォルダ
2. キーワードを入力します。
3. ドロップダウン・リストから、検索結果表示用に次のいずれかのソート・オプションを選択します。
   - **「名前」**。このオプションを選択すると、アプリケーションまたはフォルダの名前で結果がソートされます。

- **「最終更新日時」**。このオプションを選択すると、オブジェクトが最後に更新された日時で結果がソートされます。

4. **「実行」**をクリックします。「検索結果」画面が表示され、取得されたオブジェクトが示されます。

フォルダの作成

サブフォルダを作成することによってアプリケーションを編成できます。これらのサブフォルダ（トピック領域などを表します）は、他のサブフォルダにネストできます。作成したサブフォルダは、サービス・マネージャによって、アプリケーション参照画面にハイパーリンクとして表示されます。このハイパーリンクをクリックすると、フォルダの内容を参照できます。

フォルダを作成するには、最初にアプリケーション参照画面で**「フォルダの追加」**をクリックします。「フォルダの作成」画面が表示されます。この画面でフォルダのプロパティを定義します（表 5-3 を参照）。画面入力の完了後、**「作成」**をクリックします。参照画面が再表示され、新規フォルダが表示されます。

表 5-3「フォルダの作成」画面のパラメータ

パラメータ	値
フォルダ名	フォルダの名前。これは必須フィールドです。
説明	フォルダの説明。
有効	このオプションを選択すると、フォルダが使用可能になります。
タイトル・アイコン URI	このフォルダが現行フォルダになったときに、画面上部に表示され、アイコンとして使用されるイメージの URI。イメージ・フォーマットはユーザーのデバイスにあわせて OracleAS Wireless によって選択されるため、この URI でフォーマット・タイプを指定する必要はありません。
メニュー・アイコン URI	メニュー・リストでフォルダの横に表示され、アイコンとして使用されるイメージの URI。イメージ・フォーマットはユーザーのデバイスにあわせて OracleAS Wireless によって選択されるため、この URI でフォーマット・タイプを指定する必要はありません。
タイトル・オーディオ URI	ユーザーがフォルダにアクセスしたときに、音声 XML ゲートウェイによって読み込まれるオーディオ・ファイル（**.wav** ファイルなど）の URI。オーディオ・ファイル・フォーマットはゲートウェイにあわせて OracleAS Wireless によって選択されるため、この URI でフォーマット・タイプを指定する必要はありません。
メニュー・オーディオ URI	メニュー・リストのフォルダとともに音声 XML ゲートウェイによって読み込まれるオーディオ・ファイル（**.wav** ファイルなど）の URI。オーディオ・ファイル・フォーマットはゲートウェイにあわせて OracleAS Wireless によって選択されるため、この URI でフォーマット・タイプを指定する必要はありません。

アプリケーションの作成

アプリケーションを作成するには、サービス・マネージャのアプリケーション作成ウィザードを使用します。作成プロセスは複数のステップに分かれており、1 ステップごとに 1 つの画面で示されます。定義する必要があるのは、アプリケーションに必要で、適用可能なパラメータのみです。不要な画面（またはパラメータ）はスキップでき、**「終了」**をクリックするとアプリケーションの作成が完了します。

アプリケーション作成ウィザードにアクセスするには、参照ページで**「アプリケーションの作成」**をクリックします。

アプリケーション・タイプの選択

「アプリケーションの作成」をクリックすると、「アプリケーション・タイプ」画面が表示されます（図 5-3）。この画面で、作成するアプリケーションのタイプを選択します。次の 4 つのタイプがあります。

- マルチチャネル・アプリケーション

  マルチチャネル・アプリケーションの場合、アプリケーションのコンテンツは HTTP アダプタから取得されます。アプリケーション結果は、音声、メッセージ、ブラウザも含めて複数のデバイス・チャネルに変換およびレンダリングされます。

- J2ME Midlet

  MIDlet は、Java の MIDP（Mobile Information Device Profile）仕様をサポートするデバイスで実行されるアプリケーションです。このアプリケーションは、J2ME アプリケーションをダウンロードすると起動され、スマート・デバイスからオンラインおよびオフラインで実行できます。

- マルチチャネル・アプリケーション

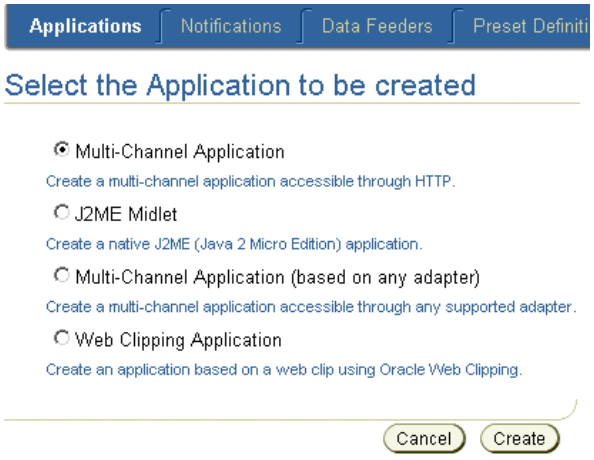
  これらのアプリケーションの場合、アプリケーション・コンテンツは、HTTP アダプタや SQL アダプタなど、任意のアダプタから取得されます。アプリケーション結果は、音声、メッセージ、ブラウザも含めて複数のデバイス・チャネルに変換およびレンダリングされます。

- Web クリッピング・アプリケーション

  Web クリッピング・アプリケーションのコンテンツは、Web ベースのアプリケーションをモバイル対応にできる Web クリッピング・スタジオから取得されます。アプリケーション結果は、音声、メッセージ、ブラウザも含めて複数のデバイス・チャネルに変換およびレンダリングされます。

アプリケーション・タイプを選択した後で、**「作成」**をクリックすると、アプリケーション作成ウィザードが起動されます。

図 5-3「アプリケーション・タイプ」画面



マルチチャネル・アプリケーションの作成

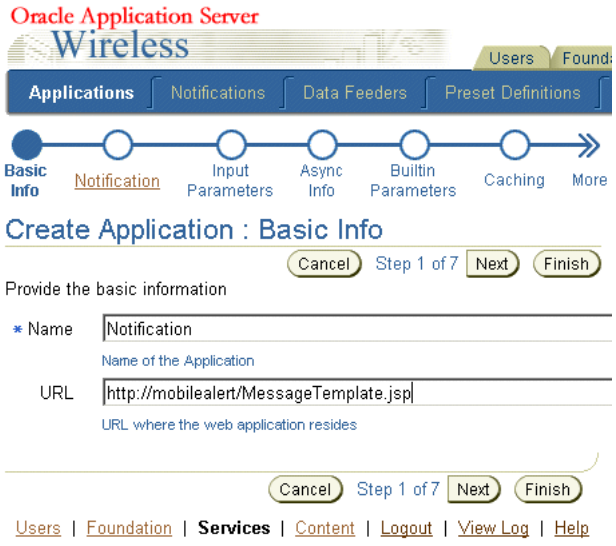
「マルチチャネル・アプリケーション」を選択して**「作成」**をクリックすると、ウィザードの第 1 ページが表示されます。このページでは、アプリケーションに関する基本情報を入力します（図 5-4）。

アプリケーションに関する基本情報の入力

アプリケーションの一意名を入力し（同一フォルダ内に存在する 2 つのアプリケーションに同じ名前は設定できません）、オプションで、OracleAS Wireless XML または XHTML を生成するリモート・アプリケーションを指し示す URL を入力します。このアプリケーションが通知アプリケーションによって使用されるメッセージ・テンプレートである場合は、URL を定義する必要はありません。

この時点でアプリケーションの作成を完了し、**「終了」**をクリックしてウィザードを終了するか、または**「次」**をクリックしてアプリケーションをさらに詳細に定義することもできます。OracleAS Wireless ウィザードのいずれかの時点で**「取消」**をクリックすると、入力した値はすべて消去されます。

図 5-4 基本情報の入力



通知関連情報の入力

このアプリケーションが通知（アラート）に基づいて起動される場合は、ウィザードのステップ 2（図 5-5）を入力します。

このアプリケーションが、通知メッセージのコンテンツを生成するために通知エンジンによって起動される場合は、「通知有効化」オプションを選択します。

注意： 通知メッセージのコンテンツ生成時に、これらのアプリケーションでは、モバイル・アプリケーション（JSP として、あるいは MXML または XHTML で作成）によって指定されるビジネス・ロジックを起動できます。通知アプリケーションの入力および出力パラメータは、アプリケーションの入力パラメータにマップして Web アプリケーションに渡すことができるため、Web アプリケーションは適切な処理を実行できます。

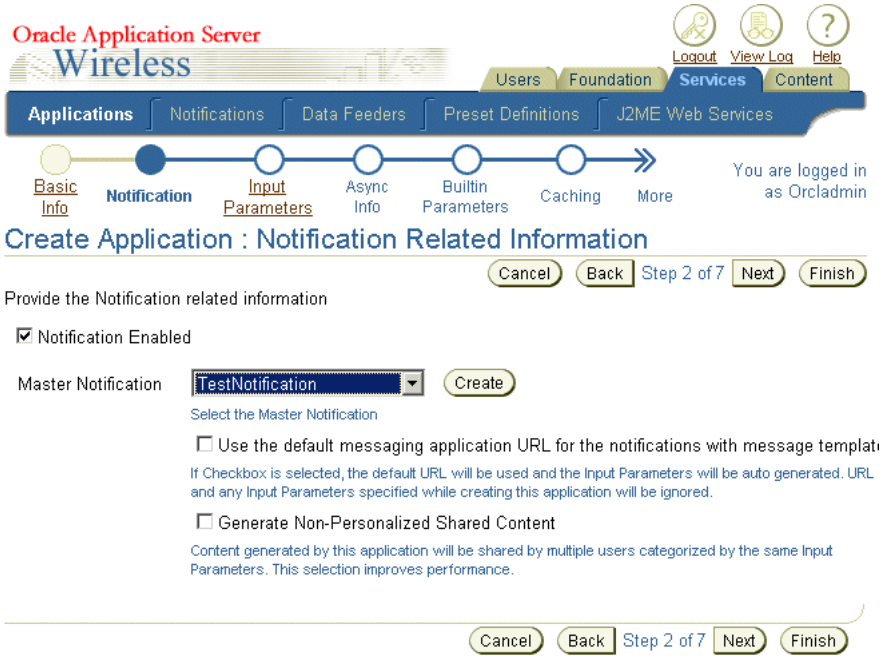
「通知有効化」オプションを選択した後は、次のことを実行します。

- 「通知」ドロップダウン・リストから通知を選択します。リストされている通知の中に必要な通知がない場合は、**「作成」**をクリックし、通知作成ウィザードを使用して通知アプリケーションを作成します。通知の詳細は、5-43 ページの「通知の管理」を参照してください。

- デフォルトのメッセージ生成メカニズムを使用する場合は、「メッセージ・テンプレートを使用した通知には、デフォルトのメッセージング・アプリケーションの URL を使用します。」を選択します。OracleAS Wireless には、通知テンプレートを基に通知メッセージを生成するデフォルトのアプリケーション JSP が用意されています。メッセージ・テンプレートが含まれている通知とともにこのオプションを選択すると、デフォルトの JSP によってメッセージ生成が処理されます。モバイル・アプリケーションの URL を提供する必要はありません。詳細は、5-48 ページの「ステップ 3: メッセージ・テンプレートの作成」を参照してください。
- 着信イベントごとにモバイル・アプリケーションが 1 回のみ起動されるようにするには、「非パーソナライズ共有コンテンツの生成」を選択します。この場合、モバイル・アプリケーションに渡されるユーザー情報はシステム・ユーザーで、生成されたコンテンツはこのイベントに対する権限のあるユーザーによって共有されます。モバイル・アプリケーションでは、ユーザーごとの特別な処理は実行できず、各ユーザーに対するパーソナライズ・メッセージを生成できません。ただし、このオプションを選択すると、モバイル・アプリケーションは複数の通知に対して 1 回のみ起動されるため、パフォーマンスが向上します。通知の詳細は、第 11 章「通知エンジン」を参照してください。

「**次**」または「**終了**」をクリックします。

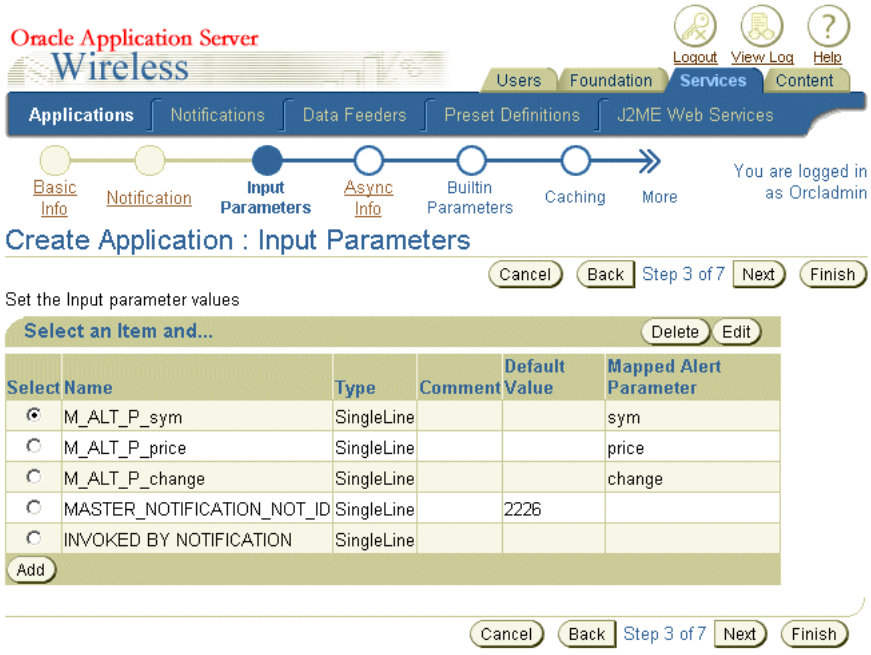
図 5-5 アプリケーションの通知情報の入力



アプリケーションの入力パラメータの入力

「入力パラメータ」画面（図 5-6）には、モバイル・アプリケーション（JSP、XHTML、MXML）に渡されるパラメータが表示されます。アプリケーションのパラメータは、この画面で定義するか、またはアプリケーションをアプリケーション・リンクとして公開するときにコンテンツ・マネージャによって定義できます。コンテンツの公開の詳細は、『Oracle Application Server Wireless 管理者ガイド』を参照してください。

図 5-6「入力パラメータ」画面



エンド・ユーザーからの入力が必要なパラメータの作成

「空で OK」チェックボックスを選択しないままにして値を必要とするパラメータを作成し、さらにこのパラメータに値を入力しないと、OracleAS Wireless によって、実行時にこの値の入力をユーザーに求めるプロンプトが表示されます。

コンテンツ・マネージャによって変更可能なパラメータの作成

「変更可能」オプションを選択すると、コンテンツ・マネージャでは、このアプリケーションからアプリケーション・リンクを作成するときにこのパラメータを変更できます。このオプションを選択しない場合、コンテンツ・マネージャではこのパラメータを変更できません。

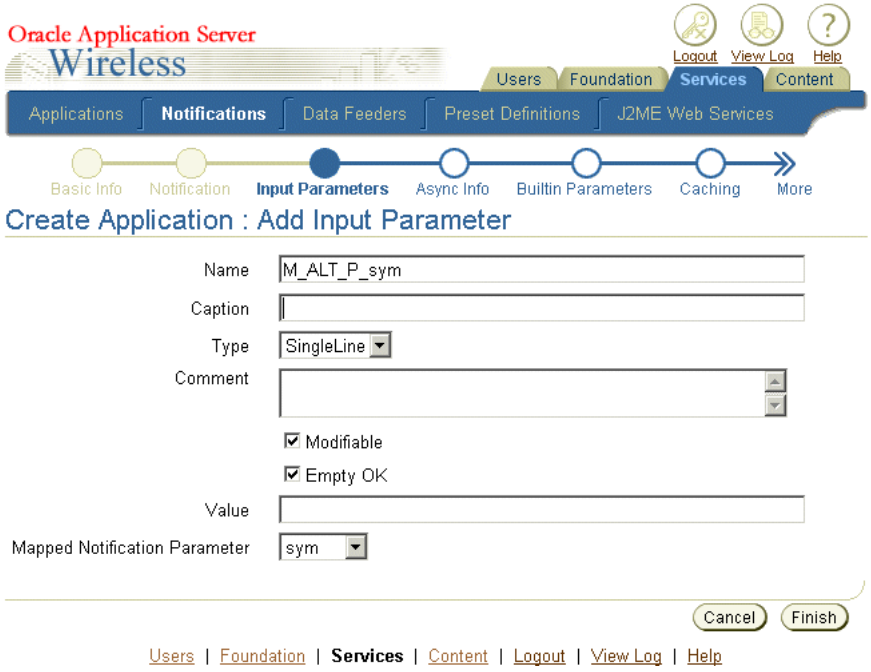
アプリケーションへの新規入力パラメータの追加

新規パラメータをアプリケーションに追加するには、最初に**「1 行追加」**をクリックします。次に、表 5-4 にリストされているパラメータを定義し、**「次」**または**「終了」**をクリックします。図 5-7 に、新規入力パラメータの値を入力する「入力パラメータ」画面を示します。

表 5-4 入力パラメータの追加

パラメータ	説明
名前	入力パラメータ名。
キャプション	入力パラメータを説明する表示ラベル。ユーザーからの処理または入力を要求します。
タイプ	入力パラメータのタイプを選択します。選択肢は、「単一行」、「複数行」、「列挙」、「パスワード」です。
コメント	入力パラメータの説明。
変更可能	このオプションを選択すると、コンテンツ・マネージャでパラメータを変更できます。
空で OK	入力パラメータに値が必要ない場合は、このオプションを選択します。
値	必要な場合は、パラメータのデフォルト値を入力します。デフォルト値を入力しない場合は、Wireless によって、ユーザーに値の入力を求めるプロンプトが表示されます（つまり、パラメータに対して「空で OK」のフラグも選択されていない場合）。
マップされた通知パラメータ	ドロップダウン・リストからパラメータを選択します。これらは「通知関連情報」画面で選択したマスター通知固有のパラメータです。

図 5-7 入力パラメータの追加



入力パラメータの削除 入力パラメータを削除するには、削除するパラメータを入力パラメータ値から選択して**「削除」**をクリックします。

入力パラメータの編集 入力パラメータを編集するには、編集するパラメータを入力パラメータ値から選択して**「適用」**をクリックします。

非同期情報の入力

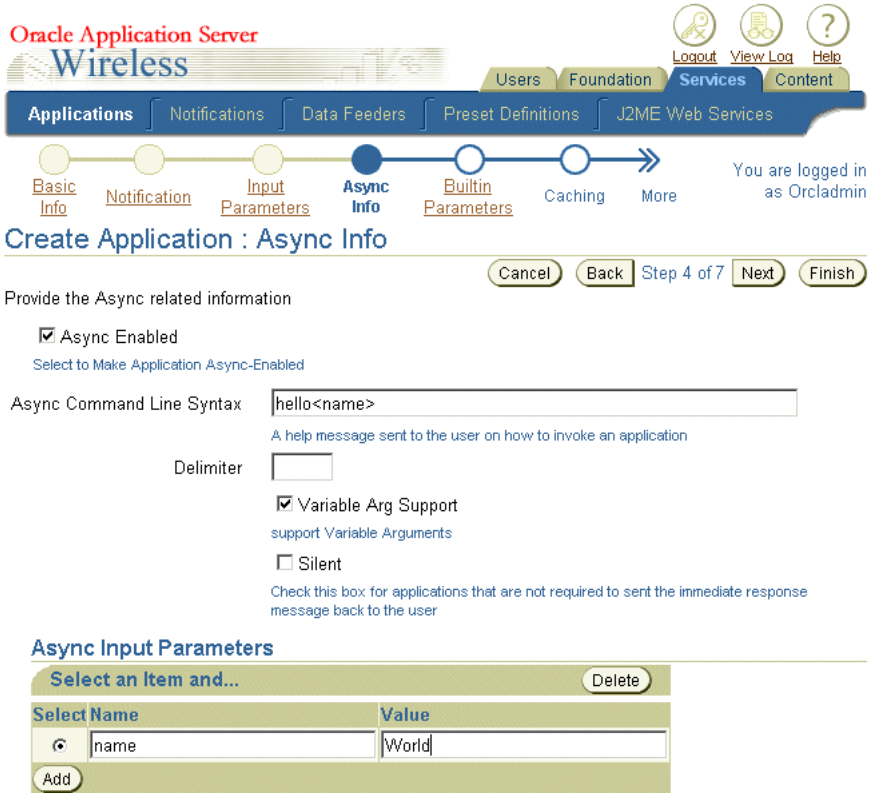
「非同期可能」オプションを選択した場合、このアプリケーションは、公開されると、非同期デバイスでメッセージを使用してアクセスできます。ユーザーは、OracleAS Wireless サーバーにリクエスト・メッセージを送信してアプリケーションを起動します。非同期リスナーは、メッセージを取得し、適切な非同期アプリケーションにルーティングします。次に、実行したリクエストの結果を含むメッセージをユーザーに送信して、メッセージでユーザーに返信します。複雑な非同期アプリケーションでは、メッセージのラウンドトリップが数回必要となる場合があります。OracleAS Wireless では、同じデバイスから送信されたメッセージに対するセッションが保持されます。

「非同期可能」チェックボックスを選択し、表 5-5 に示されている値を定義して非同期情報を設定します。図 5-8 は、非同期情報を定義した状態の「非同期情報」画面です。

表 5-5 非同期情報の定義

パラメータ	値
非同期コマンドライン構文	ユーザーが非同期リスナーに対してアプリケーション・ヘルプ・コマンドを発行したときに戻されるヘルプ・テキストを入力します。通常、このヘルプ・テキストでは、アプリケーションの短縮名またはアプリケーションのコマンド引数を提示して、アプリケーションの起動方法を説明します。
区切り文字	このアプリケーションの引数を区切る区切り文字を入力します。空白（" "）がデフォルトの区切り文字です。
可変引数のサポート	非同期アプリケーションに渡される引数の数が変化する場合、または非同期アプリケーションで定義されている引数の数を超える場合はこのオプションを選択します。
サイレント	非同期アプリケーションがアプリケーション結果のメッセージを戻す必要がない場合はこのオプションを選択します。

図 5-8「非同期情報」画面



非同期入力パラメータの追加「追加」をクリックして入力パラメータを追加し、次に「名前」フィールドに引数の名前を入力します。「値」フィールドに引数のデフォルト値を入力します。このフィールドを空白のままにすると、ユーザーに値の入力を求めるアプリケーションが作成されます。

「次」または「終了」をクリックします。

組込みパラメータの設定

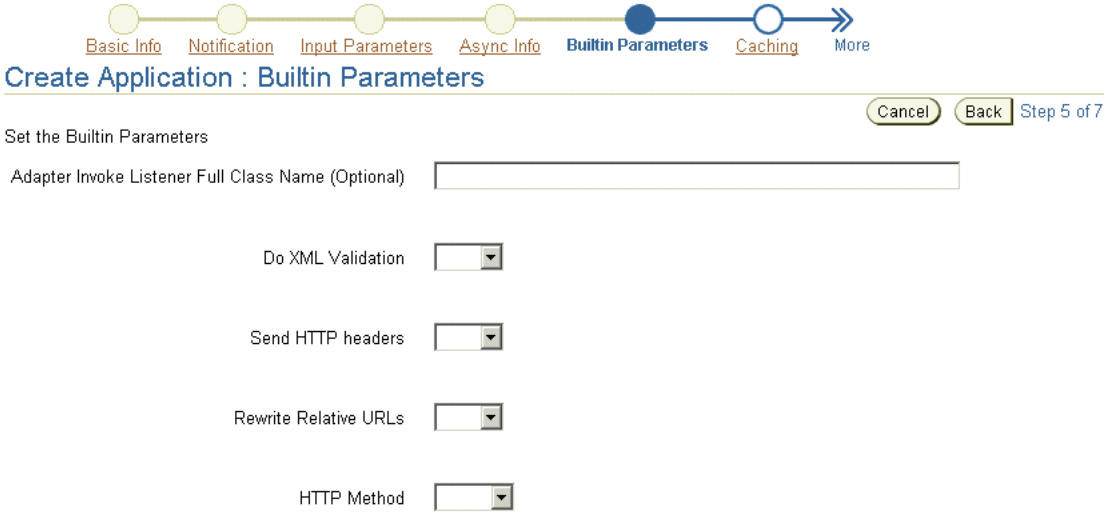
組込みパラメータは、事前定義済の HTTP アダプタ・パラメータです。これらのパラメータは適切な値にデフォルト設定されるため、設定する必要はありません。図 5-9 に、ウィザードの「組込みパラメータ」画面を示します。

ただし、これらのデフォルト値を上書きする必要がある場合は、次のように情報を提供します。

- 構文が定義されている CatSpeech サーバーの URI を入力します。
- オーディオ・ライブラリ・クラスが定義されているオーディオ・ライブラリのベース URI を入力します。
- アダプタ起動リスナーの完全なクラス名を入力します。アダプタ起動リスナーは実行時に起動され、HTTP リクエスト・パラメータなど、実行時の状態の様々な面をアプリケーションの実行前に検査します。
- モバイル・アプリケーションによって生成された XML を HTTP アダプタで検証する場合は、「XML 検証の実行」に「true」を選択します。検証を行うと、ランタイム・パフォーマンスが低下する場合があります。
- HTTP アダプタで HTTP ヘッダーを使用してモバイル・アプリケーションに情報を送信する場合は、「HTTP ヘッダーの送信」に「true」を選択します。
- HTTP アダプタで、モバイル・アプリケーションの JSP、XHTML または MXML ファイル内の URL 出現箇所をリライトする場合は、「相対 URL のリライト」に「true」を選択します。URL は、モバイル・アプリケーション・サーバー自体ではなく、マルチチャネル・サーバーの URL を参照するようにリライトされます。
- HTTP メソッド（「POST」または「GET」）を選択します。これは、モバイル・アプリケーション・ページへの HTTP 接続を行うときに、HTTP アダプタで使用されるメソッドです。
- モバイル・アプリケーション・サーバーの入力エンコーディングを設定します。このエンコーディングは、HTTP アダプタがモバイル・アプリケーション・サーバーとの HTTP 接続を確立するときに使用されます。IANA キャラクタ・セット名を使用します。

「次」または**「終了」**をクリックします。

図 5-9「組込みパラメータ」画面



キャッシュ情報の設定

「キャッシュ可能」オプションを使用すると、キャッシュ可能アプリケーションを作成できます。これらのアプリケーションの場合、HTTP アダプタは、モバイル・アプリケーションが初めて起動されたときのみ、アプリケーションからコンテンツを取得します。後続の起動では、HTTP アダプタは、モバイル・アプリケーションからではなく WebCache からコンテンツを取得します。この結果、キャッシュ可能アプリケーションではアプリケーションの起動時間が短縮されます。

時間によって変化し、一定期間が経過すると無効になるアプリケーション・コンテンツに対しては、失効の頻度（更新間隔）を指定します。たとえば、毎週月曜日の午前 6 時に失効の頻度を指定した場合、キャッシュされているコンテンツは、毎週この時間に失効化されます。新しいコンテンツは、次回 HTTP アダプタによってアプリケーションからコンテンツが取得されたときに WebCache に格納されます。モバイル・アプリケーションのコンテンツが Web Cache に格納されないようにする場合は、「キャッシュ可能」オプションを選択せずに、**「次」**をクリックします。

コンテンツが Web Cache に格納されるアプリケーションを作成するには、「キャッシュ可能」チェックボックス（図 5-10 を参照）を選択して、キャッシュの頻度を数値で入力します。数値の単位は「単位」ドロップダウン・リストから選択します。

「単位」ドロップダウン・リストを使用して、次の時間単位の中から選択します。

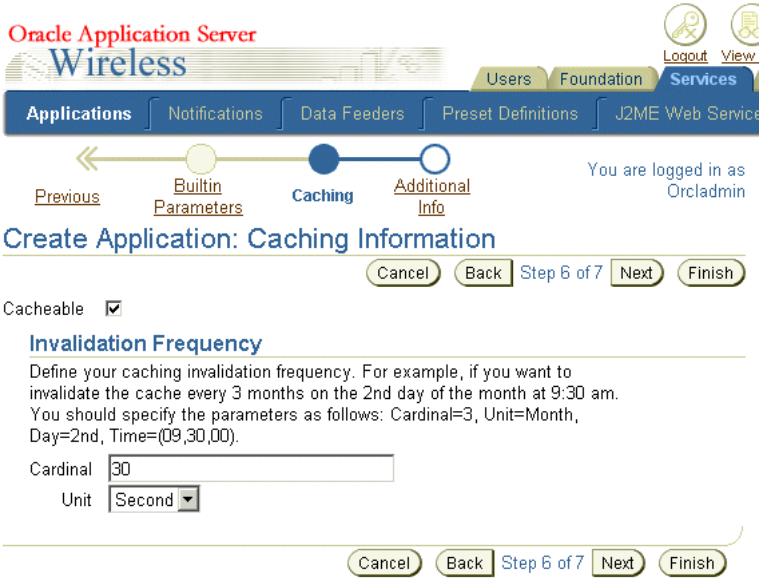
- 秒
- 分

- 時
- 曜日
- 週
- 月

ドロップダウン・リストを使用して、失効の頻度の日付と時間（適用可能な場合）を選択します。

「次」または**「終了」**をクリックします。

図 5-10 キャッシュ情報の設定



追加情報の設定

ウィザードの最後の画面（図 5-11）では、アプリケーションの表示モジュール構成の属性を設定できます。表 5-6 で、追加のパラメータを説明します。値の定義後、「終了」を選択するとマスター・アプリケーションの作成を完了できます。

表 5-6 マルチチャネル・アプリケーションの追加パラメータ

パラメータ	値
有効	アプリケーションを使用可能にして起動できるようにするにはこのオプションを選択します。
ロケーションに依存	ロケーションに固有のコンテンツを含むアプリケーションを作成するには、「ロケーション・ベース」チェックボックスを選択します。このオプションを選択した場合は、フラッシュライト・アイコンをクリックすると起動されるリストから、適切なリージョン ID も選択する必要があります。
モジュール化可能	モジュール化可能アプリケーション（他のアプリケーションからコール可能なアプリケーション）としてデプロイできるアプリケーションを作成する場合は、この「モジュール化可能」オプションを選択します。このオプションを選択した場合は、アプリケーション・リンクの構成ページのプラグインに使用される「構成 URL」、およびカスタマイズ・ページをプラグインするための「カスタマイズ URL」を指定する必要があります。詳細は、『Oracle Application Server Wireless 管理者ガイド』のコンテンツ・マネージャに関する章を参照してください。
OMP URL	OMP（Oracle Mobile Protocol）の URL を入力します。これは、このアプリケーションの位置を特定および起動するための一意の URL 識別子です。
メニュー・アイコン URI	現行アプリケーションになったときに、アプリケーションの横に表示されるアイコンとして使用されるイメージの URI を入力します。
タイトル・アイコン URI	メニュー・リストでアプリケーションの横に表示されるアイコンとして使用されるイメージの URI を入力します。
メニュー・オーディオ URI	ユーザーがこのアプリケーションをメニューから選択したときに読み込まれるオーディオ・ファイルの URI を入力します。
タイトル・オーディオ URI	メニュー・リストに読み込まれるオーディオ・ファイルの URI を入力します。
順序	順序番号を入力します。

図 5-11 アプリケーション作成ウィザードの「追加情報」ページ

J2ME アプリケーションの作成

OracleAS Wireless J2ME（Java 2 Micro Edition）アプリケーションは、J2ME ランタイムおよびライブラリ上にプログラミングされる J2ME MIDlet です。J2ME MIDlet 作成ウィザードを使用すると、MIDlet を OracleAS Wireless J2ME プロビジョニング・サーバーにアップロードできます。その後は、J2ME MIDlet をサポートしている PC またはデバイスに MIDlet をダウンロードできます。

J2ME Midlet 作成ウィザードには、次の 4 つのステップがあります。

- MIDlet に関する基本情報の入力
- 転送可能コンテンツの指定
- デバイス要件の指定
- MIDlet の追加情報の設定

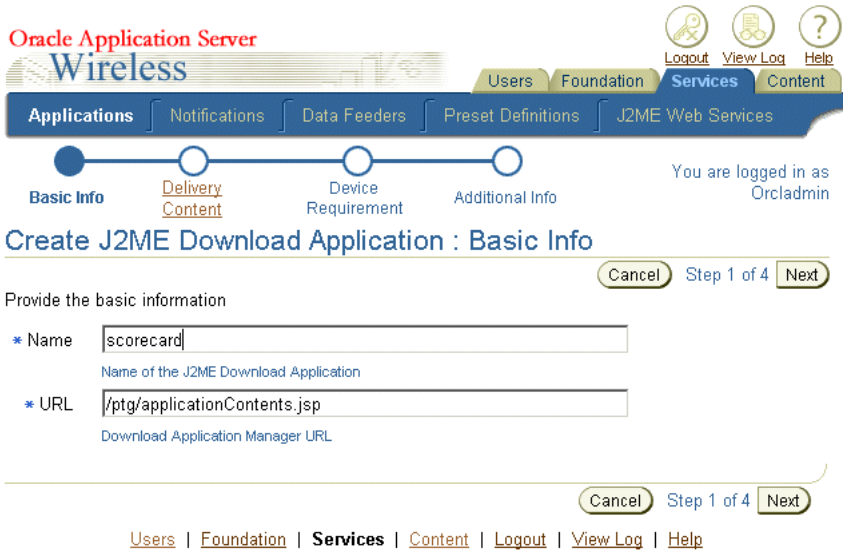
入力する必要があるのは、作成する MIDlet に関連する情報のみです。関連のない情報は、各ウィザード・ページの**「終了」**ボタンをクリックしてスキップできます。

MIDlet に関する基本情報の入力

ウィザードのステップ 1（図 5-12 を参照）では、2 つの必須パラメータを定義します。1 つは J2ME アプリケーション名、もう 1 つは J2ME アプリケーションのダウンロード・ページを生成するモバイル・アプリケーションの URL です。デフォルトで、OracleAS Wireless サーバーには、デフォルトの J2ME ダウンロード・ページが用意されています。

「次」をクリックします。

図 5-12 J2ME MIDlet に関する基本情報の入力



転送可能コンテンツの指定

OracleAS Wireless の J2ME MIDlet アプリケーションの場合、転送コンテンツはアプリケーションの中心部分である J2ME MIDlet バイナリです。転送コンテンツには、JAD（Java アプリケーション記述子）と JAR（Java アーカイブ・ファイル）が含まれます。コンテンツの各バージョンは、異なるデバイス要件にそれぞれ対応しています。

コンテンツのバージョンの入力

ステップ 2（図 5-13 を参照）で、コンテンツのバージョンを入力します。このアプリケーションの作成を完了すると、別のデバイス用にコンテンツの別バージョンを作成できます。アプリケーションの名前とバージョンで J2ME MIDlet が一意に識別されます。

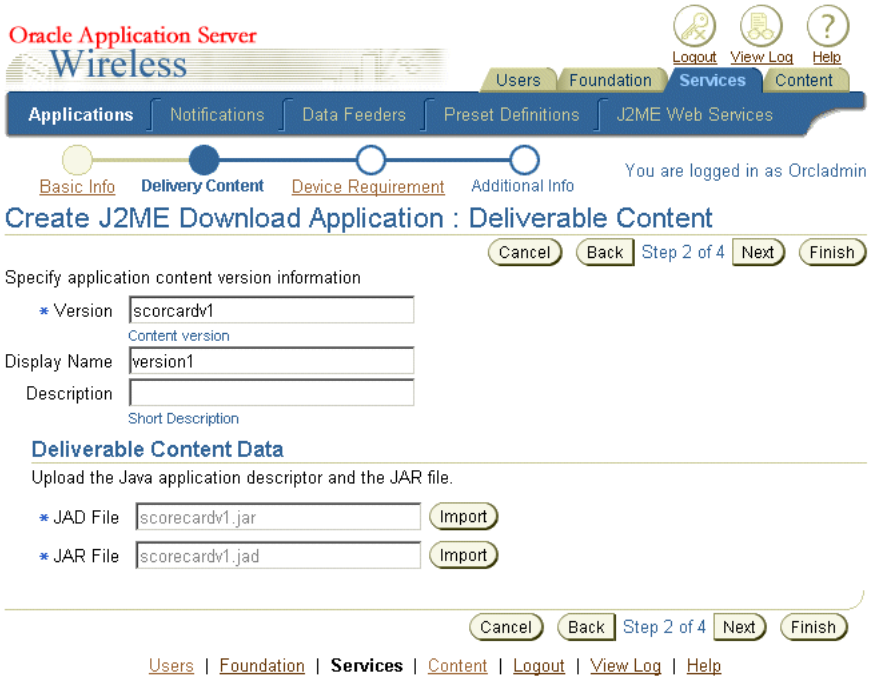
オプションで、このバージョンのコンテンツの表示名と説明も入力できます。

JAD ファイルと JAR ファイルのインポート

「インポート」ボタンを使用すると、このアプリケーションの JAD ファイルと JAR ファイルを参照および選択できます。これらのファイルをインポートするには、**「インポート」**ボタンをクリックします。「インポート・ファイル」ウィンドウが表示されます。**「参照」**ボタンをクリックし、ファイルを選択して**「インポート」**をクリックします。

この時点で、**「終了」**をクリックすると、アプリケーションの作成を完了できます。**「次」**をクリックすると、「デバイス要件」画面（図 5-14）が起動されます。

図 5-13 転送可能コンテンツの指定



デバイス要件の設定

デバイス要件基準は、実行時に J2ME がデバイスにダウンロードされるときに評価されます。転送可能コンテンツのバージョンごとにデバイス要件が異なります。デバイスからダウンロードが要求されると、J2ME プロビジョニング・サーバーでは、アプリケーションを要求しているデバイスのプロファイルと要件が一致するバージョン番号が選択されます。

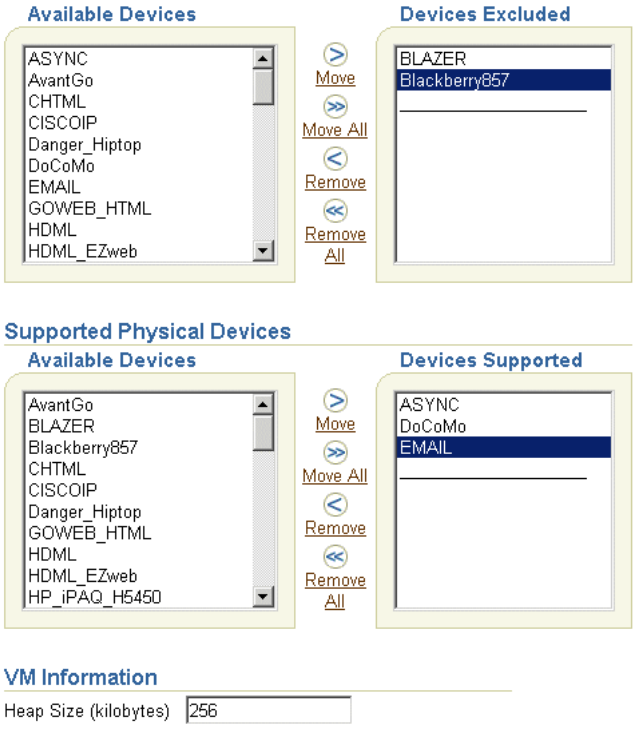
「除外されたデバイス」セクションでは、選択したデバイスへの MIDlet のダウンロードを禁止できます。デバイスを除外するには、右矢印ボタン（> および >>）を使用して「使用可能デバイス」ペインから「除外されたデバイス」ペインにデバイスを移動します。「除外されたデバイス」にリストされているデバイスにこの MIDlet をダウンロードしようとすると、デバイスにエラー・メッセージが表示されます。左矢印キー（< および <<）を使用すると、「除外されたデバイス」ペインから「使用可能デバイス」ペインにデバイスを移動できます。同様に、「サポートされる物理デバイス」セクションの矢印キーを使用すると、デバイス（このバージョンのコンテンツのダウンロードをサポートしているデバイス）を選択できます。

注意： 同じデバイスを除外とサポートの両方に選択することはできません。OracleAS Wireless では、このような相反する指定が行われたデバイスは自動的に除外されます。

この J2ME アプリケーションを実行している JVM（Java Virtual Machine）のヒープ・サイズ要件を指定します。

「次」または「終了」をクリックします。

図 5-14 転送可能コンテンツのデバイスの選択



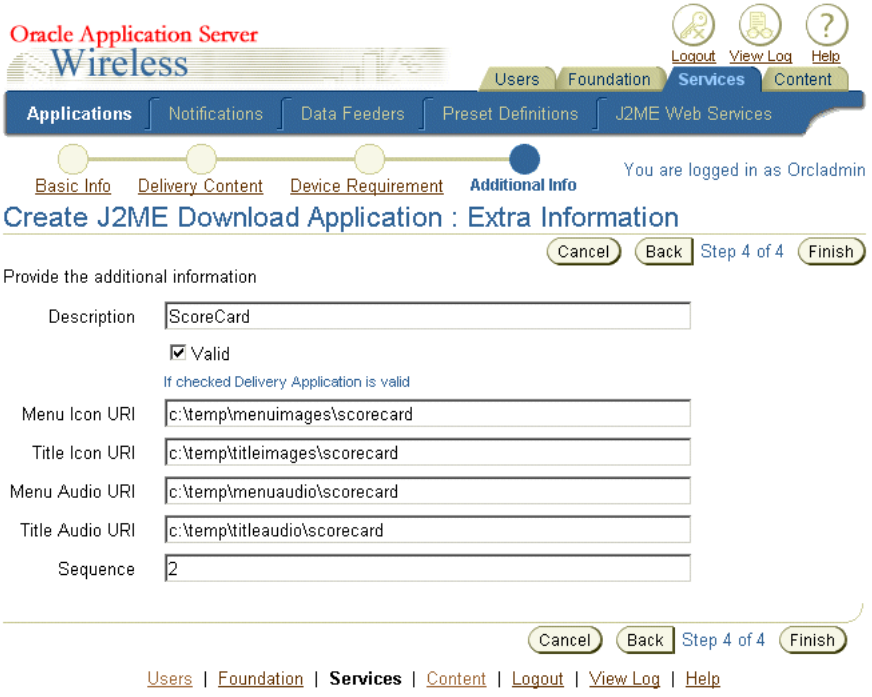
追加情報の設定

「追加情報」画面（図 5-15）のパラメータを定義すると、MIDlet アプリケーションの表示情報を設定できます。**「終了」**をクリックすると、J2ME MIDlet アプリケーションの作成が完了します。表 5-7 で、これらのパラメータを説明します。

表 5-7 MIDlet アプリケーションの追加の値

パラメータ	値
説明	J2ME アプリケーションの説明を入力します。この説明はデバイス上に表示されます。
有効	アプリケーションを起動できるようにするにはこのオプションを選択します。
メニュー・アイコン URI	アプリケーションが現行アプリケーションになったときに、そのアプリケーションの横に表示されるアイコンとして使用されるイメージの URI を入力します。
タイトル・アイコン URI	メニュー・リストでアプリケーションの横に表示されるアイコンとして使用されるイメージの URI を入力します。
メニュー・オーディオ URI	ユーザーがこのアプリケーションをメニューから選択したときに読み込まれるオーディオ・ファイルの URI を入力します。
タイトル・オーディオ URI	メニュー・リストに読み込まれるオーディオ・ファイルの URI を入力します。
順序	順序番号を入力します。

図 5-15 MIDlet アプリケーションに関する追加情報の入力



J2ME MIDlet アプリケーションの作成方法の詳細は、第 12 章「J2ME の開発とプロビジョニング」を参照してください。

マルチチャネル・アプリケーション（任意のアダプタに基づく）の作成

最初に、「アプリケーション・タイプ」画面（図 5-3）から「マルチチャネル・アプリケーション (任意のアダプタに基づく)」オプションを選択して、マルチチャネル・アプリケーションを作成します。

注意： アプリケーションを作成するには、最後まで順序に従う必要があります。いずれかの時点で**「取消」**をクリックしてウィザードを終了すると、入力した値はすべて消去されます。

ステップ 1: アプリケーションに関する基本情報の入力

「フォルダの参照」画面から、**「アプリケーションの作成」**をクリックします。アプリケーション作成ウィザードの「基本情報」画面が表示されます（図 5-16）。この画面を使用して、アプリケーションの構成パラメータを定義します。構成パラメータについては、表 5-8 で説明します。

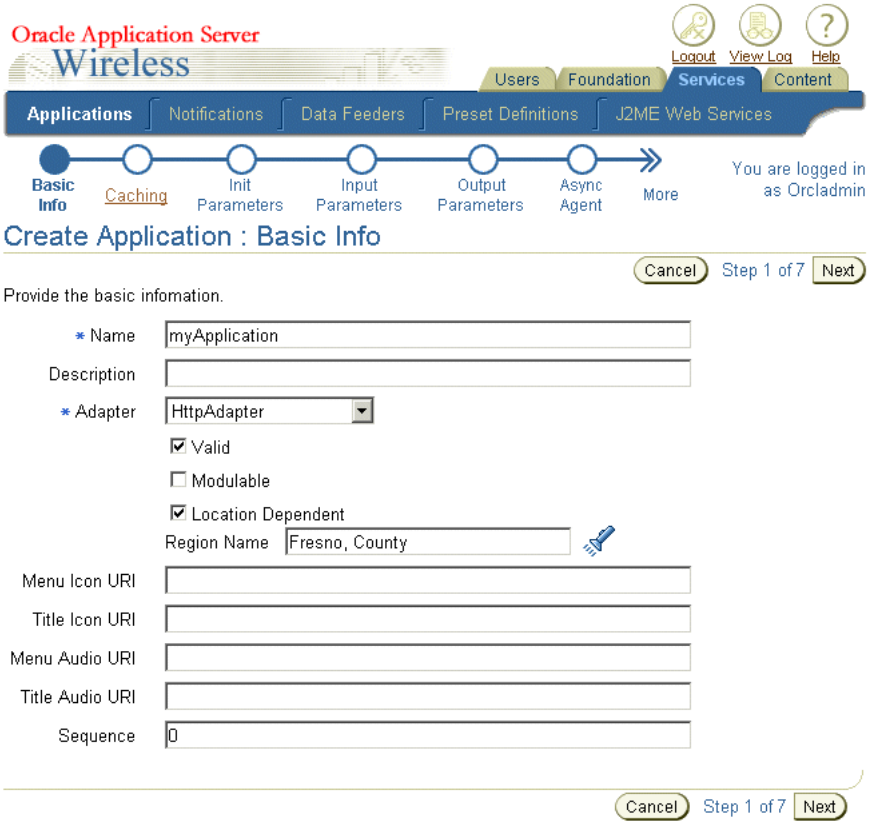
表 5-8 アプリケーションの基本構成パラメータ

パラメータ	値
名前	アプリケーション名。
説明	アプリケーションのオプションの説明。
アダプタ	選択可能なアダプタのドロップダウン・リスト。**注意：**SQL アダプタと Web Integration アダプタは、このリリースでは使用されなくなりました。アダプタの詳細は、『Oracle Application Server Wireless 管理者ガイド』を参照してください。
有効	アプリケーションを使用可能にするには、「有効」チェックボックスを選択します。
モジュール化可能	このチェックボックスを選択すると、別のアプリケーション内のモジュール・コンポーネントとしてデプロイ可能なアプリケーションが作成されます。モジュール化可能アプリケーションは、複数のアプリケーションによる再利用が可能で、エンド・ユーザーの入力を必要とするアプリケーションに対して一貫性のあるユーザー・インタフェースを提供します。
ロケーションに依存	アプリケーションを指定したリージョンに固有のアプリケーションにするには、このチェックボックスを選択します。このオプションを使用すると、実行時のロケーション取得が可能になります。
リージョン名	「ロケーションに依存」オプションを選択した場合は、このボタンをクリックしてリージョンを選択する必要があります。
言語	表示言語のドロップダウン・リスト。
タイトル・アイコン URI	このアプリケーションが現行アプリケーションになったときに、画面上部に表示されるアイコンとして使用されるイメージの URI。イメージ・フォーマットはユーザーのデバイスにあわせて OracleAS Wireless によって選択されるため、この URI でフォーマット・タイプを指定する必要はありません。

表 5-8 アプリケーションの基本構成パラメータ（続き）

パラメータ	値
メニュー・アイコン URI	メニュー・リストでサービスの横に表示されるアイコンとして使用されるイメージの URI。イメージ・フォーマットはユーザーのデバイスにあわせて OracleAS Wireless によって選択されるため、この URI でフォーマット・タイプを指定する必要はありません。
タイトル・オーディオ URI	ユーザーがアプリケーションにアクセスしたときに、音声 XML ゲートウェイによって読み込まれるオーディオ・ファイル（.WAV ファイルなど）の URI。オーディオ・ファイル・フォーマットはデバイスにあわせて OracleAS Wireless によって選択されるため、この URI でフォーマット・タイプを指定する必要はありません。
メニュー・オーディオ URI	メニュー・リストのアプリケーションとともに音声 XML ゲートウェイによって読み込まれるオーディオ・ファイル（WAV ファイルなど）の URI。オーディオ・ファイル・フォーマットはデバイスにあわせて OracleAS Wireless によって選択されるため、この URI でフォーマット・タイプを指定する必要はありません。
順序	このフィールドに入力する整数値によって、サービスやフォルダが出力デバイスに表示される順序を変更できます。デフォルトでは、順序番号、名前の順にソートされて表示されます。順序フィールドに値を入力すると、サービスやフォルダの表示順序を再配置できます。

図 5-16 アプリケーション作成ウィザードの「基本情報」画面

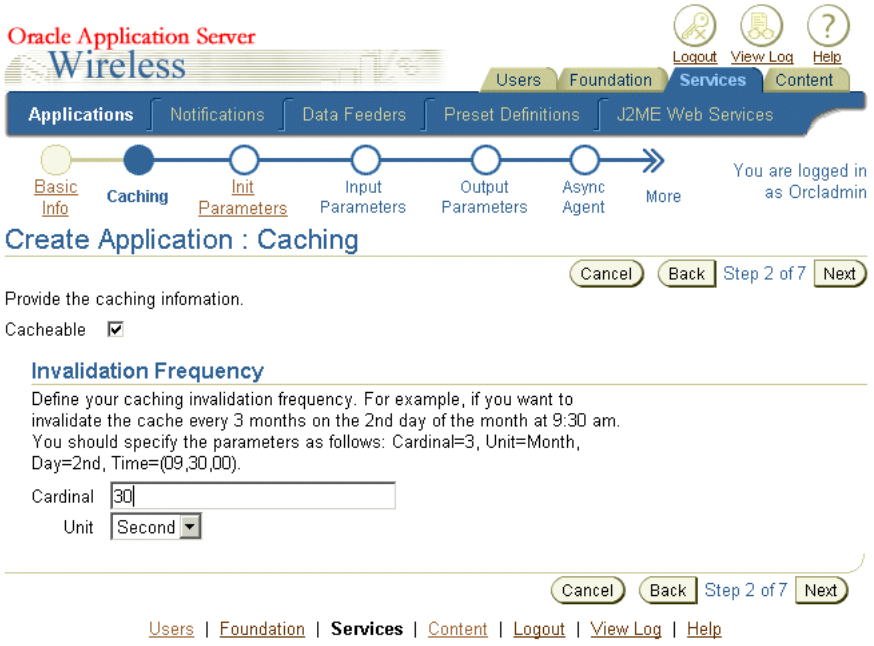


ステップ 2: キャッシュ情報の入力

変更コンテンツが含まれるアプリケーションに対して「キャッシュ可能」チェックボックス（図 5-17）を選択します。このオプションを選択すると、アダプタの起動と変換が不要になります。キャッシュ可能アプリケーションを作成する場合は、頻度も指定する必要があります。OracleAS Wireless サーバーは、この頻度で、失効レポートの発行によって Web ページが変更されたキャッシュを通知します。失効の頻度を定義するには、「カーディナル」フィールドに整数値を入力し、画面のドロップダウン・リストを使用して、時間間隔を詳細に定義します。キャッシュ可能なアプリケーションを作成しない場合は、「キャッシュ可能」チェックボックスを選択しないまま、**「次」**をクリックします。

「次」をクリックします。「初期パラメータ」画面が表示されます。

図 5-17 アプリケーション作成ウィザードの「キャッシュ情報」ページ



ステップ 3: アプリケーションの初期パラメータの入力

「初期パラメータ」画面には、ステップ 1 で選択したアダプタの初期パラメータが含まれます。すべてのアダプタに初期パラメータがあるわけではありません。初期パラメータの値を入力して、「**次**」をクリックします。選択したアダプタに初期パラメータが含まれていない場合は、「**次**」をクリックします。

注意： SQL アダプタと Web Integration アダプタは、このリリースでは使用されなくなりました。

デバッグなどの目的でリスナーをプラグインする場合は、「アダプタ起動リスナーの完全なクラス名」フィールドにリスナーのクラスを指定します。これらのリスナー・メソッドは、次の場合にコールされます。

- HTTP アダプタ起動の開始時
- リモート JSP に接続する前
- リモート JSP に接続した後

- HttpAdapter 起動の終了時
- エラー発生時

注意： OC4J 構成 / アプリケーション XML ファイル内のクラスパスを指定するか、または JAR ファイルを wireless/lib にコピーする必要があります。

ステップ 4: アプリケーションの入力パラメータの選択

「入力パラメータ」画面（図 5-18）には、ステップ 1 で選択したアダプタの入力パラメータが表示されます。アプリケーション作成ウィザードは、アダプタの定義を問い合せて、この画面に表示するパラメータを特定します。表 5-9 で、HTTP アダプタと OC4J アダプタを使用するアプリケーションの入力パラメータを説明します。

表 5-9 HTTP アダプタと OC4J アダプタの入力パラメータ

パラメータ	値
名前	入力パラメータ名。OracleAS Wireless のサービス作成ウィザードは、アダプタ定義を問い合せて入力パラメータ名を設定します。
コメント	Web Integration アダプタに基づくアプリケーションの場合は、OracleAS Wireless によって、パラメータを使用する WIDL サービスの名前がこのフィールドに自動的に移入されます。 他のアダプタに基づくアプリケーションの場合は、このフィールドを使用してパラメータの説明を記述できます。コメントは内部でのみ使用されます。
必須	パラメータに値が必要な場合はこのチェックボックスを選択します。値を必要としないパラメータ（オプションのパラメータなど）の場合は、このオプションを選択しないでください。
デフォルト値	ほとんどのパラメータの場合、この値はそのパラメータのデフォルト値を表します。デフォルト値を指定すると、OracleAS Wireless はユーザーに値の入力を求めません。デフォルト値は、コンテンツ・マネージャによって作成されるアプリケーション・リンクで指定された値で上書きでき、パラメータがユーザーに表示される場合は、ユーザーが OracleAS Wireless Customization を使用して上書きできます。 `PAsection` パラメータは Web Integration アダプタによって使用されます。`PAsection` の場合、この値は Web サービスで使用する必要がある WIDL サービスの名前です。名前はドロップダウン選択リストから選択できます。`PAsection` の値を指定しないと、OracleAS Wireless サービスには、WIDL インタフェース内のすべての WIDL サービスが含まれます。

「入力パラメータ」画面では、入力パラメータの選択の他に、このアプリケーションのアダプタ実装に対する入力パラメータの追加および削除ができます。

入力パラメータの選択

アプリケーションの入力パラメータを選択するには、使用する入力パラメータの横の**「選択」**ラジオ・ボタンをクリックして、**「次」**をクリックします。

アダプタへの新規入力パラメータの追加

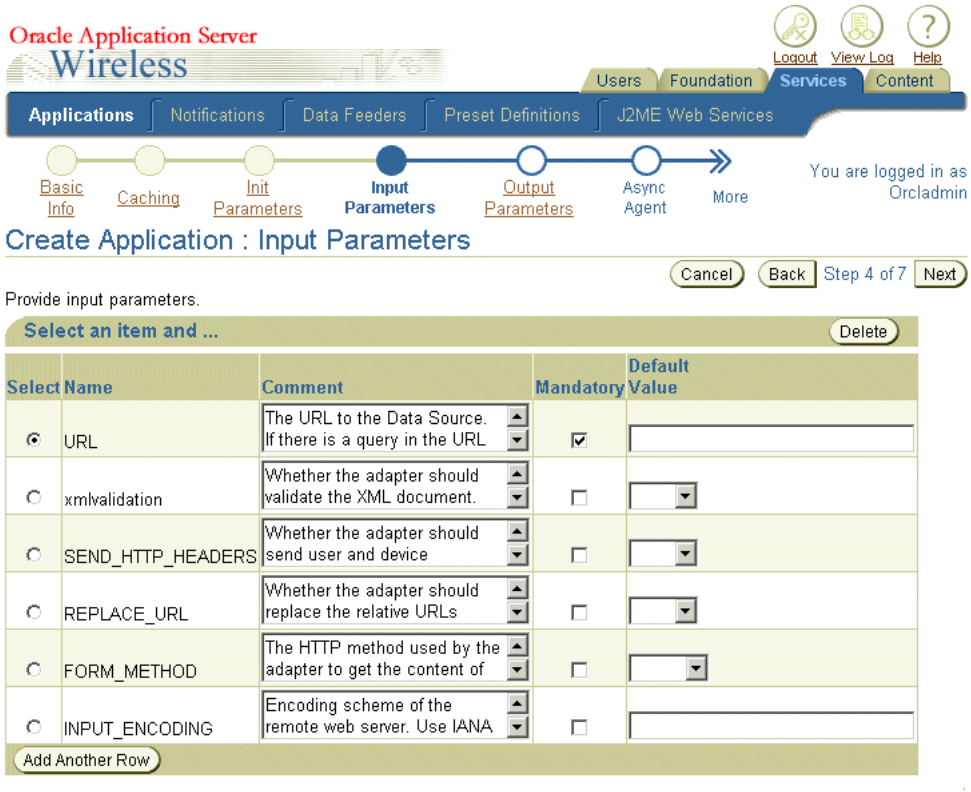
ステップ 1 で選択したアダプタに新規パラメータを追加するには、**「1 行追加」**をクリックします。表 5-9「HTTP アダプタと OC4J アダプタの入力パラメータ」の説明に従ってパラメータの値を入力して、**「次」**をクリックします。

SQL アダプタと Web Integration アダプタのパラメータの詳細は、『Oracle Application Server Wireless 管理者ガイド』を参照してください。

入力パラメータの削除

入力パラメータを削除するには、このアプリケーションのアダプタ実装から削除するパラメータを選択して、**「削除」**をクリックします。**「次」**をクリックします。

図 5-18 アプリケーション作成ウィザードの「入力パラメータ」画面



HTTP アダプタの入力パラメータの設定

HTTP アダプタは、リモート・コンテンツを取得し、MobileXML として配信します。表 5-10 で、HTTP アダプタの入力パラメータを説明します。

表 5-10 HTTP アダプタの入力パラメータ

パラメータ	説明
URL	データ・ソースの URL。URL に問合せがある場合、その文字と URL は次のようにエンコードされる必要があります。 http://my.host.com:80/Hello.jsp?fn=First+Name&ln=Last+Name これは必須パラメータです。
REPLACE_URL	アダプタで結果の中の相対 URL を絶対 URL に置き換えるかどうかを示します。結果の中に相対 URL がないことを確認した場合にのみ、このパラメータを false に設定してください。デフォルト値は true です。
FORM_METHOD	URL のコンテンツを取得するためにアダプタで使用される HTTP メソッド。サポートされているメソッドは GET と POST です。デフォルトのメソッドは GET です。
INPUT_ENCODING	リモート Web サーバーのエンコーディング・スキーマ。IANA キャラクタ・セット名（ISO-8859-1、UTF-8 など）を使用して定義します。

ステップ 5: アプリケーションの出力パラメータの選択

「出力パラメータ」画面では、ステップ 1 で選択したアダプタの出力パラメータを選択するか、またはアプリケーションに出力パラメータを追加できます。アプリケーション作成ウィザードは、アダプタの定義を問い合せて、この画面に表示するパラメータを特定します。

注意： HTTP アダプタと OC4J アダプタを使用するアプリケーションには、出力パラメータを定義する必要はありません。

表 5-11 で、アダプタの出力パラメータを説明します。

表 5-11 アダプタの出力パラメータ

パラメータ	値
名前	出力パラメータ名。アプリケーション作成ウィザードは、アダプタ定義を問い合せて出力パラメータ名を設定します。
キャプション	パラメータを説明するラベル。OracleAS Wireless で、ユーザーに入力を求めるプロンプトを表示するときに使用されます。
コメント	Web Integration アダプタに基づくアプリケーションの場合は、OracleAS Wireless によって、パラメータを使用する WIDL サービスの名前がこのフィールドに自動的に移入されます。 他のアダプタに基づくアプリケーションの場合は、このフィールドを使用してパラメータの説明を記述できます。コメントは内部でのみ使用されます。
ユーザー・カスタマイズ可能	このパラメータの値をエンド・ユーザーが設定できるかどうかを指定します。ほとんどの入力パラメータをカスタマイズ可能に指定できます。

出力パラメータを選択するには、ラジオ・ボタンを使用して適切な出力パラメータを選択し、次に**「適用」**をクリックします。出力パラメータを削除するには、出力パラメータを選択し、**「削除」**ボタンをクリックします。

アダプタへの新規出力パラメータの追加

アダプタに対する出力パラメータの追加または削除の完了後、**「次」**をクリックします。OracleAS Wireless によって、作成したアプリケーション内に `PASection` が検出されない場合は、「確認」画面が表示されます。「確認」画面にリストされた値を確認します。値が正しい場合は、**「終了」**ボタンをクリックしてマスター・アプリケーションの作成を完了します。

マスター・アプリケーションに `PASection` が含まれている場合は、結果トランスフォーマの作成画面が表示されます。

ステップ 6: 非同期エージェント・サービスの作成（オプション）

非同期エージェントをアプリケーションに割り当てることによって、HTTP 以外のプロトコルでアクセス可能なアプリケーションを作成します。

非同期エージェント・アプリケーションに関する値を設定するには、最初に「非同期エージェント」チェックボックスを選択します。「非同期コマンドライン構文」フィールドに、ユーザーが非同期サーバーに対してアプリケーション・ヘルプ・コマンドを発行したときに戻されるヘルプ・テキストを入力します。「区切り文字」フィールドには、非同期エージェント・サービスの区切り文字パラメータを入力します。

> **注意：** 空白（" "）がデフォルトの区切り文字です。

「非同期アプリケーション引数リスト」セクションを完了する手順は、次のとおりです。

1. **「1 行追加」**をクリックします。
2. 「名前」フィールドに引数の名前を入力します。
3. 引数がコマンドラインに表示される順序を表す番号を入力します。
4. 引数のデフォルト値を入力します。このフィールドを空白のままにすると、ユーザーに値の入力を求めるアプリケーションが作成されます。
5. **「次」**をクリックします。

ステップ 7: 結果トランスフォーマの選択（オプション）

アダプタの出力パラメータを設定すると、OracleAS Wireless によって、入力パラメータに `PASection` が含まれているかどうかがチェックされます。この値は、チェーン・サービス・シーケンスのエントリ・ポイントであるサービスを識別するために WIDL アダプタによって使用されます。アプリケーション作成ウィザードで、`PASection` の入力アダプタが検出された場合は、「結果トランスフォーマ」画面が起動されます。

トランスフォーマの画面では、アダプタのトランスフォーマを選択するか、またはローカル・ファイル・システムから XSLT スタイルシートをインポートすることによって新規トランスフォーマを追加できます。

注意： MobileXML を戻すアダプタを選択した場合は、このステップをスキップできます。

ステップ 1 で選択したアダプタのトランスフォーマを選択するには、ラジオ・ボタンを使用して**「適用」**をクリックします。アダプタからトランスフォーマを削除するには、「選択」ラジオ・ボタンを使用してトランスフォーマを選択し、**「削除」**をクリックします。

XSLT スタイルシートのインポート

1. 編集する `PASection` を表すタブをクリックします。各パネルには、XSLT スタイルシートを入力するためのテキスト・エディタが含まれています。「インポート」ボタンをクリックして XSLT スタイルシートをインポートすることもできます。
2. XSLT スタイルシートの編集完了後、**「次」**をクリックします。デバイス・トランスフォーマ画面が表示されます。結果トランスフォーマを作成しない場合は、この画面を空白のままにし、「確認」画面が表示されるまで**「次」**をクリックします。
3. 値が正しい場合は、**「終了」**をクリックしてアプリケーションの作成を完了します。

新規結果トランスフォーマの追加

新規結果トランスフォーマを追加する手順は、次のとおりです。

1. 「名前」フィールドにトランスフォーマ名を入力します。
2. **「インポート」**ボタンをクリックして、ローカル・ファイル・システムから XSLT スタイルシートを取得します。「コンテンツ」ウィンドウにスタイルシートが表示されます。
3. スタイルシートに必要な変更を加えます。
4. **「追加」**をクリックします。
5. **「終了」**をクリックして、マスター・サービスの作成を完了します。

これでアプリケーションが作成されました。このマスター・サービスは、これに基づくアプリケーションがコンテンツ・マネージャによってユーザー・グループに公開されるまでは、ユーザーに表示されません。

Web クリッピング・アプリケーションの作成

Wireless Web クリッピング・サーバーを使用すると、Wireless サービス管理者は、Web コンテンツをクリップおよびスクレイプし、Wireless Web クリッピング・サーバーのリポジトリに永続的に保存される Wireless Web クリッピング・アプリケーションを作成できます。モバイル・デバイス・ユーザーが Wireless Web クリッピング・アプリケーションを要求すると、HTTP アダプタは、そのアプリケーションを取得して、モバイル・デバイスに対する処理および配信のために OracleAS Wireless に配信します。

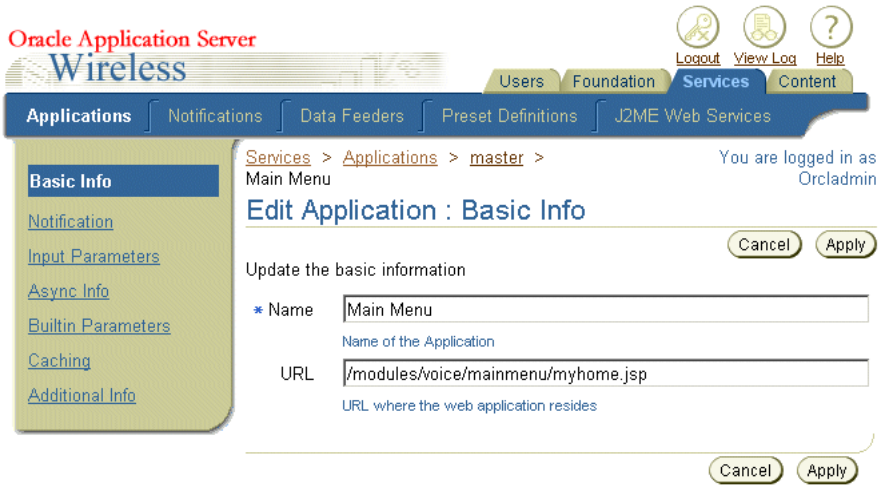
サービス・マネージャから Web クリッピング・マネージャにアクセスします。Web クリッピング・マネージャを使用すると、Web クリッピングを作成、編集および削除できます。また、クリッピング用のモバイル・アプリケーションを Java アプリケーションまたは JSP としてダウンロードできます。既存の Web クリッピングに基づいて、デフォルト・アプリケーションを作成できます。モバイル・アプリケーションの作成後、Web アプリケーションのクリップした部分は、複数のモバイル・デバイスから起動できます。詳細は、第 13 章「Web スクレイピング」を参照してください。

アプリケーション・ウィザードの最初のページでは、「Web クリッピング・アプリケーション」タイプを選択できます。**「作成」**をクリックすると、Web クリッピング・マネージャのページが表示されます。既存の Web クリッピングを選択して、**「デフォルト・アプリケーションの作成」**をクリックします。Web クリッピングに基づいて新規アプリケーションが作成されます。

アプリケーションの編集

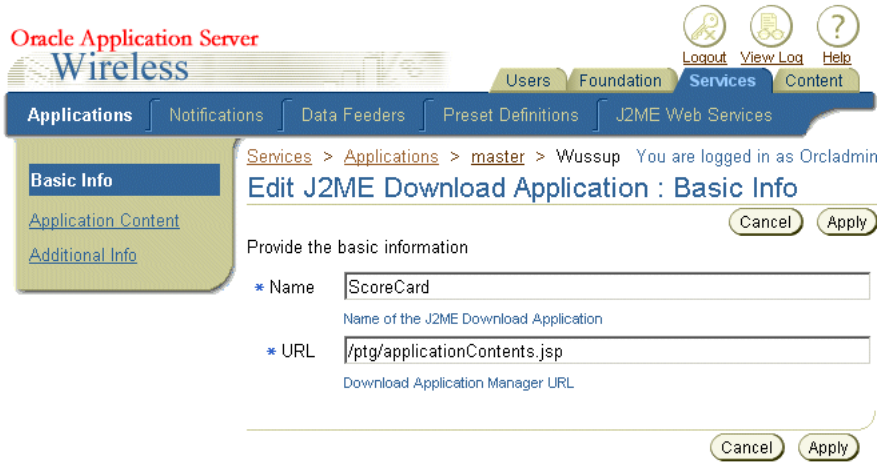
アプリケーション参照画面で**「編集」**ボタンを使用すると、基本情報から追加情報まで、アプリケーションに関するすべての情報を編集できます。アプリケーションを編集するには、この参照画面でアプリケーションを選択して**「編集」**をクリックします。フィールドに選択したアプリケーションの設定値が移入された状態の「基本情報」編集ページが表示されます(図 5-19)。編集画面の左側のパネルから、基本構成、初期パラメータ、入力パラメータ、出力パラメータおよび非同期プロパティの値など、編集する値を選択できます。値を変更した後、**「適用」**をクリックして変更内容を保存します。**「取消」**をクリックすると、値が元の状態に戻ります。編集するパラメータの詳細は、5-9 ページの「アプリケーションの作成」を参照してください。

図 5-19 アプリケーション編集用の「基本情報」画面



J2ME アプリケーションの編集時には、JVM、プロファイルおよびデバイス要件の最大ダウンロード・サイズの値を編集できます。図 5-20 は、J2ME アプリケーション編集用の「基本情報」画面です。

図 5-20 J2ME アプリケーションの編集



アプリケーションの削除

アプリケーションを削除するには、アプリケーション参照画面からアプリケーションを選択して**「削除」**をクリックします。

アプリケーションのデバッグ

サービス・マネージャを使用すると、デバイス・シミュレータと OracleAS Wireless XML またはデバイスでアプリケーションを同時に表示できます。

XSLT スタイルシートまたは Java クラスの形式のトランスフォーマでは、OracleAS Wireless のアダプタから戻されたコンテンツが、特定のプラットフォームに最も適したフォーマットに変換されます。

アプリケーションをテストする手順は、次のとおりです。

1. アプリケーション参照画面から、アプリケーションを選択します。
2. **「デバッグ」**をクリックします。「デバッグ・サービス」画面が表示されます。
3. 次の出力フォーマットの中から選択します。

   - Adapter XML Result

     この結果タイプを選択すると、ソースとターゲット出力デバイスとの中間のフォーマットである AdapterResult フォーマットで OracleAS Wireless のソース・コンテンツを参照できます。AdapterResult フォーマットのソース・コンテンツをターゲット・デバイスに配信するには、SimpleResult フォーマットに変換する必要があ

ります。「結果」パネルにテキストが表示されない場合、AdapterResult は生成されません。

- OracleAS Wireless XML Result

  OracleAS Wireless XML Result を選択すると、ソース・コンテンツが、アダプタから戻された出力の OracleAS Wireless の SimpleResult フォーマットで表示されます。

- Device Result

  デバイス・トランスフォーマ・ドロップダウン・メニューに、リポジトリ内のデバイスがリストされます。論理デバイスを選択すると、そのデバイス用の最終的なマークアップ言語を参照できます。

4. **「パラメータの設定」**をクリックします。
5. **「アプリケーションの実行」**をクリックします。アプリケーションがデバイス・シミュレータに表示されます。選択した結果が「アプリケーション結果」ウィンドウに表示されます。

アプリケーションのクイック公開

アプリケーションのテストとデバッグを終了した後は、アプリケーションをコンテンツ・マネージャを使用して公開するかわりに、アプリケーション・リンクとしてホーム・フォルダに公開できます。アプリケーションをホーム・フォルダに公開すると、そのアプリケーションをデバイス・ポータルから表示できます。

アプリケーションをホーム・フォルダに公開するには、最初に参照画面からアプリケーションを選択して、**「クイック公開」**をクリックします。アプリケーション・リンク名を入力して、**「作成」**をクリックします。

フォルダとアプリケーションの移動

サービス・マネージャの移動機能を使用すると、アプリケーションとフォルダを編成できます。

アプリケーションを移動するには、最初にフォルダまたはアプリケーションを選択して**「移動」**をクリックします。「移動」画面が表示されます。この「移動」画面のリストから新しい場所を選択します。**「ここに移動」**をクリックします。

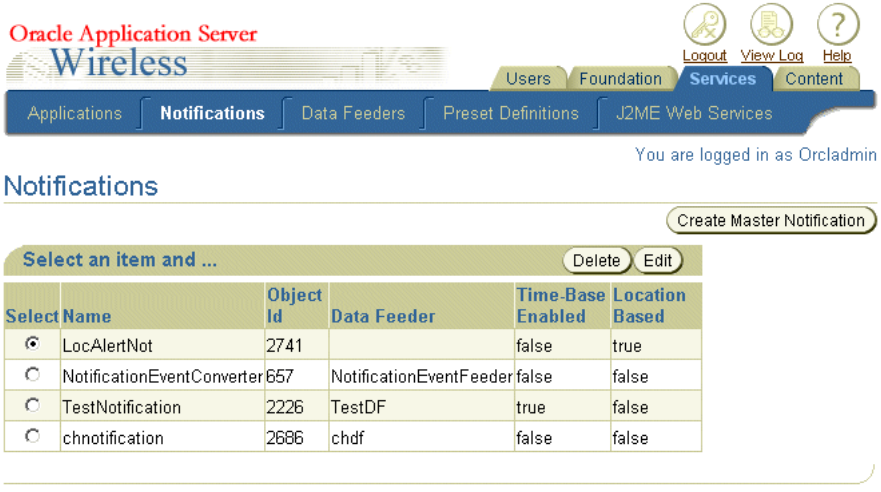
通知の管理

サービス・マネージャの「通知」タブでは、通知（アラート）を作成、編集および削除できます。「通知」タブを選択すると、通知の参照画面が表示され（図 5-21）、現在の通知のリストが示されます。「参照」画面の通知は、名前、OID、データ・フィーダおよび時間の値で構成されています。表 5-12 で、参照画面の要素を説明します。

表 5-12 通知の参照画面の要素

要素	説明
名前	通知名。
オブジェクト ID	データベース内の通知の ID。
データ・フィーダ	通知に使用されるデータ・フィーダまたはコンテンツ・ソース。
時間ベース可能	事前定義済時間で表示される通知を示します。

図 5-21 通知の参照画面



マスター通知の作成

通知作成ウィザードでは、マスター通知の作成方法がステップごとに示されます。このウィザードは、参照画面で**通知の作成**ボタンをクリックすると起動され、プロセスの各ステップごとに別々の画面を表示します。

注意： 作成した通知は、システム・マネージャがその通知を通知エンジン・プロセスに連結できるように、アプリケーションにマップします。この通知は、システム・マネージャが、通知エンジン・プロセスとデータ・フィーダ・エンジン・プロセスの両方を起動するとアクティブになります。

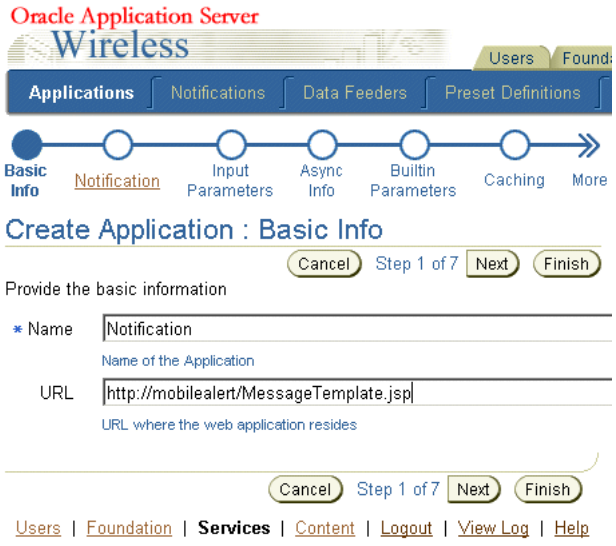
ステップ 1: 通知の基本構成パラメータの入力

通知作成ウィザードの最初の画面である「基本情報」画面（図 5-22）で、通知の次の構成パラメータを定義します。表 5-13 で、「基本情報」画面のパラメータを説明します。

表 5-13 通知の基本構成パラメータ

パラメータ	値
名前	通知名。これは必須パラメータです。
説明	通知の説明。
サブスクライバ・フィルタリング・フック	Java クラス名。このフックを使用すると、これらの通知がメッセージング・サーバーに送信される前に、限定された通知の対象となるサブスクライバをフィルタ処理で除外できます。
値ベース	この通知がイベントの受信に基づいてトリガーされるかどうかを指定します。
データ・フィーダ名	データ・フィード・ソースのドロップダウン・リスト。この通知が値ベースの場合、このフィールドに入力された値はデータ・フィーダを指し示している必要があります。
ロケーションベース可能	この通知がロケーション条件の検証に基づいてトリガーされるかどうかを指定します。
時間ベース可能	この通知が事前定義済時間でトリガーされるかどうかを指定します。頻度の選択肢は、毎日、平日および週末です。タイム・ゾーン情報はユーザー・プロファイルによって提供されます。

図 5-22 マスター通知作成ウィザードの「基本情報」画面



「次」をクリックします。「トリガー条件」画面が表示されます（図 5-23）。

ステップ 2: 通知のトリガー条件の設定

「トリガー条件」画面では、エンド・ユーザーのデバイスで通知を起動する条件を設定できます。たとえば、株価についてユーザーにアラートする通知を作成する場合は、株価が特定の価格を上回ったときまたは下回ったときにエンド・ユーザーが通知を要求できる条件を設定します。

表 5-14 で、「トリガー条件」画面のパラメータを説明します。

表 5-14 通知のトリガー条件

パラメータ	値
条件名	通知に対するアラート・トリガーの名前。トリガー名は 30 文字以内で、英数字とアンダースコアのみ使用できます。さらに、最初の文字には数字を使用できず、また SQL 予約語は使用できません。エンド・ユーザーが通知アプリケーションにサブスクライブした場合にこのラベルが表示されます。
トリガー・パラメータ	トリガー・パラメータは、トリガー条件を定義するデータ・フィーダ内の要素です。たとえば、株価アラート・サービス用のデータ・フィーダに stock price という出力パラメータが含まれている場合は、条件名に対するトリガー・パラメータとして stock price を選択できます。データ・フィーダの出力パラメータの設定方法は、5-69 ページの「データ・フィーダの出力パラメータの編集」を参照してください。
条件タイプ	エンド・ユーザーが設定する値と関連して、通知をトリガーする条件。
デフォルト値	パラメータのデフォルト値。デフォルト値を指定すると、OracleAS Wireless はユーザーに値の入力を求めません。デフォルト値は、コンテンツ・マネージャによって作成されるアプリケーションで指定された値で上書きでき、パラメータがユーザーに表示される場合は、ユーザーが OracleAS Wireless Customization を使用して上書きできます。

トリガー条件間の関係の設定

AND リレーション（両方の条件と一致）または OR リレーション（いずれかの条件と一致）を選択します。

トリガー条件の選択

トリガー条件を選択する手順は、次のとおりです。

1. トリガー条件のリストから、トリガー条件を選択します。
2. 「条件タイプ」、「トリガー・パラメータ」または「デフォルト値」フィールドを必要に応じて編集します。
3. **「適用」**をクリックします。

新規トリガー条件の追加

新規トリガー条件を追加する手順は、次のとおりです。

1. 「条件」フィールドにトリガー条件名を入力します。
2. 「キャプション」フィールドに、エンド・ユーザーに入力を求めるプロンプトを表示するときに使用するテキストを入力します。
3. 「トリガー・パラメータ」フィールドのドロップダウン・リストからトリガー・パラメータを選択します。
4. 「条件タイプ」フィールドのドロップダウン・リストから条件タイプを選択します。条件タイプは、トリガー・パラメータのデータ型によって異なります。

データ型が数値の場合は、次の条件が含まれます。

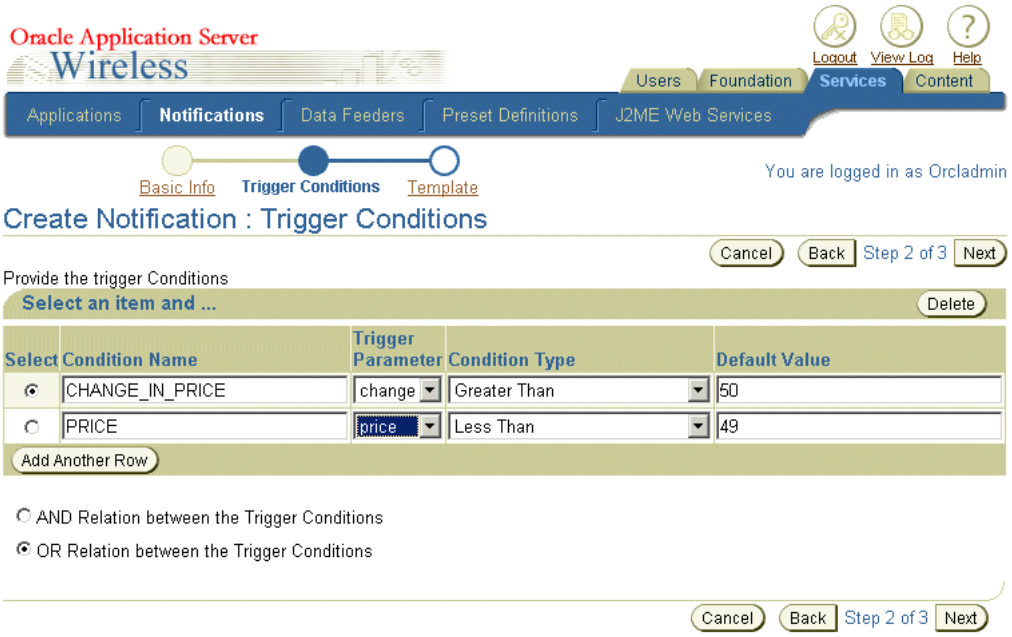
- 次より小さい
- 次より大きい
- 等しい
- 以下
- 以上
- 絶対値より小さい
- 絶対値より大きい
- 絶対値に等しい
- 絶対値以下
- 絶対値以上
- 値の変更（このタイプの条件値は、0（ゼロ）または1のみです。0（ゼロ）はトリガーされないことを意味し、1は値が変更されたときにトリガーされることを意味します。デフォルト値は0（ゼロ）です。）

データ型がテキストの場合は、次の条件タイプが含まれます。

- 完全一致
- 不一致
- 含む
- 含まない
- 先頭
- 最後
- 値の変更（このタイプの条件値は、0（ゼロ）または1のみです。0（ゼロ）はトリガーされないことを意味し、1は値が変更されたときにトリガーされることを意味します。デフォルト値は0（ゼロ）です。）

5. 「デフォルト値」フィールドにトリガー条件のデフォルト値を入力します。
6. **「追加」**をクリックします。
7. **「次」**をクリックします。「メッセージ・テンプレート」画面が表示されます。

図 5-23 トリガー条件の設定

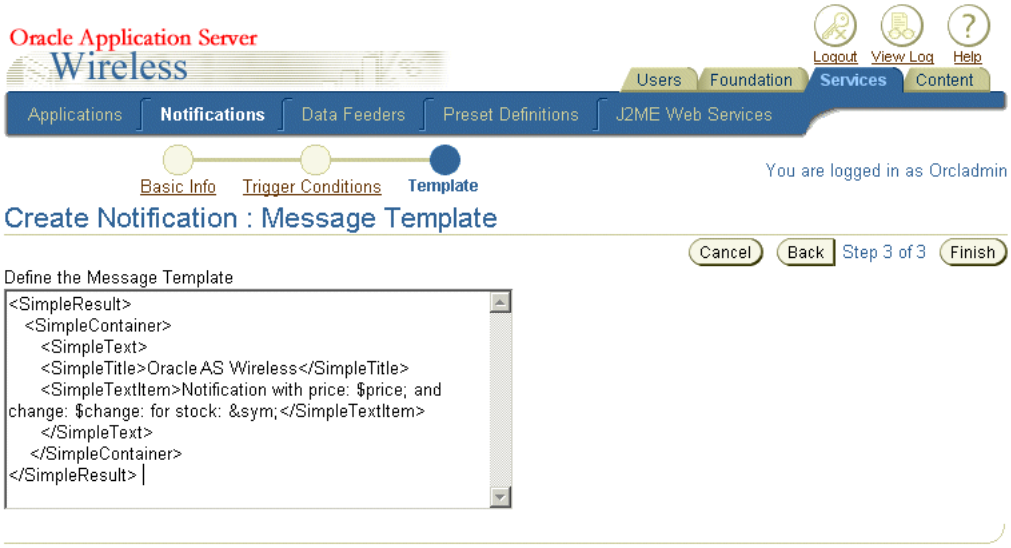


ステップ 3: メッセージ・テンプレートの作成

「メッセージ・テンプレート」画面（図 5-24）では、SimpleText のスタイルシートを入力してメッセージを作成できます。このスタイルシートでは、データ・フィーダの出力値は動的値です。次のスタイルシートでは、これらの値は sym、price および change で表されています。

```
<SimpleResult>
   <SimpleContainer>
      <SimpleText>
      <SimpleTitle>OracleAS Wireless</SimpleTitle>
      <SimpleTextItem>Notification with price: $price; and change: $change: for
stock: &sym;</SimpleTextItem>
      </SimpleText>
    </SimpleContainer>
</SimpleResult>
```

図 5-24「メッセージ・テンプレート」画面

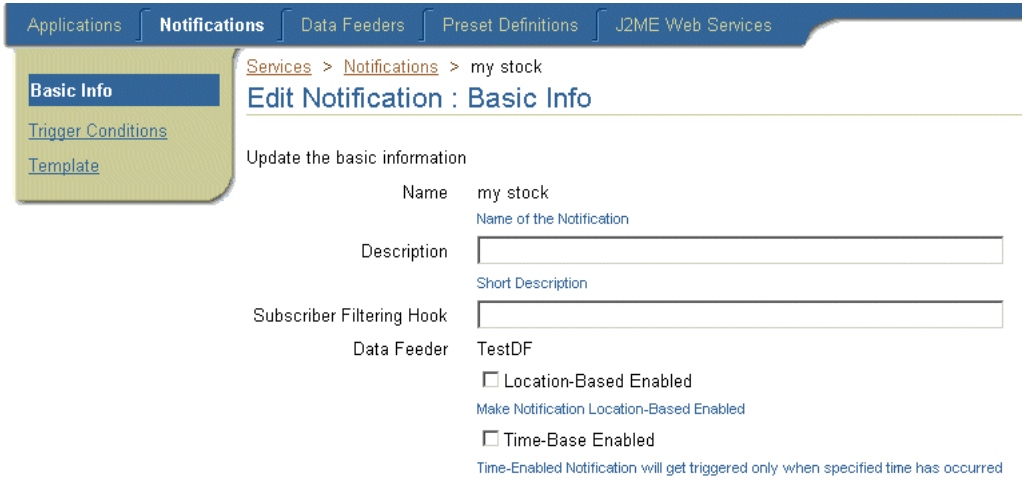


注意： OracleAS Wireless では、入力した値はウィザード全体を完了するまでコミットされません。

通知の編集

通知の参照画面の「編集」ボタンを使用すると、通知の基本構成パラメータ、トリガー条件およびメッセージ・テンプレートを編集できます。通知を編集するには、最初に参照画面から通知を選択して**「編集」**をクリックします。通知編集用の「基本情報」画面が、フィールドに選択した通知に設定された値が移入された状態で表示されます（図 5-25）。**「適用」**をクリックして変更内容を保存します。**「取消」**をクリックすると、値が元の状態に戻ります。通知のパラメータの定義方法は、5-44 ページの「マスター通知の作成」を参照してください。

図 5-25 通知編集用の「基本情報」画面



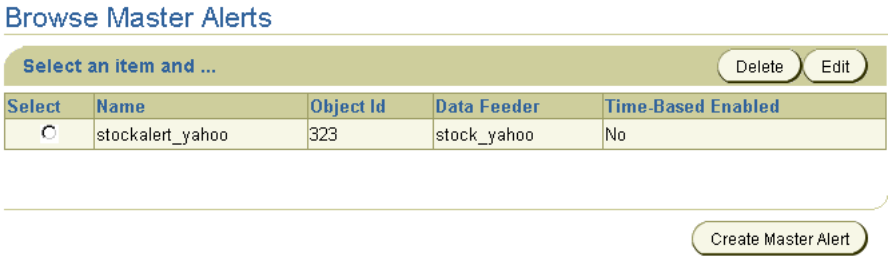
マスター・アラートの管理（使用中止）

サービス・マネージャの「アラート」タブでは、マスター・アラートを作成、編集および削除できます。「アラート」タブを選択すると、アラートの参照画面が表示され、現在のマスター・アラートのリストが、名前、OID、データ・フィーダおよび時間値の要素で示されます（図 5-26）。表 5-15 で、マスター・アラートの参照リストの要素を説明します。

表 5-15 マスター・アラートの参照画面の要素

要素	説明
名前	マスター・アラート名。
オブジェクト ID	データベース内のアラートの ID。
データ・フィーダ名	マスター・アラートに使用されるデータ・フィーダまたはコンテンツ・ソース。
時間ベース可能	事前定義済時間で表示されるアラートを示します。

図 5-26 マスター・アラートの参照画面



マスター・アラートの作成

マスター・アラート作成ウィザードでは、マスター・アラートの作成方法がステップごとに示されます。このウィザードは、マスター・アラートの参照画面で**マスター・アラートの作成**ボタンをクリックすると起動され、プロセスの各ステップごとに別々の画面を表示します。マスター・アラートは、システム・マネージャが、アラート・エンジン・プロセスとデータ・フィーダ・エンジン・プロセスの両方を起動するとアクティブになります。

ステップ 1: マスター・アラートの基本構成パラメータの入力

マスター・アラート作成の最初の画面である「基本情報」画面では、マスター・アラートの基本構成パラメータを入力します（図 5-27）。

図 5-27 マスター・アラート作成ウィザードの「基本情報」画面

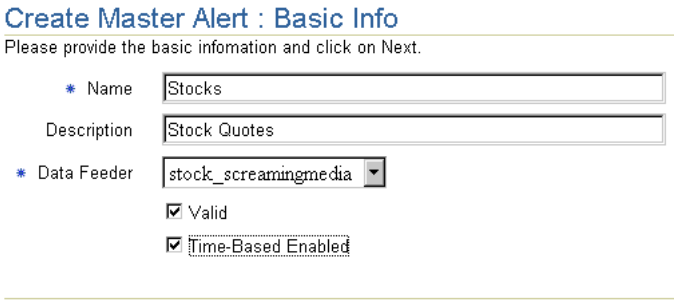


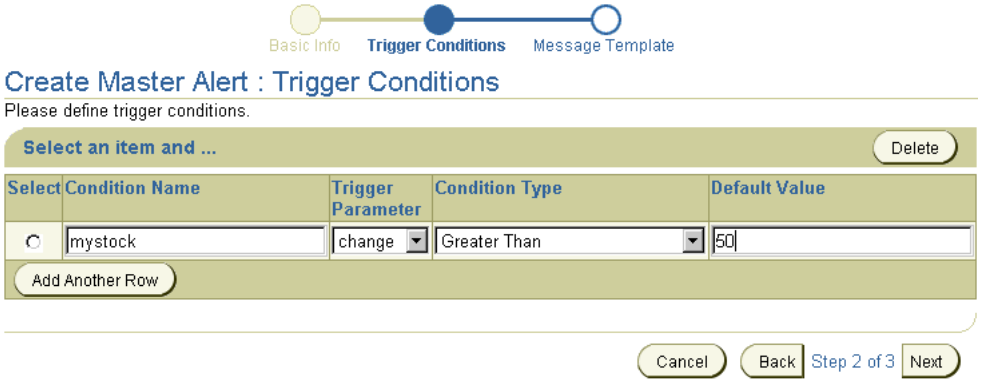
表 5-16 で、「基本情報」画面のパラメータを説明します。

表 5-16 マスター・アラートの基本構成パラメータ

パラメータ	値
名前	アラート名。これは必須パラメータです。
説明	アラートの説明。
サブスクライバ・フィルタリング・フック	Java クラス名。このフックを使用すると、これらのアラートがメッセージング・サーバーに送信される前に、限定されたアラートの対象となるサブスクライバをフィルタ処理で除外できます。
データ・フィーダ名	データ・フィード・ソースのドロップダウン・リスト。これは必須パラメータです。
時間ベース可能	このアラートが事前定義済時間でトリガーされるかどうかを指定します。頻度の選択肢は、毎日、平日および週末です。タイム・ゾーン情報は、ユーザー・プロファイルから取得されます。

「次」をクリックします。「トリガー条件」画面が表示されます（図 5-28）。

図 5-28 マスター・アラート作成ウィザードの「トリガー条件」画面



ステップ 2: マスター・アラートのトリガー条件の設定

「トリガー条件」画面では、エンド・ユーザーのデバイスでアラートを起動する条件をエンド・ユーザーが設定できるようにできます。たとえば、株価についてユーザーに通知するアラートを作成する場合は、株価が特定の価格を上回ったときまたは下回ったときにエンド・ユーザーが通知を要求できるアラート条件を設定できます。表 5-17 で、「トリガー条件」画面のパラメータを説明します。

表 5-17 マスター・アラートのトリガー条件

パラメータ	値
条件名	マスター・アラートに対するアラート・トリガーの名前。トリガー名には、英数字とアンダースコアのみ使用でき、30 文字以内で設定する必要があります。さらに、最初の文字には数字を使用できず、また SQL 予約語は使用できません。エンド・ユーザーがアラート・サービスにサブスクライブした場合にこのラベルが表示されます。
トリガー・パラメータ	トリガー・パラメータは、トリガー条件を定義するデータ・フィーダ内の要素です。たとえば、株価アラート・サービス用のデータ・フィーダに stock price という出力パラメータが含まれている場合は、条件名に対するトリガー・パラメータとして stock price を選択できます。データ・フィーダの出力パラメータの設定方法は、5-69 ページの「データ・フィーダの出力パラメータの編集」を参照してください。
条件タイプ	エンド・ユーザーが設定する値と関連して、アラートをトリガーする条件。
デフォルト値	パラメータのデフォルト値。デフォルト値を指定すると、OracleAS Wireless はユーザーに値の入力を求めません。デフォルト値は、コンテンツ・マネージャによって作成されるアプリケーションで指定された値で上書きでき、パラメータがユーザーに表示される場合は、ユーザーが OracleAS Wireless Customization を使用して上書きできます。

トリガー条件の選択

トリガー条件を選択する手順は、次のとおりです。

1. トリガー条件のリストから、トリガー条件を選択します。
2. 「条件タイプ」、「トリガー・パラメータ」または「デフォルト値」フィールドを必要に応じて編集します。
3. **「適用」**をクリックします。

新規トリガー条件の追加

新規トリガー条件を追加する手順は、次のとおりです。

1. 「条件」フィールドにトリガー条件名を入力します。
2. 「キャプション」フィールドに、エンド・ユーザーに入力を求めるプロンプトを表示するときに使用するテキストを入力します。
3. 「トリガー・パラメータ」フィールドのドロップダウン・リストからトリガー・パラメータを選択します。
4. 「条件タイプ」フィールドのドロップダウン・リストから条件タイプを選択します。条件タイプは、トリガー・パラメータのデータ型によって異なります。

   データ型が数値の場合は、次の条件が含まれます。

   - 次より小さい
   - 次より大きい
   - 等しい
   - 以下
   - 以上
   - 絶対値より小さい
   - 絶対値より大きい
   - 絶対値に等しい
   - 絶対値以下
   - 絶対値以上
   - 値の変更（このタイプの条件値は、0（ゼロ）または1のみです。0（ゼロ）はトリガーされないことを意味し、1は値が変更されたときにトリガーされることを意味します。デフォルト値は0（ゼロ）です。）

データ型がテキストの場合は、次の条件タイプが含まれます。

- 完全一致
- 不一致
- 含む
- 含まない
- 先頭
- 最後
- 値の変更（このタイプの条件値は、0（ゼロ）または1のみです。0（ゼロ）はトリガーされないことを意味し、1は値が変更されたときにトリガーされることを意味します。デフォルト値は0（ゼロ）です。）

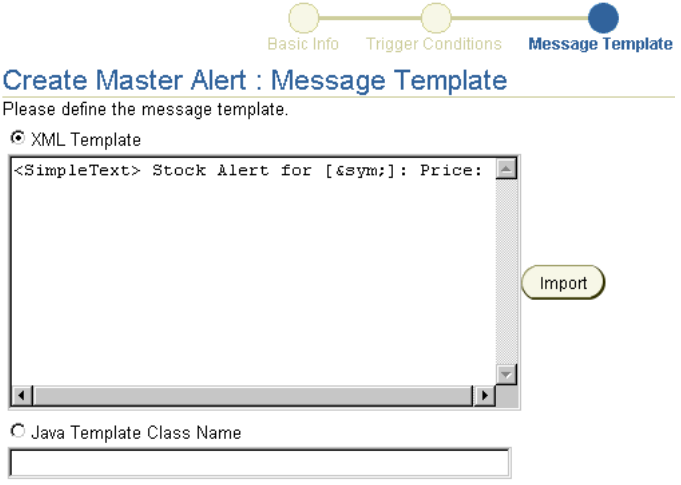
5. 「デフォルト値」フィールドにトリガー条件のデフォルト値を入力します。
6. **「追加」**をクリックします。
7. **「次」**をクリックします。「メッセージ・テンプレート」画面が表示されます。

ステップ3: マスター・アラートのメッセージ・テンプレートの作成

「メッセージ・テンプレート」画面（図5-29）では、メッセージ・テンプレートのインポート、またはフックの指定ができます。データ・フィーダの出力値は、SimpleTextのスタイルシートでは動的値です。次のスタイルシートでは、これらの値は`&price`および`&change`で表されています。

```
<SimpleText> Stock Alert for [&sym;]: Price: &price; Change:
&change;</SimpleText>
```

図 5-29 マスター・アラート作成ウィザードの「メッセージ・テンプレート」画面



メッセージ・テンプレートのインポート

メッセージ・テンプレートをインポートする手順は、次のとおりです。

1. 「メッセージ・テンプレート」ラジオ・ボタンを選択します。
2. **「インポート」**をクリックして、ローカル・ファイル・システムからメッセージ・テンプレートを取得します。
3. **「次」**をクリックして、マスター・アラートの作成を完了します。

注意： OracleAS Wireless では、入力した値はウィザード全体を完了するまでコミットされません。

フックの指定

プログラミング・フックを指定してメッセージ・テンプレートを作成する手順は、次のとおりです。

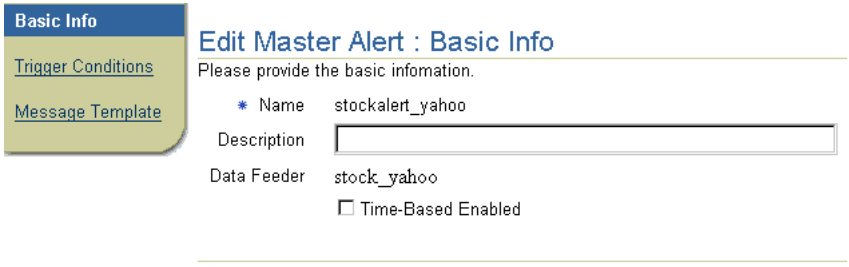
1. 「Java テンプレート・クラス名」を選択します。
2. フック名を入力します。
3. **「次」**をクリックして、マスター・アラートの作成を完了します。

マスター・アラートの編集

マスター・アラートの参照画面の「編集」ボタンを使用すると、マスター・アラートの基本構成パラメータ、トリガー条件およびメッセージ・テンプレートを編集できます。

マスター・アラートの基本構成パラメータを編集するには、参照画面からマスター・アラートを選択して**「編集」**をクリックします。マスター・アラート編集用の「基本情報」画面が、選択したマスター・アラートに設定された値がフィールドに移入された状態で表示されます（図 5-30）。必要に応じて基本構成値を編集します。マスター・アラートの基本構成パラメータの詳細は、5-51 ページの「ステップ 1: マスター・アラートの基本構成パラメータの入力」を参照してください。**「OK」**をクリックして変更内容を保存します。**「取消」**をクリックすると、値が元の状態に戻り、マスター・アラートの参照画面に戻ります。

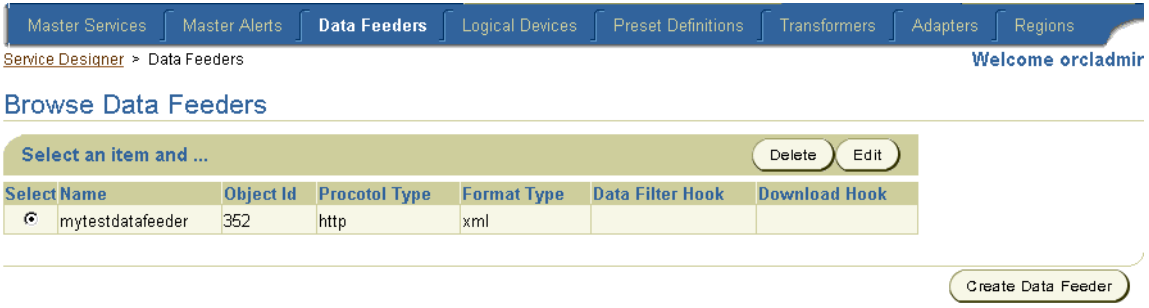
図 5-30 マスター・アラート編集用の「基本情報」画面



データ・フィーダの管理

サービス・マネージャの「データ・フィーダ名」タブ（図 5-31）では、データ・フィーダを作成、編集および削除できます。データ・フィーダは、OracleAS Wireless のオブジェクトで、内部または外部のコンテンツ・ソースからデータをダウンロードし、OracleAS Wireless モバイル・アラート用の共通フォーマットに変換します。

図 5-31 データ・フィーダの参照画面



「**データ・フィーダ名**」タブをクリックすると、データ・フィーダの参照画面が表示され、現在のデータ・フィーダがリストされます。表 5-18 で、データ・フィーダ・リストの要素を説明します。

表 5-18 データ・フィーダの参照画面の要素

要素	説明
名前	データ・フィーダ名。
オブジェクト ID	リポジトリ内のデータ・フィーダのオブジェクト ID（OID）。
プロトコル・タイプ	コンテンツ・プロバイダにアクセスし、データを取得するためにデータ・フィーダによって使用されるプロトコル。
フォーマット・タイプ	取得したコンテンツのデータ・フォーマット・タイプ。フォーマットには、区切り記号付きテキスト（カンマで区切られた値など）、XML および固定列テキストがあります。
データ・フィルター・フック	データ保存前の後処理を可能にする `DataFeedFilterHook` を実装する Java クラス名。
フックのダウンロード	`FeedDownloadHook` を実装する Java クラス名。この Java インタフェースを実装すると、ダウンロード時にダウンロード URL または POST ページを構築できます。

データ・フィーダの作成

データ・フィーダ作成ウィザードを使用すると、データ・フィーダを作成できます。このウィザードはデータ・フィーダの参照画面で**「データ・フィーダの作成」**ボタンをクリックすると起動され、プロセスの各ステップごとに別々の画面を表示して、データ・フィーダの作成方法を順にガイドします。データ・フィーダを作成した後は、そのデータ・フィーダをマスター・アラートに割り当てることができます。データ・フィーダ（および結果としてデータ・フィーダを使用してそのコンテンツを導出するアラート）は、システム・マネージャがデータ・フィーダ・プロセスを起動するまでアクティブになりません。

ステップ 1: データ・フィーダに関する基本情報の入力

データ・フィーダ作成ウィザードの「基本情報」画面（図 5-32）では、データ・フィーダの基本プロパティを入力できます。

図 5-32 データ・フィーダ作成ウィザードの「基本情報」画面

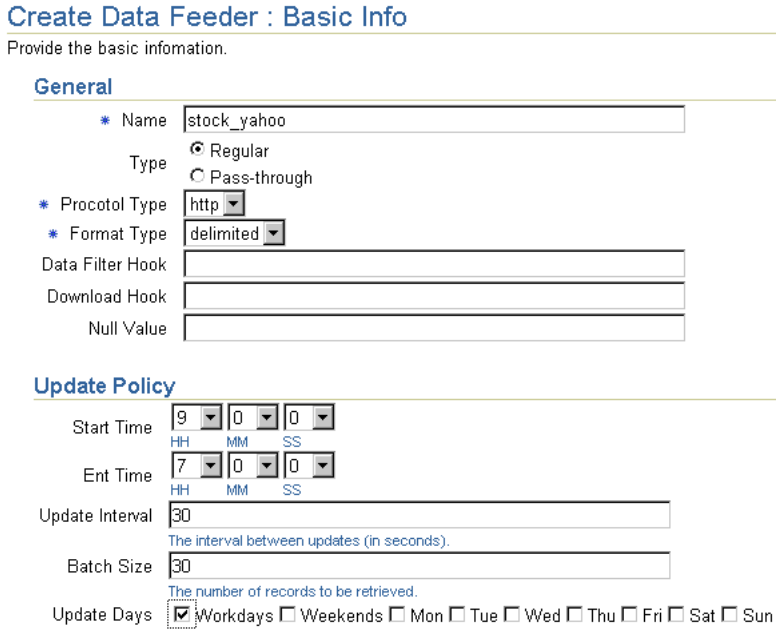


表 5-19 で、データ・フィーダ作成ウィザードの「基本情報」画面のパラメータを説明します。

表 5-19 データ・フィーダ作成ウィザードの「基本情報」画面のパラメータ

パラメータ	値
名前	コンテンツ・プロバイダ名。これは必須パラメータです。
タイプ	組込みのデータ取得フレームワークを使用してデータをプルする場合は、「標準」を選択します。Java クラスを使用してデータを取得するプッシュ・アプリケーションの場合は、「パススルー」を選択します。「パススルー」を選択した場合は、Java クラスを指定する必要があります。これは必須パラメータです。
プロトコル・タイプ	コンテンツ・プロバイダにアクセスし、データを取得するためにデータ・フィーダによって使用されるプロトコル。ドロップダウン・メニューには次のオプションがあります。 ■ sql: SQL データベース。指定したデータ・フィードに対して SQL 問合せを実行し、出力を読み取ります。 ■ app: ローカル・アプリケーション。アプリケーションをサブプロセスとして実行し、.stdout ファイルを読み取ります。 ■ http: HTTP。URL を構築し、リモート Web サイトで GET/POST を実行し、必要に応じて認証を行います。 ■ ftp: FTP。リモート Web サーバーに接続し、認証を行ってファイルをダウンロードします。ユーザー名とパスワードが必要です。 ■ file: ローカル・ファイル。ファイル・システムの任意のファイルから読み取ります。
フォーマット・タイプ	取得したコンテンツのデータ・フォーマット・タイプ。ドロップダウン・メニューには次のオプションがあります。 ■ delimited: 区切り記号付きテキストを解析します。デフォルトの区切り文字はカンマ（,）です。 ■ fixed: 固定列テキスト。固定列の位置で区切られたテキストを解析します。 ■ xml: 優先入力フォーマット。
データ・フィルター・フック	Java クラス名。このオプションを使用すると、プロバイダから提供された単一列を 2 つの列に分割したり、コンテンツ・データをコンテンツ・キャッシュ表にフィードする前にデータをフィルタ処理で除外するなど、追加のロジック用にデータ・フィーダをカスタマイズできます。
フックのダウンロード	Java クラス名。このオプションを使用すると、データをダウンロードする新規 URL を生成することによって、データ・フィーダをカスタマイズできます。
NULL 値	適用不可のデータのマーク付けに使用される N/A などの文字列。プロバイダによって使用される文字列が異なります。
開始時刻	データのダウンロードを開始する時刻。
終了時刻	データのダウンロードを終了する時刻。

表 5-19 データ・フィーダ作成ウィザードの「基本情報」画面のパラメータ（続き）

パラメータ	値
更新間隔	ダウンロードの間の間隔（秒）。1 日に 1 回ダウンロードする場合は、この値を「0（ゼロ）」に設定します。
バッチ・サイズ	ダウンロードのバッチ・サイズ。サイズを「1」に設定すると、OracleAS Wireless では、一度に 1 個のパラメータがダウンロードされます。サイズを「10」に設定すると、一度に 10 個のパラメータがダウンロードされます。
更新日	データの更新指定日。

「次」をクリックします。「初期パラメータ」画面が表示され、選択したプロトコル・タイプの初期パラメータが示されます。

ステップ 2: データ・フィーダの初期パラメータの入力

「初期パラメータ」画面には、5-59 ページの「ステップ 1: データ・フィーダに関する基本情報の入力」で選択したプロトコル・タイプとフォーマット・タイプに固有の初期パラメータが表示されます。表 5-20 で、これらの初期パラメータを説明します。

表 5-20 データ・フィーダ・プロトコルの初期パラメータ

パラメータ	説明
HTTP プロトコルには、次の初期パラメータが含まれています。	
HTTP URI	コンテンツ・ソースの HTTP アドレスのフルパス。
ユーザー名	ユーザー名。保護サイトからデータを取得する場合はこの値を入力します。
パスワード	パスワード。保護サイトからデータを取得する場合はこの値を入力します。
HTTP メソッド	GET メソッドまたは POST メソッドのいずれかを選択します。
ファイル・プロトコルには、次の初期パラメータが含まれています。	
ファイル・パス	c:¥temp¥file.txt などのファイル・パス。
FTP プロトコルには、次の初期パラメータが含まれています。	
FTP URI	FTP リクエストのパス。
ユーザー名	ユーザー名。
パスワード	パスワード。
FTP モード	「テキスト」モードまたは「バイナリ」モードのいずれかを選択します。
SQL プロトコルには、次の初期パラメータが含まれています。	
接続文字列	データベース接続文字列。

表 5-20 データ・フィーダ・プロトコルの初期パラメータ（続き）

パラメータ	説明
問合せ	SQL 問合せ。
ファイル・プロトコルには、次の初期パラメータが含まれています。	
ファイル・パス	コンテンツ・ソースのファイル・パス。

HTTP プロトコルの初期パラメータの入力

HTTP プロトコルと XML フォーマット・タイプを使用するデータ・フィーダの初期パラメータを入力する手順は、次のとおりです。

1. コンテンツ・ソースの HTTP URI を入力します。
2. ユーザー名を入力します。
3. パスワードを入力します。
4. GET または POST HTTP メソッドのいずれかを選択します。
5. フィードによって XML が受け入れられる場合は、XML を標準のフィード XML フォーマットに変換する XSL スタイルシートをインポートする必要があります。
6. **「次」**をクリックします。「入力パラメータ」画面が表示されます。

HTTP プロトコルと区切り記号付きフォーマットを使用するデータ・フィーダの初期パラメータを入力する手順は、次のとおりです。

1. コンテンツ・ソースの HTTP URI を入力します。
2. ユーザー名を入力します。
3. パスワードを入力します。
4. フォーマット・タイプの区切り文字を入力します。たとえば、カンマ（,）を入力します。
5. 選択したフォーマット・タイプの引用符文字を入力します。たとえば、引用符（"）を入力します。
6. **「次」**をクリックします。「入力パラメータ」画面が表示されます。

HTTP プロトコルと固定列フォーマットを使用するデータ・フィーダの初期パラメータを入力する手順は、次のとおりです。

1. コンテンツ・ソースの HTTP URI を入力します。
2. ユーザー名を入力します。
3. パスワードを入力します。

4. GET または POST HTTP メソッドのいずれかを選択します。
5. **「次」**をクリックします。「入力パラメータ」画面が表示されます。

ファイル・プロトコルの初期パラメータの入力

ファイル・プロトコルと XML フォーマットを使用するデータ・フィーダの初期パラメータを入力する手順は、次のとおりです。

1. ファイル・パスを入力します。たとえば、「c:¥temp¥file.txt」と入力します。
2. フィードによって XML が受け入れられる場合は、XML を標準の XML に変換する XSL スタイルシートをインポートする必要があります。
3. **「次」**をクリックします。「入力パラメータ」画面が表示されます。

ファイル・プロトコルと区切り記号付きフォーマットを使用するデータ・フィーダの初期パラメータを入力する手順は、次のとおりです。

1. ファイル・パスを入力します。
2. フォーマット・タイプの区切り文字を入力します。たとえば、カンマ（,）を入力します。
3. 選択したフォーマット・タイプの引用符文字を入力します。たとえば、引用符（"）を入力します。
4. **「次」**をクリックします。「入力パラメータ」画面が表示されます。

ファイル・プロトコルと固定列フォーマットを使用するデータ・フィーダの初期パラメータを入力する手順は、次のとおりです。

1. ファイル・パスを入力します。
2. **「次」**をクリックします。「入力パラメータ」画面が表示されます。

FTP プロトコルの初期パラメータの入力

FTP プロトコルと XML フォーマットを使用するデータ・フィーダの初期パラメータを入力する手順は、次のとおりです。

1. FTP URI を入力します。
2. ユーザー名を入力します。
3. パスワードを入力します。
4. 「テキスト」モードまたは「バイナリ」FTP モードのいずれかを選択します。
5. フィードによって XML が受け入れられる場合は、XML を標準の XML に変換する XSL スタイルシートをインポートする必要があります。
6. **「次」**をクリックします。「入力パラメータ」画面が表示されます。

FTP プロトコルと区切り記号付きフォーマットを使用するデータ・フィーダの初期パラメータを入力する手順は、次のとおりです。

1. FTP URI を入力します。
2. ユーザー名を入力します。
3. パスワードを入力します。
4. 「テキスト」モードまたは「バイナリ」モードのいずれかを選択します。
5. フォーマット・タイプの区切り文字を入力します。たとえば、カンマ（,）を入力します。
6. 選択したフォーマット・タイプの引用符文字を入力します。たとえば、引用符（"）を入力します。
7. **「次」**をクリックします。「入力パラメータ」画面が表示されます。

FTP プロトコルと固定列フォーマットを使用するデータ・フィーダの初期パラメータを入力する手順は、次のとおりです。

1. FTP URI を入力します。
2. ユーザー名を入力します。
3. パスワードを入力します。
4. 「テキスト」モードまたは「バイナリ」FTP モードのいずれかを選択します。
5. **「次」**をクリックします。「入力パラメータ」画面が表示されます。

SQL プロトコルの初期パラメータの入力

SQL プロトコルと XML フォーマットを使用するデータ・フィーダの初期パラメータを入力する手順は、次のとおりです。

1. 接続文字列を入力します。
2. SQL 問合せを入力します。
3. フィードによって XML が受け入れられる場合は、XML を標準の XML に変換する XSL スタイルシートをインポートする必要があります。
4. **「次」**をクリックします。「入力パラメータ」画面が表示されます。

SQL プロトコルと区切り記号付きフォーマットを使用するデータ・フィーダの初期パラメータを入力する手順は、次のとおりです。

1. 接続文字列を入力します。
2. 問合せを入力します。
3. フォーマット・タイプの区切り文字を入力します。たとえば、カンマ（,）を入力します。

4. 選択したフォーマット・タイプの引用符文字を入力します。たとえば、引用符（"）を入力します。
5. **「次」**をクリックします。「入力パラメータ」画面が表示されます。

SQL プロトコルと固定列フォーマットを使用するデータ・フィーダの初期パラメータを入力する手順は、次のとおりです。

1. 接続文字列を入力します。
2. 問合せを入力します。
3. **「次」**をクリックします。「入力パラメータ」画面が表示されます。

アプリケーション・プロトコルの初期パラメータの入力

アプリケーション・プロトコルと XML フォーマットを使用するデータ・フィーダの初期パラメータを入力する手順は、次のとおりです。

1. コマンドラインを入力します。
2. フィードによって XML が受け入れられる場合は、XML を標準の XML に変換する XSL スタイルシートをインポートする必要があります。
3. **「次」**をクリックします。「入力パラメータ」画面が表示されます。

アプリケーション・プロトコルと区切り記号付きフォーマットを使用するデータ・フィーダの初期パラメータを入力する手順は、次のとおりです。

1. コマンドラインを入力します。
2. フォーマット・タイプの区切り文字を入力します。たとえば、カンマ（,）を入力します。
3. 選択したフォーマット・タイプの引用符文字を入力します。たとえば、引用符（"）を入力します。
4. **「次」**をクリックします。「入力パラメータ」画面が表示されます。

アプリケーション・プロトコルと固定列フォーマットを使用するデータ・フィーダの初期パラメータを入力する手順は、次のとおりです。

1. コマンドラインを入力します。
2. **「次」**をクリックします。「入力パラメータ」画面が表示されます。

ステップ 3: データ・フィーダの入力パラメータの入力

「入力パラメータ」では、データ・フィーダの入力パラメータを入力できます。「入力パラメータ」画面には、5-59 ページの「ステップ 1: データ・フィーダに関する基本情報の入力」で選択したフォーマット・タイプに固有の入力パラメータが表示されます。表 5-21 で、データ・フィーダの入力パラメータを説明します。

表 5-21 データ・フィーダの入力パラメータ

入力パラメータ	説明
内部名	キャッシュ表の列、およびアラート・フレームワークにおける条件の設定のために、このパラメータに対して内部的に使用される名前。
データ型	ドロップダウン・リストには次のオプションがあります。 ■ NUMBER: 数値入力用。 ■ TEXT_30: 最大 30 文字までのテキスト。 ■ TEXT_80: 最大 80 文字までのテキスト。 ■ TEXT_150: 最大 150 文字までのテキスト。 ■ TEXT_800: 最大 800 文字までのテキスト。 ■ TEXT_1200: 最大 1200 文字までのテキスト。
外部名	外部プロバイダへのマッピング。
列番号	区切り記号付き値の列番号。この入力パラメータは区切り記号付きフォーマットに固有です。
開始位置	値の開始列。この入力パラメータは固定列パラメータに固有です。
終了位置	値の終了列。この入力パラメータは固定列パラメータに固有です。
キャプション	エンド・ユーザーがアラートにサブスクライブした場合に表示されるキャプション。例: Stock Symbol。
デフォルト値	パラメータのデフォルト値。

入力パラメータを入力する手順は、次のとおりです。

1. **「1 行追加」**をクリックします。行が表示されます。
2. 行に次のように入力します。
   a. 内部名を入力します。
   b. データ型を入力します。
   c. 外部名を入力します。
   d. 列番号を入力します。このパラメータは区切り記号付きフォーマットに固有です。

e. 開始位置を入力します。このパラメータは固定列フォーマットに固有です。

f. 終了位置を入力します。このパラメータは固定列フォーマットに固有です。

g. キャプションを入力します。

h. デフォルト値を入力します。

3. **「次」**をクリックします。「出力パラメータ」画面が表示されます。

ステップ 4: データ・フィーダの出力パラメータの入力

「出力パラメータ」画面では、データ・フィーダの出力パラメータを入力できます。「出力パラメータ」画面には、5-59 ページの「ステップ 1: データ・フィーダに関する基本情報の入力」で選択したフォーマット・タイプに固有のパラメータが表示されます。出力パラメータ（表 5-22 の説明を参照）は、コンテンツ・プロバイダから取得されたデータです。データ・フィーダの出力パラメータにアラートを設定します。

表 5-22 データ・フィーダの出力パラメータ

出力パラメータ	説明
内部名	キャッシュ表の列およびアラート・フレームワークの条件の設定のために、このパラメータに対して内部的に使用される名前。
データ型	ドロップダウン・リストには次のオプションがあります。 ■ NUMBER: 数値入力用。 ■ TEXT_30: 最大 30 文字までのテキスト。 ■ TEXT_80: 最大 80 文字までのテキスト。 ■ TEXT_150: 最大 150 文字までのテキスト。 ■ TEXT_800: 最大 800 文字までのテキスト。 ■ TEXT_1200: 最大 1200 文字までのテキスト。
外部名	外部プロバイダへのマッピング。
列番号	区切り記号付き値の列番号。この出力パラメータは区切り記号付きフォーマットに固有です。
開始位置	値の開始列。この出力パラメータは固定列パラメータに固有です。
終了位置	値の終了列。この出力パラメータは固定列パラメータに固有です。
キャプション	OracleAS Wireless によってパラメータに使用されるラベル。エンド・ユーザーがアラート・サービスにサブスクライブした場合にこのラベルが表示されます。

出力パラメータを入力する手順は、次のとおりです。

1. **「1 行追加」**をクリックします。行が表示されます。
2. 行に次のように入力します。
   a. 内部名を入力します。
   b. データ型を選択します。
   c. 外部名を入力します。
   d. 列番号を入力します。このパラメータは区切り記号付きフォーマットに固有です。
   e. 開始位置を入力します。このパラメータは固定列フォーマットに固有です。
   f. 終了位置を入力します。このパラメータは固定列フォーマットに固有です。
   g. キャプションを入力します。
3. **「終了」**をクリックしてデータ・フィーダの作成を完了します。データ・フィーダの参照画面が再表示され、新規データ・フィーダが表示されます。

データ・フィーダの編集

データ・フィーダの参照画面の「編集」ボタンを使用すると、データ・フィーダの基本構成、初期パラメータ、入力パラメータおよび出力パラメータを編集できます。

データ・フィーダの基本構成の編集

データ・フィーダの基本構成を編集する手順は、次のとおりです。

1. データ・フィーダの参照画面から、編集するデータ・フィーダを選択します。
2. **「編集」**をクリックします。
3. データ・フィーダの基本構成編集用の画面が、選択したデータ・フィーダに設定された値がフィールドに移入された状態で表示されます。
4. 必要に応じて値を編集します。データ・フィーダの基本構成パラメータの詳細は、5-59 ページの「ステップ 1: データ・フィーダに関する基本情報の入力」を参照してください。
5. **「OK」**をクリックして変更内容を保存します。**「取消」**をクリックすると、基本構成の値が元の状態にリセットされ、データ・フィーダの参照画面に戻ります。

データ・フィーダの初期パラメータの編集

データ・フィーダの初期パラメータを編集する手順は、次のとおりです。

1. メニューから、「初期パラメータ」を選択します。初期パラメータ編集用の画面が、選択したデータ・フィーダに設定された初期パラメータが移入された状態で表示されます。
2. 必要に応じて初期パラメータを編集します。データ・フィーダの初期パラメータの詳細は、5-61 ページの「ステップ 2: データ・フィーダの初期パラメータの入力」を参照してください。
3. 「**OK**」をクリックして変更内容を保存します。「**取消**」をクリックすると、初期パラメータの値が元の状態にリセットされ、データ・フィーダの参照画面に戻ります。

データ・フィーダの入力パラメータの編集

データ・フィーダの入力パラメータを編集する手順は、次のとおりです。

1. メニューから、「入力パラメータ」を選択します。入力パラメータ編集用の画面が、選択したデータ・フィーダに設定された値が移入された状態で表示されます。
2. 必要に応じて値を編集します。データ・フィーダの入力パラメータの詳細は、5-66 ページの「ステップ 3: データ・フィーダの入力パラメータの入力」を参照してください。
3. 「**OK**」をクリックして変更内容を保存します。「**取消**」をクリックすると、入力パラメータが元の状態に戻り、データ・フィーダの参照画面に戻ります。

データ・フィーダの出力パラメータの編集

データ・フィーダの出力パラメータを編集する手順は、次のとおりです。

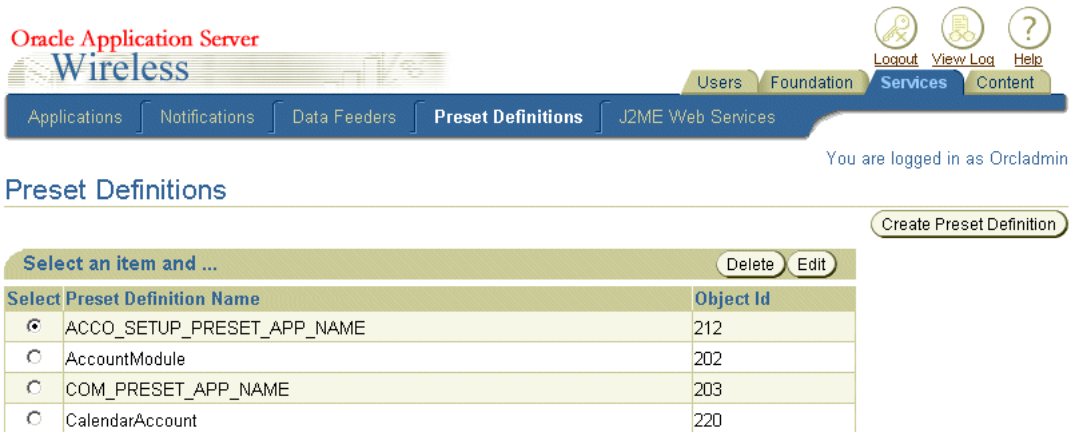
1. メニューから、「出力パラメータ」を選択します。出力パラメータ編集用の画面が、選択したデータ・フィーダに設定された値が移入された状態で表示されます。
2. 必要に応じて値を編集します。データ・フィーダの出力パラメータの詳細は、5-67 ページの「ステップ 4: データ・フィーダの出力パラメータの入力」を参照してください。
3. 「**OK**」をクリックして変更内容を保存します。「**取消**」をクリックすると、出力パラメータが元の状態に戻り、データ・フィーダの参照画面に戻ります。

プリセット定義の管理

プリセット定義を使用すると、ユーザーは独自の入力パラメータを入力することによってアプリケーションをパーソナライズできます。ユーザーがアプリケーションをリクエストすると、アプリケーションによってユーザー定義の入力パラメータ（プリセット）がロードされます。通常、これらのプリセットは、アプリケーションがユーザーに対してリスト表示でき、ユーザーは、項目を選択してそのアプリケーションを実行する必要があります。

注意： プリセット定義はユーザー・グループ内のすべてのユーザーがアクセス可能です。

図 5-33 プリセット定義の参照画面



「プリセット定義」タブを選択すると、デフォルトでプリセット定義の参照画面が表示され、現在のプリセット定義のリストが示されます（図 5-33）。この画面では、プリセット定義を作成、編集および削除できます。プリセット定義の参照画面には、次のパラメータが含まれています。

表 5-23 プリセット定義の参照画面のパラメータ

パラメータ	説明
プリセット定義名	プリセット定義名。
オブジェクト ID	データベースに格納されているオブジェクト ID。

プリセット定義の作成

サービス・マネージャを使用すると、プリセット定義を作成できます。この場合のサービス・マネージャは、定義済の各プリセット定義に値を追加できる 1 つのテンプレートです。ユーザーは、アプリケーションの起動時に、入力パラメータとしていずれかのプリセット定義から値を選択します。

新規プリセット定義を作成するには、参照画面で**「プリセット定義の作成」**ボタンをクリックします。「プリセット定義の作成」画面が表示されます（図 5-34）。この画面で、プリセット定義の一意名を入力します。さらに、このプリセット定義が Wireless Customization Portal のユーザー用ではない場合は、「システム・オブジェクト」を選択します。通常、プリセット定義は、ユーザーが独自のプリセット値を作成できるように Customization Portal に表示されます。この時点で**「終了」**をクリックしてプリセット定義の作成を完了するか、または 5-71 ページの「プリセット属性の追加」の説明に従ってプリセット属性を追加できます。

プリセット属性の追加

プリセット属性を使用すると、エンド・ユーザーが OracleAS Wireless サーバーに入力および保存する入力パラメータの関連を定義できます。「プリセットの作成」画面で**「追加」**ボタンをクリックして、表に属性を追加します。表示された空白行で、表 5-24 の説明に従って次のパラメータを定義します。

表 5-24 プリセット記述パラメータ

パラメータ	値
属性名	プリセット属性名。
説明	プリセット属性のオプションの説明。
値形式	テキストの場合は、正規表現 `org.apache.regexp.RE` と一致する任意の値を入力します。たとえば、数値の場合は `[:digit:]` のように入力します。 数値の場合は、`Java.text.DecimalFormat` の書式と一致する任意の値を入力します。たとえば、通貨の場合は `#,##0.0` のように入力します。

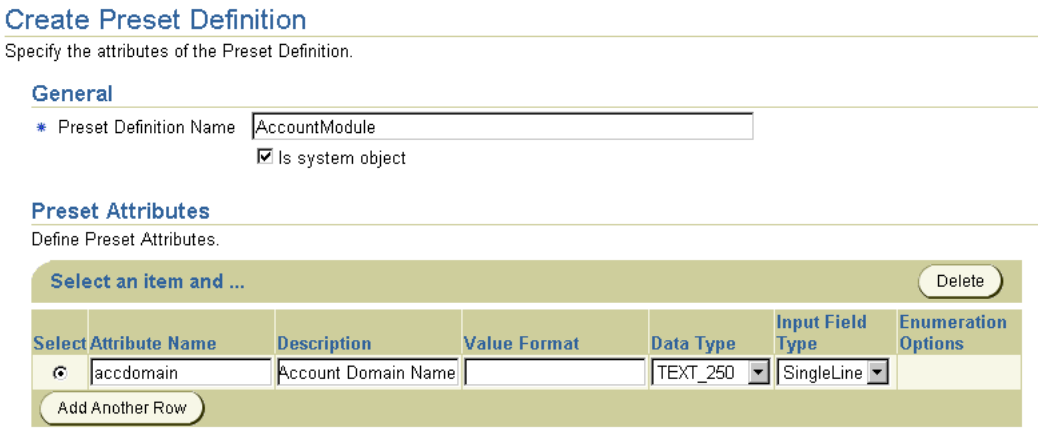
表 5-24 プリセット記述パラメータ（続き）

パラメータ	値
列型	ドロップダウン・リストには次のオプションがあります。 ■ NUMBER: 数値入力用。 ■ TEXT_30: 最大 30 文字までのテキスト。 ■ TEXT_80: 最大 80 文字までのテキスト。 ■ TEXT_150: 最大 150 文字までのテキスト。 ■ TEXT_250: 最大 250 文字までのテキスト。 ■ TEXT_500: 最大 500 文字までのテキスト。 ■ TEXT_800: 最大 800 文字までのテキスト。 ■ TEXT_1200: 最大 1200 文字までのテキスト。
入力フィールド・タイプ	次のプリセット・タイプの中から選択します。 ■ 単一行 : 名前など、単一行のエントリの場合に選択します。 ■ 複数行 : 番地など、複数行のエントリの場合に選択します。 ■ 列挙 : 表示および非表示などのエントリの場合に、条件を割り当てるために選択します。列挙オプションの詳細は、5-73 ページの「プリセット属性の列挙オプションの追加、編集および削除」を参照してください。

プリセットを追加した後、**「終了」**をクリックします。**「取消」**をクリックすると、すべての値が消去され、プリセット定義の参照画面に戻ります。

複数行のプリセット属性を追加すると、名前、番地、電話番号などの関連項目を定義できます。

図 5-34「プリセット定義の作成」画面



プリセット定義の編集

プリセット定義を編集するには、参照画面からプリセット定義を選択して**「編集」**をクリックします。「プリセット定義の編集」画面が表示されます。必要に応じてプリセット定義を編集します。プリセット記述子の詳細は、5-71 ページの「プリセット属性の追加」を参照してください。**「OK」**をクリックして変更内容をコミットします。プリセット定義の参照画面が再表示されます。

プリセット属性の列挙オプションの追加、編集および削除

「プリセット定義列挙オプションの編集」画面を使用すると、プリセット属性の列挙オプションを編集、追加または削除できます。

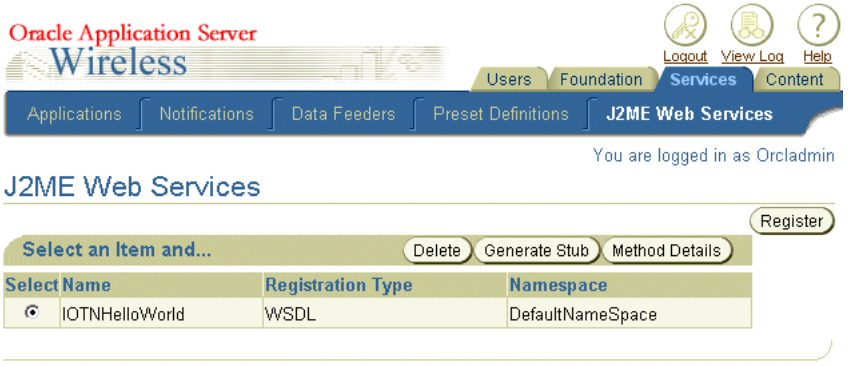
プリセット記述子の列挙オプションを編集する手順は、次のとおりです。

1. 「プリセット定義の作成」画面または「プリセット定義の編集」画面の「プリセット属性」セクションで、「列挙」を選択します。
2. **「編集」**をクリックします。「プリセット定義列挙オプションの編集」画面が表示されます。
3. 「プリセット定義列挙オプションの編集」画面で、必要に応じて次の操作を実行します。
   - ドロップダウン・リストから、編集または削除するオプションを選択します。
   - **「追加」**をクリックして新規列挙オプションを追加します。
4. **「完了」**をクリックします。「プリセット定義の作成」画面または「プリセット定義の編集」画面が再表示されます。

J2ME Web サービスの管理

J2ME Web サービスは、J2ME プロキシ・サーバーでホスティングされるサービスで、J2ME デバイスで実行されている J2ME MIDlet から起動されます。

図 5-35 J2ME Web サービスの参照画面



J2ME Web サービスの登録

J2ME Web サービスを登録するには、WSDL（Web Service Definition Language）URL、JAR ファイルの URL またはローカル JAR ファイルのいずれかを指定します。J2ME Web サービスの登録後、J2ME スタブ・クラスをダウンロードして、独自の J2ME MIDlet で使用します。サービス・マネージャの J2ME Web サービスの参照画面（図 5-35）では、Web サービスのメソッドの詳細を表示できます。

WSDL 経由の J2ME Web サービスの登録

J2ME Web サービスは、通常の Web サービスに基づいて登録できます。登録するには、最初に参照画面で**「登録」**をクリックします。「J2ME Web サービスを J2ME プロキシ・サーバーに登録」画面が表示されます（図 5-36）。この画面で、「WSDL 経由」オプションを選択し、通常の Web サービスの WSDL に対する URL を入力します。

JAR ファイル URL 経由の J2ME Web サービスの登録

JAR ファイルにパッケージ化されている通常の Java クラスに基づいて J2ME Web サービスを登録することもできます。この Java クラスは、ダウンロード用の Web サイトに置かれているか、または OracleAS Wireless Web サーバーに格納されています。J2ME Web サービスを登録するには、「JAR ファイル URL 経由」オプション（図 5-36 を参照）を選択して、JAR をダウンロードする Web サイトの URL、または OracleAS Wireless Web サーバー上の JAR ファイルの URL のいずれかを入力します。また、JAR ファイルにパッケージ化されている Java クラスのクラス名も指定する必要があります。

ローカル JAR ファイル経由の J2ME Web サービスの登録

「ローカル JAR ファイル経由」オプションを使用すると、クライアント・マシンに格納されている JAR ファイルを使用して J2ME Web サービスを登録できます。このときは、JAR ファイルを OracleAS Wireless サーバーにアップロードします。この登録を実行するには、「ローカル JAR ファイル経由」オプションを選択してから、**「インポート」**ボタンをクリックします。「インポート」ウィンドウで、「参照」機能を使用して JAR ファイルを検索し選択します。**「インポート」**をクリックしてローカル JAR ファイルをアップロードします。このオプションを使用するときは、ローカル JAR ファイルにパッケージ化されている Java クラスのクラス名も指定する必要があります。

J2ME Web サービスの名前空間の指定

J2ME プロキシ・サーバーでは、名前が競合しないように、名前空間別に J2ME Web サービスが格納されます。このため、J2ME Web サービスの登録時には、既存の名前空間を選択するか、または新規名前空間を入力できます。登録オプションを選択して名前空間を入力した後、**「終了」**をクリックすると J2ME Web サービスが登録されます。

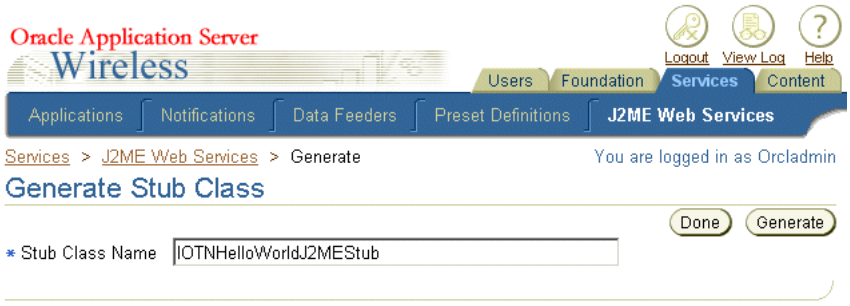
図 5-36 J2ME Web サービスの登録

スタブ・クラスの生成

J2ME Web サービスを MIDlet 内で使用できるように、MIDlet に J2ME スタブ・クラスを組み込む必要があります。

スタブ・クラスを組み込むには、参照ページから J2ME Web サービスを選択して、**「スタブの生成」**をクリックします。「スタブ・クラスの生成」画面（図 5-37）で、スタブ・クラス名を入力します。OracleAS Wireless では、この名前でスタブ・クラスが生成されます。スタブ・クラスをダウンロードした後、MIDlet でコンパイルします。

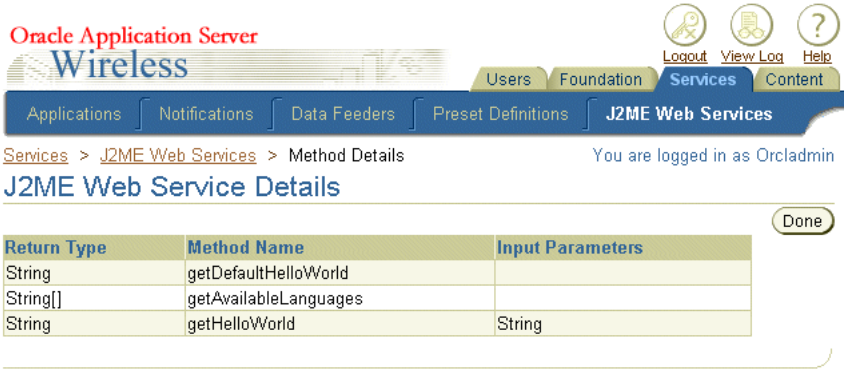
図 5-37 スタブ・クラスの生成



クラス・メソッドの詳細の表示

J2ME Web サービスを J2ME プロキシ・サーバーに登録した後は、Web サービスのメソッドの詳細を表示できます。メソッドの詳細を表示するには、参照ページから J2ME Web サービスを選択して、**「メソッドの詳細」**をクリックします。「J2ME Web サービスの詳細」画面が表示され（図 5-38）、名前、戻り型およびパラメータ・タイプなどのメソッドの詳細が示されます。

図 5-38 J2ME Web サービスのメソッド詳細の表示



J2ME Web サービスのコーディングの詳細は、第 12 章「J2ME の開発とプロビジョニング」の項を参照してください。

6

Mobile Studio

項ごとに様々なトピックを記載しています。各項の内容は、次のとおりです。

- 概要
- Mobile Studio のスタート・ガイド
- Mobile Studio のカスタマイズ

概要

この章では、OracleAS Wireless Mobile Studio の概要を説明します。Mobile Studio は、OracleAS Wireless プラットフォーム用のモバイル・アプリケーションを開発、テストおよびデプロイするための完全なオンライン・ホスティング環境です。また、Mobile Studio は Web ポータルとしても機能し、社内およびインターネット上で Wireless 開発者のコミュニティをサポートします。

Mobile Studio には単純かつ直感的で使用しやすい Web ベースのユーザー・インタフェースが用意されており、開発者は Wireless アプリケーションを迅速に構成、テストおよびデプロイできます。開発者が自分のワークステーションにダウンロードしたりインストールする必要はなく、単純に Web ブラウザを使用して Mobile Studio にアクセスします。アプリケーションを Mobile Studio に登録した後、開発者は任意のモバイル・デバイスまたはシミュレータ（音声など）を使用して、登録したアプリケーションをテストできます。テスト後ただちにリアルタイム・ログにアクセスし、問題をトラブルシューティングできます。アプリケーションのテストを終了した後、開発者は 1 回のボタン・クリックで、そのアプリケーションを本番サーバーにデプロイできます。

サービス・プロバイダは（外観、操作方法およびコンテンツをカスタマイズすることによって）Mobile Studio の特性を簡単に設定し、既存の Web サイトに統合できます。Mobile Studio は、相互開発ツールとして、OracleAS Wireless サーバー・プラットフォームの最新情報や付随情報を一度に取得する方法として、およびサード・パーティ・コンテンツ・プロバイダのためのサービス・デプロイ・ポータルとしての機能を担います。これにより、サービス・プロバイダは開発者コミュニティを容易にサポートし、新規の開発者を惹き付けることができます。

Mobile Studio の主な機能

Mobile Studio には、次の主要な機能が含まれています。

- 完全にオンライン化されたホスティング環境。ダウンロードするものはありません。
- アプリケーション開発者を対象とした単純な Web ベースのユーザー・インタフェース。これに対して、OracleAS Wireless Tools は、システム管理者と高度な開発者を対象としています。
- 任意のモバイル・デバイスやシミュレータ（音声など）から開発したアプリケーションへの即時アクセス。
- デバッグ・ログへの即時アクセスによるインタラクティブなテスト。
- 1 回のクリックで本番サーバーにデプロイできるオプション機能。

アプリケーション・プロバイダには、次の機能が用意されています。

- 既存の開発者をサポートしながら、新規の開発者を惹き付ける開発者ポータルとして機能。

- 複数の言語とキャラクタ・セットのデフォルトでのサポート。
- 対象はエンジニアではなく、Web マスター。したがって、アプリケーションの簡単なカスタマイズにコーディングは不要です。

Oracle Technology Network の Mobile Studio

Mobile Studio の特性を設定して既存の Web サイトに統合する方法の例は、Oracle Technology Network（http://studio.oraclemobile.com）でホスティングされている Mobile Studio にアクセスしてください。インターネットにアクセスできる開発者、システム・インテグレータまたは独立系ソフトウェア・ベンダーは、このインスタンスを使用して、すべてのデバイスから即時にアクセスできるモバイル・アプリケーションを迅速に作成およびテストできます。このユニークな環境によって、企業は開発所要時間の短縮、生産性の向上、飛躍的に簡略化されたテスト・サイクルなどのメリットが得られます。

Mobile Studio のスタート・ガイド

Mobile Studio のメイン・ページには、次の URL からアクセスします。

http://<*studio_server*>:<*studio_port*>/studio

<*studio_server*> と <*studio_port*> は、Mobile Studio インスタンスを実行しているホストの名前とポート番号です。これらは、インストール時に Oracle Installer によって設定されます。

注意： Mobile Studio は、一般的に普及している Netscape および Internet Explorer ブラウザの最新バージョンにあわせて最適化されています。Netscape 4.x と Internet Explorer 4.x では、Mobile Studio の動作保証はありません。

ログインと登録

Mobile Studio は、Single Sign-on（SSO）用の構成でデプロイされます。したがって、ユーザー・プロファイル情報（ユーザー ID とパスワードを含む）は、Oracle Internet Directory（OID）リポジトリに格納され、SSO 対応のすべてのアプリケーションによって共有されます。

すべてのユーザー・アカウントは、Oracle Internet Directory（OID）サーバーがサポートする中央のリポジトリで作成および管理されます。Mobile Studio が SSO 用に構成されると、共有リポジトリ内のすべてのユーザーは、それぞれの SSO ユーザー ID とパスワードでログインし、Mobile Studio を使用できます。新しいユーザーは、アカウントを作成してから Mobile Studio にアクセスする必要があります。

Mobile Studio を使用したアプリケーションの作成

OracleAS Wireless プラットフォームのアプリケーションを作成する最初の手順は、ユーザー自身の環境で、各自のツールを使用してアプリケーションを開発することです。アプリケーションのプレゼンテーション層の生成に使用するメカニズムは、Mobile Studio に対して透過的であり、動的ページ生成のテクノロジ（CGI、JSP、ASP など）のすべてがサポートされます。要件は、次の 2 点のみです。

- 生成するページは、XHTML など、OracleAS Wireless が認識できるマークアップ言語で記述する必要があります。
- アプリケーションへのエントリ・ポイントは、Mobile Studio サーバーからアクセス可能な HTTP URL であることが必要です。

次の例は、XHTML で記述した簡単な Hello World アプリケーションです。

図 6-1 簡単な Hello World の例

hello.jsp

```
<?xml version="1.0" encoding="UTF-8"?>

<%@ page contentType="application/vnd.wap.xhtml+xml; charset=UTF-8" %>
<html xmlns=http://www.w3.org/1999/xhtml>
  <head>
     <link rel="stylesheet" type="text/css" href="hello.css"/>
     <title>Hello World Example</title>
  </head>
  <body>
    <div>Hello World!</div>
 </body>
</html>
```

hello.css

```
body      { font: normal 9pt Arial, Helvetica, sans-serif; color: #FF0000 }
```

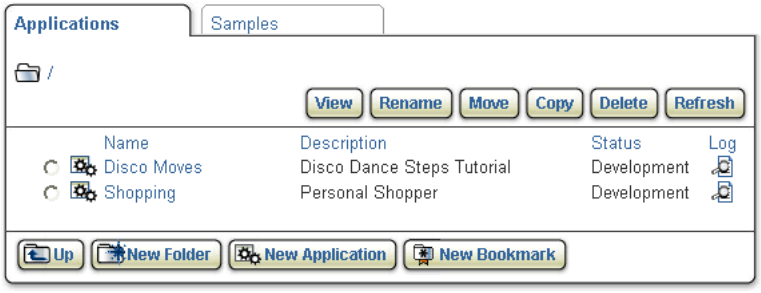
1. この 2 つのファイルを、Mobile Studio からアクセスできる Web サーバーにアップロードします。
2. Mobile Studio にログインします。
3. 「マイ Studio」ページで、**「新しいアプリケーション」**をクリックします。
4. アプリケーションの短縮名（Hello など）、URL、説明（必要に応じて）およびコメント（各自の参照用）を入力します。
5. **「作成」**をクリックして、新しいアプリケーションを Mobile Studio に登録します。

アプリケーションのテスト

アプリケーションを Mobile Studio に登録した後は、実際のモバイル・デバイスまたはデバイス・エミュレーション・ソフトウェアを使用して、そのアプリケーションをテストできます。Mobile Studio アプリケーションには、すべてのモバイル・デバイスからアクセスできるのみでなく、HTTP、音声、メッセージなどの複数のチャネルを介してアクセスできます。有効なアクセス・ポイント（HTTP アクセス用の URL や音声アクセス用の電話番号など）のリストは、Mobile Studio 管理者にお問い合せください。

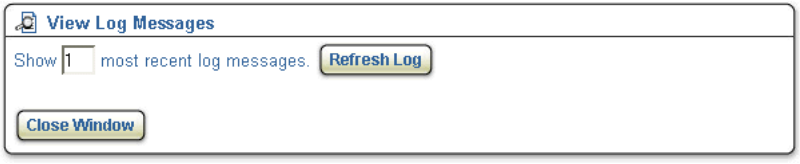
エラーが発生した（または疑いがある）場合は、アプリケーションの横にある**「ログ」**アイコンをクリックして、リアルタイム・デバッグ・ログを確認できます。

図 6-2 ウィンドウの例



「ログ」アイコンをクリックすると、ログ情報が新しいウィンドウに表示されます。

図 6-3「ログ・メッセージの表示」ウィンドウ



関連項目： ログの詳細は、Mobile Studio のオンライン・ヘルプを参照してください。

アプリケーションのデプロイ

テストを終了したアプリケーションは、OracleAS Wireless の本番インスタンスにデプロイできます。「マイ Studio」ホーム・ページで、デプロイするアプリケーションを選択して**「デプロイ」**をクリックします。**「デプロイ」**ボタンが使用できない場合は、Mobile Studio 管理者に連絡してください。

> **関連項目：** 詳細は、Mobile Studio のオンライン・ヘルプを参照してください。

Mobile Studio のカスタマイズ

Mobile Studio を様々な方法でカスタマイズし、顧客の既存の Web サイトとの強力な統合を実現できます。

サンプル・サービスの作成

Mobile Studio には 4 種類のサンプル・サービスが付属しています。Mobile Studio の登録済ユーザーは、これらのサービスにアクセスしてそれぞれのソース・コードを表示できます。これらのサービスは、OracleAS Wireless の様々な機能を使用してモバイル・アプリケーションを開発する方法を示すために作成されました。既存のサンプル・サービスを削除または編集するには、『Oracle Application Server Wireless 管理者ガイド』を参照してください。次の項では、新しいサンプル・サービスの作成方法について詳細を説明します。

サンプル・サービスを作成するには、XHTML または OracleAS Wireless XML を使用して最初にアプリケーションを開発する必要があります。サンプル・サービスは、必ず Mobile Studio からアクセスできる場所にホスティングします。また、アプリケーションのソース・コードが含まれている文書をホスティングすることも必要です。ソース・コードの表示を選択した場合（つまり、この文書のフォーマットに HTML 構文を使用できる場合）は、このファイルのコンテンツが取得され、HTML 文書の本体タグの内側に埋め込まれます。

> **注意：** 作成したサンプル・サービスを登録して Mobile Studio に表示するには、Mobile Studio 管理者ツールを使用する必要があります。詳細は、『Oracle Application Server Wireless 管理者ガイド』を参照してください。

特性の設定

Mobile Studio における特性の設定とは、サイトの特定の外観と操作方法を指します。たとえば、使用するイメージ（ロゴ、枠、アイコン）や、サイトの外観と操作方法を構成するページのテキスト内容（フォント・サイズ、色）です。この項では、新しい特性を設定して外観と操作方法をカスタマイズする方法について詳細を説明します。高度なカスタマイズ（ページのレイアウトやフローの変更など）が必要な場合は、6-9 ページの「JSP ページ」および付録 G「JSP タグ・ライブラリ」を参照してください。

特性設定の新規作成では Java によるコーディングは必要ありません。特性設定の定義には、宣言的なアプローチが使用されます。特性設定の新規作成に必要なのは、HTML に関する知識のみです。

特性設定を新規作成する手順は、次のとおりです。

1. OracleAS Wireless インストールのルート・ディレクトリにナビゲートし、J2EE アプリケーションがデプロイされているディレクトリ（$IASW_ROOT/wireless/j2ee/applications など）を検索します。特性設定の定義は、studio/studio-web/sites サブディレクトリ（以降の説明では特性設定ディレクトリと呼びます）に格納されています。初期状態のインストールでは、この特性設定ディレクトリに、default というフォルダがあり、このフォルダには、Mobile Studio に付属しているデフォルトの特性設 s 定が格納されています。
2. 特性設定ディレクトリに新しいフォルダを作成することによって、新しい特性設定を作成します。

注意： フォルダの名前が、使用する特性設定の名前です。

各特性設定には、site.properties と呼ばれるファイルが必要です。このファイルには、使用するテキストとイメージのリソースに対する宣言が含まれています。たとえば、common.css.filename は、このファイル内にあるキーで、Mobile Studio で使用するカスケード・スタイルシート（CSS）ファイルを制御します。Mobile Studio に付属している特性設定には、サポートしている各ロケールのプロパティ・ファイルが含まれています。たとえば、プロパティ・ファイル site_fr.properties は、フランス語（fr）ロケールに使用されます。

ヒント： 特性設定を新規作成する簡単な方法は、Mobile Studio に付属しているデフォルトの特性設定をコピーし、必要に応じて編集することです。Mobile Studio に付属しているデフォルトの特性設定を直接編集して、外観と操作方法をカスタマイズすることもできます。

---

site.properties ファイルに宣言できる有効なキーは、マスター・ファイル SiteResources.properties によって制御されます。追加のキー（通常は高度なカスタマイズ用）が必要な場合は、このマスター・ファイルを $IASW_ROOT/wireless/server/classes/messages/oracle/panama/studio に配置できます。このファイルへの変更を有効にするには、OracleAS Wireless サーバーの再起動が必要です。

注意： Mobile Studio に表示する特性設定を宣言するには、Mobile Studio 管理者ツールを使用する必要があります（Mobile Studio は、デフォルトでは Mobile Studio に含まれている特性設定を表示します）。詳細は、『Oracle Application Server Wireless 管理者ガイド』を参照してください。

複数ロケールのサポート

Mobile Studio に付属しているデフォルトの特性設定には、28 のロケールに対するバンドルされたサポートが用意されています。ただし、初期状態のインストールで使用できるのは、英語ロケール（en）のみです。他のロケールに対するサポートを使用可能にするには、『Oracle Application Server Wireless 管理者ガイド』を参照してください。

作成した特定設定でロケールをサポートする場合、またはデフォルトの特性設定で追加のロケールをサポートする場合は、関連する変換を含めたリソース・ファイルを作成する必要があります（特性設定ディレクトリの詳細は、6-7 ページの「特性の設定」を参照してください）。

たとえば、ヒンディー語（hi）のロケールを作成する手順は、次のとおりです。

1. リソース・バンドル・ファイルを特性設定ディレクトリに作成します。使用する Mobile Studio の特性設定ディレクトリにナビゲートします。最も簡単な方法は、サポートされているロケールに対する既存のリソース・バンドルをコピーして適切なキーを変換し、site_hi.properties という新しいファイルとして保存する方法です。
2. サーバー側エラー・メッセージのためのリソース・バンドル・ファイルを作成します。$IASW_ROOT/wireless/server/classes/messages/oracle/panama/studio ディレクトリにナビゲートします。最も簡単な方法は、サポートされているロケールに対する既存のリソース・バンドルをコピーして適切なキーを変換し、messages_hi.properties という新しいファイルとして保存する方法です。
3. 変換したクライアント側エラー・メッセージを作成します。$IASW_ROOT/wireless/j2ee/applications/studio/studio-web/Javascript ディレクトリにナビゲートします。最も簡単な方法は、このフォルダ内の既存のファイルをコピーして適切なキーを翻訳し、ommsg_hi.js という新しいファイルとして保存する方法です。

注意： サポートするロケールを使用可能にするには、Mobile Studio 管理者ツールを使用する必要があります。詳細は、『Oracle Application Server Wireless 管理者ガイド』を参照してください。

Mobile Studio では、リクエストによる優先ロケールのリストを検査して、表示するロケールを判断します。アルゴリズムは、Java でリソース・バンドルをロードする方法に類似しています。初期状態のインストールでのデフォルトのロケールは、英語（en）です。

JSP ページ

この項では、Mobile Studio で使用される主な JSP のカスタマイズの詳細について説明します。

- JSP ページ : login.jsp
- JSP ページ : registraton.jsp
- JSP ページ : loginPortlet.jsp
- JSP ページ : pageHeader.jsp
- JSP ページ : pageFooter.jsp
- JSP ページ : pageMenu.jsp
- JSP ページ : pagePortlets.jsp
- JSP ページ : profile.jsp
- JSP ページ : home.jsp
- Java Beans
- JSP ページ : testAppInfoBox.jsp

JSP ページ : login.jsp

ログインせずに他のページにアクセスしようとしたユーザーは、適切なエラー・メッセージとともに、このページにリダイレクトされます。

> **注意 :** 統合モードの個々のページでは、ユーザーはそれぞれの OID ユーザー名とパスワードを使用してログイン（または自己登録）します。login.jsp が使用されるのは、Mobile Studio がスタンドアロン・モードで実行されている場合のみです。

ユーザーは、このページで次のアクションを実行します。

- Mobile Studio へのログイン
- 新規ユーザーとしての登録
- メニューを使用した情報ページの参照

図 6-4 Mobile Studio「ログイン」ページ（一部のみ）

表 6-1 Mobile Studio「ログイン」ページのリソース

名前	説明	例
login.text.info	情報テキスト	「OracleAS Wireless Studio は、Wireless アプリケーションを迅速に作成、テストおよびデプロイするためのオンライン環境です。」
login.text.title	ページ・タイトル	「OracleAS Wireless Studio へようこそ」
login.image.frontpage	ログイン・ページ上の 340 × 340 のスプラッシュ・イメージ	/images/frontpage.gif
common.label.register	「登録」ボタンのラベル	「登録」

JSP ページ : registraton.jsp

ユーザーは「登録」フォームに入力して Mobile Studio に登録できます。

> **注意 :** 統合モードの個々のページでは、ユーザーはそれぞれの OID ユーザー名とパスワードを使用してログイン（または自己登録）します。registraton.jsp が使用されるのは、Mobile Studio がスタンドアロン・モードで実行されている場合のみです。

図 6-5「登録」ページ（一部のみ）

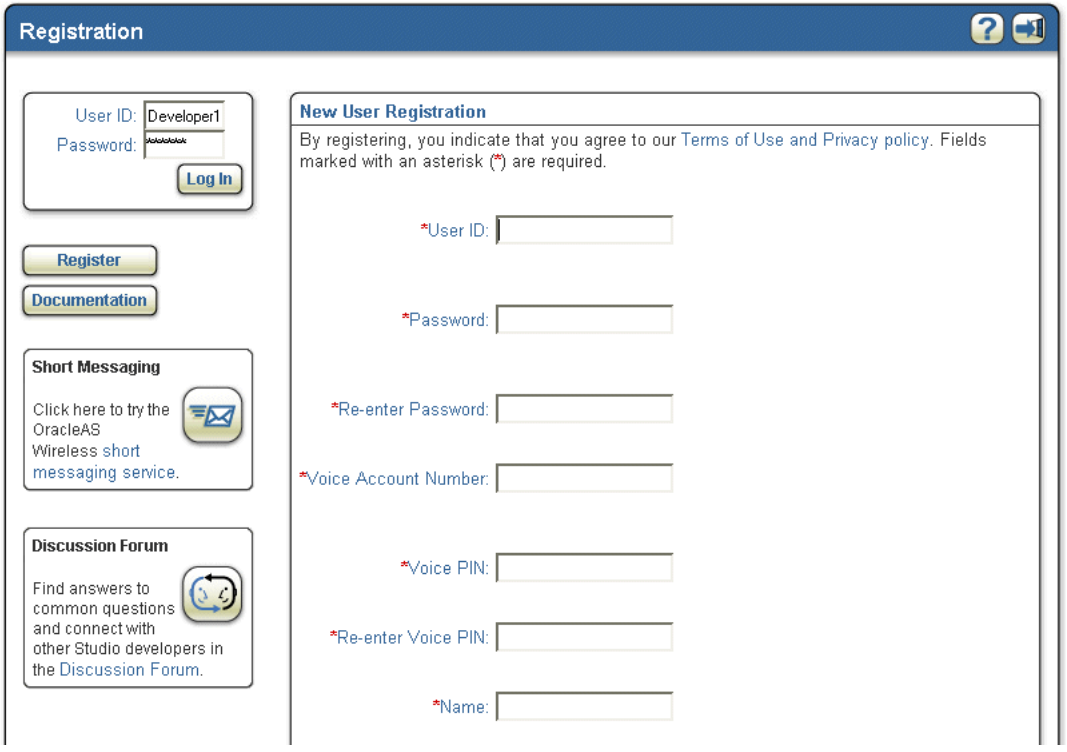


表 6-2「登録」ページのリソース

名前	説明	例
common.href.register	「登録」ページの URL	Registration.jsp
common.label.register	「登録」ボタンのラベル	「登録」
register.text.title	ページ本体のタイトル・テキスト	「新しいユーザーの登録」
register.text.info	登録に関する情報メッセージ	「登録することにより、あなたが使用条項およびプライバシ・ポリシーに同意することが示されます。アスタリスク（*）で示されるフィールドは、必須項目です。」
common.label.userid	「ユーザー ID」フィールドのラベル	「ユーザー ID」
register.hint.userid	「ユーザー ID」フィールドのヒント	「英数字によるユーザー ID を選択します。」
common.label.password	「パスワード」フィールドのラベル	「パスワード」
register.hint.password	「パスワード」フィールドのヒント	「英数字によるパスワードを選択します。」
common.label.password2	「パスワードの再入力」フィールドのラベル	「パスワードの再入力」
register.hint.password2	「パスワードの再入力」フィールドのヒント	「検証のためにパスワードを再入力してください。」
register.label.accountnumber	「アカウント番号」フィールドのラベル	「ボイス・アカウント番号」
register.hint.accountnumber	「アカウント番号」のヒント	「ボイス・ログインのために必要とされるアカウント番号を選択してください。」
register.label.voicepin	「ボイス PIN」のラベル	「ボイス PIN」
register.hint.voicepin	「ボイス PIN」のヒント	「ボイス・ログインのために数値の PIN を選択してください。」
register.label.voicepinagain	「ボイス PIN の再入力」のラベル	「ボイス PIN の再入力」
register.hint.voicepin2	「ボイス PIN の再入力」のヒント	「検証のために PIN を再入力してください。」
common.label.name	「名前」フィールドのラベル	「名前」
register.hint.name	「名前」フィールドのヒント	「姓名を入力してください。例 : 山田太郎。」
common.label.email	「電子メール」フィールドのラベル	「電子メール・アドレス」
register.hint.email	「電子メール」フィールドのヒント	「あなたの電子メール・アドレスを入力してください。例 : tyamada@company.com」
common.label.phone	「電話」フィールドのラベル	「電話番号」

表 6-2「登録」ページのリソース（続き）

名前	説明	例
register.hint.phone	「電話」フィールドのヒント	「国および地域コードを含んでいる電話番号を入力してください。例 : 16505555000」
common.label.workaddr	「勤務先住所」のラベル	「勤務先住所」
common.label.company	「会社」のラベル	「会社名」
common.label.addr	「住所行 1」のラベル	「住所行 1」
common.label.addr2	「住所行 2」のラベル	「住所行 2」
common.label.city	「市区町村」のラベル	「市区町村」
common.label.state	「都道府県」のラベル	「都道府県」
common.label.zip	「郵便番号」のラベル	「郵便番号」
common.label.country	「国」のラベル	「国」
register.label.setdefault	「デフォルトとして設定」のラベル	「デフォルトとして設定」
register.hint.workaddr	「勤務先住所」のヒント	「職場の住所を入力してください。」
common.label.homeaddr	「自宅の住所」のラベル	「自宅の住所」
register.hint.homeaddr	「自宅の住所」のヒント	「自宅の住所を入力してください。」
common.label.register	「登録」ボタンのラベル	「登録」
common.label.cancel	「取消」ボタンのラベル	「取消」

JSP ページ : loginPortlet.jsp

このページには、ユーザーがログインするためのフォームが組み込まれています。

注意 : 統合モードの個々のページでは、ユーザーはそれぞれの OID ユーザー名とパスワードを使用してログイン（または自己登録）します。loginPortlet.jsp が使用されるのは、Mobile Studio がスタンドアロン・モードで実行されている場合のみです。

表 6-3「ログイン」ポートレットのリソース

名前	説明	例
common.label.login	「ログイン」ボタンのラベル	「ログイン」
common.label.password	パスワード・テキスト入力フィールドのラベル	「パスワード」
common.label.userid	ユーザー ID テキスト入力フィールドのラベル	「ユーザー ID」
forgot.password.label	パスワードを忘れた場合のヒントを入力するフィールドのラベル	「パスワードを忘れた場合」

JSP ページ : pageHeader.jsp

このページは、JSP を表示しているすべてのカスタマに対するヘッダーとして組み込まれています。

表 6-4 ページ・ヘッダーのリソース

名前	説明	例
page.title	ウィンドウ・タイトル	「ホーム」
site.name	サイト（特性設定）名	
common.css.filename	カスケード・スタイルシート（CSS）ファイル	sites/default/om.css
global.head	HTML ヘッダーに含める追加カスタマイズ	sites/default/samplepagehead.html
page.body	本体タグのカスタマイズ	
global.header	ヘッダー・リージョンへの追加カスタマイズ	sites/default/samplepageheader.html
common.image.button.help	「ヘルプ」ボタンのイメージ名	sites/default/images/button_help.gif
common.image.button.logout	「ログアウト」ボタンのイメージ名	sites/default/images/button_logout.gif
common.image.window.left	左ウィンドウ枠のイメージ名	sites/default/images/window_left.gif
common.image.window.top	上ウィンドウ枠のイメージ名	sites/default /images/window_top.gif
common.image.window.topleft	ウィンドウ左上隅のイメージ名	sites/default/images/window_topleft.gif
common.image.window.topright	ウィンドウ右上隅のイメージ名	sites/default/images/window_topright.gif
common.href.help	「ヘルプ」ページの URL	

表 6-4 ページ・ヘッダーのリソース（続き）

名前	説明	例
common.href.logout	「ログアウト」アクションの URL	logout.jsp
common.tooltip.help	「ヘルプ」ボタンのツール・ヒント・テキスト	「ヘルプ」
common.tooltip.logout	「ログアウト」ボタンのツール・ヒント・テキスト	「ログアウト」

JSP ページ : pageFooter.jsp

このページは、JSP を表示しているすべてのカスタマに対するフッターとして組み込まれています。

表 6-5 ページ・フッターのリソース

名前	説明	例
global.footer	フッター・リージョンへの追加カスタマイズ	sites/default/samplepagefooter.html
common.image.window.bot	下ウィンドウ枠のイメージ名	sites/default/images/window_bot.gif
common.image.window.botleft	ウィンドウ左下隅のイメージ名	sites/default/images/window_botleft.gif
common.image.window.botright	ウィンドウ右下隅のイメージ名	sites/default/images/window_botright.gif
common.image.window.left	左ウィンドウ枠のイメージ名	sites/default /images/window_left.gif
common.image.window.line	ウィンドウ・セパレータの線のイメージ名	sites/default /images/window_line.gif
common.image.window.lineleft	ウィンドウ・セパレータの左結合部のイメージ名	sites/default /images/window_lineleft.gif
common.image.window.lineright	ウィンドウ・セパレータの右結合部のイメージ名	sites/default/images/window_lineright.gif
common.image.window.right	右ウィンドウ枠のイメージ名	sites/default /images/window_right.gif

JSP ページ : pageMenu.jsp

このページはほとんどのページに組み込まれており、ユーザーがブラウズするための共通のサイド・バーが用意されています。

表 6-6「ページ」メニューのリソース

名前	説明	例
common.label.home	「ホーム」ボタンのラベル	「マイ Studio」
common.label.profile	「プロファイル」ボタンのラベル	「マイ・プロファイル」
common.label.doc	「ドキュメント」ボタンのラベル	「ドキュメント」
common.label.logout	「ログアウト」ボタンのラベル	「ログアウト」
page.menu	メニュー・リージョンへの追加カスタマイズ	sites/default/samplepagemenu.html

JSP ページ : pagePortlets.jsp

このページはほとんどの JSP に組み込まれています。デフォルトの特性設定では、次のページ・ポートレットが表示されます。

- 「ショート・メッセージ」
- 「ディスカッション・フォーラム」

図 6-6 ページ・ポートレット

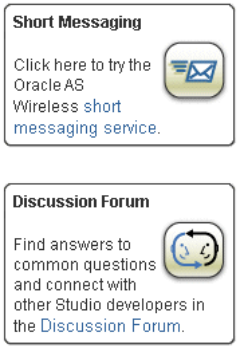


表 6-7 ページ・ポートレットのリソース

名前	説明	例
page.portlets	ページ・ポートレットに組み込まれている HTML	Sites/default/samplepageportlets.html

JSP ページ : profile.jsp

ユーザーは、このページで各自のプロファイルを表示して編集できます。

図 6-7 ページ・プロファイル

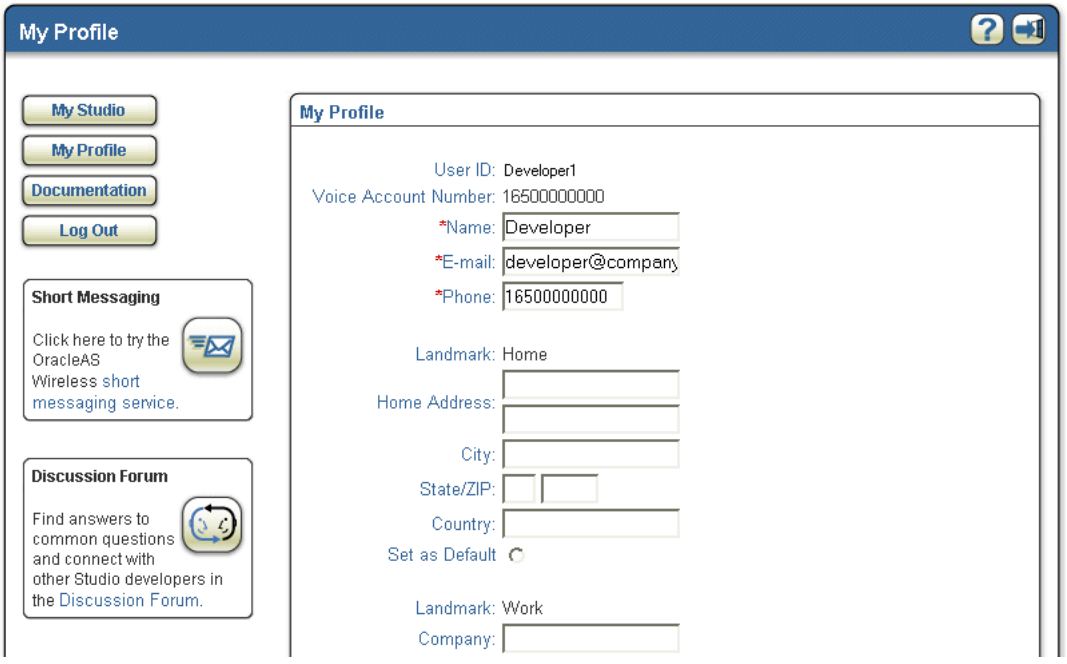


表 6-8 ページ・プロファイルのリソース

名前	説明	例
profile.body.title	ページ本体のタイトル・テキスト	「マイ・プロファイル」
common.labels.User.ID	「ユーザー ID」のラベル	「ユーザー ID」
register.label.accountnumber	「アカウント番号」のラベル	「アカウント番号」
common.labels.Name	「名前」フィールドのラベル	「名前」
common.labels.email.address	「電子メール」フィールドのラベル	「電子メール」
common.labels.Phone.Number	「電話」フィールドのラベル	「電話」
common.labels.Landmark	「ランドマーク」フィールドのラベル	「ランドマーク」
common.labels.HOME	「ホーム」フィールドのラベル	「ホーム」

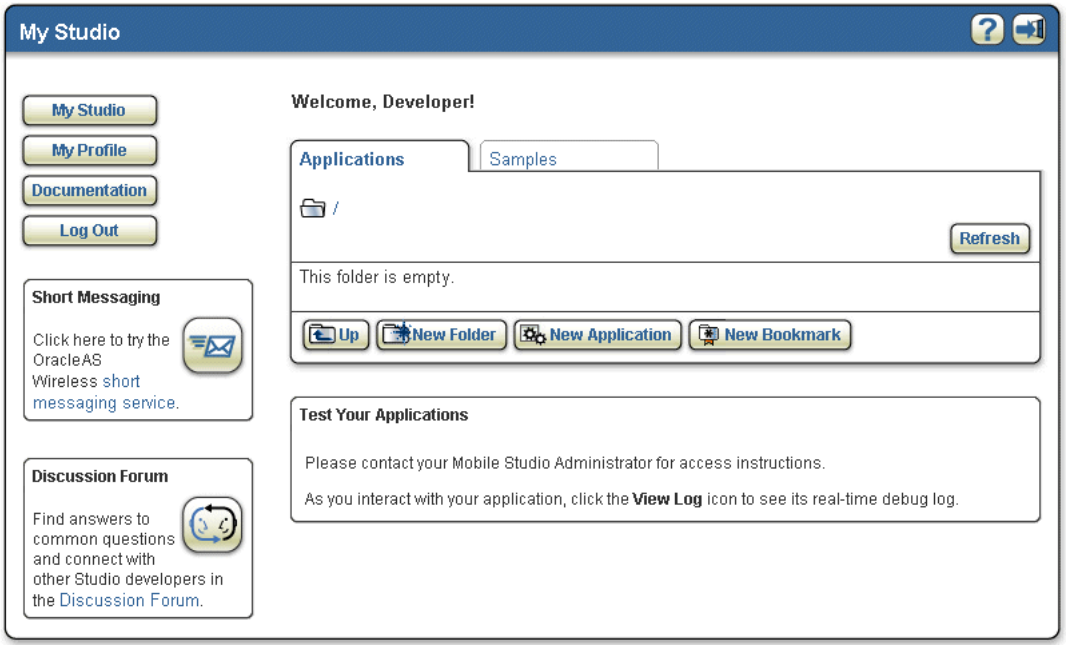
表 6-8 ページ・プロファイルのリソース（続き）

名前	説明	例
common.labels.home.address	「自宅の住所」フィールドのラベル	「自宅の住所」
common.labels.city	「市区町村」フィールドのラベル	「市区町村」
common.labels.stateZip	「都道府県 / 郵便番号」フィールドのラベル	「都道府県 / 郵便番号」
common.labels.country	「国」フィールドのラベル	「国」
common.labels.setDefault	「デフォルトを設定」フィールドのラベル	「デフォルトとして設定」
common.labels.WORK	「勤務先」フィールドのラベル	「勤務先」
common.labels.Company	「会社」フィールドのラベル	「会社」
common.labels.work.address	「勤務先住所」フィールドのラベル	「勤務先住所」
common.labels.changePin	「パスワードの変更」フィールドのラベル	「パスワードの変更」
common.labels.NewPassword	「新しいパスワード」フィールドのラベル	「新しいパスワード」
common.labels.NewPasswordAgain	「新しいパスワードの再入力」フィールドのラベル	「新しいパスワードの再入力」
profile.text.changepin	「PIN の変更」フィールドのラベル	「PIN の変更」
register.label.voicepin	「ボイス PIN」フィールドのラベル	「ボイス PIN」
register.label.voicepinagain	「ボイス PIN の再入力」フィールドのラベル	「ボイス PIN の再入力」
common.buttons.label.Save	「保存」ボタンのラベル	「保存」
common.buttons.label.Cancel	「取消」ボタンのラベル	「取消」

JSP ページ : home.jsp

このページは、ログインしたときに最初に表示されるユーザーのメイン・ページです。

図 6-8 ホーム・ページ



ユーザーは、このページで次のアクションを実行できます。

- アプリケーションの管理（作成、編集、削除、名前の変更、デプロイ、コピー、移動）
- デバイス・ポータル・フォルダの管理（作成、名前の変更、削除、コピー、移動）
- ブックマークの管理（作成、編集、削除、名前の変更、コピー、移動）
- フォルダとその内容の表示
- サイトの他の各部（サンプル・アプリケーション、デプロイの内容および登録された Web サービス）への移動

表 6-9 ホーム・ページのリソース

名前	説明	例
home.text.greeting	ユーザーへの初期画面メッセージ	「ようこそ」
common.href.home	ホーム・ページの URL	home.jsp
home.image.currfold	現行フォルダ用のイメージ（オープン状態のフォルダのイメージ）	sites/default/images/folderopen.gif
home.tooltip.upfolder	上位レベルのフォルダに関してユーザーに表示される情報メッセージ	「1 つ上のフォルダへ移動」
common.href.createfolder	新規フォルダ作成ページの URL	newFolder.jsp
home.tooltip.newfolder	新規フォルダの作成に関してユーザーに表示されるメッセージ	「新規フォルダの作成」
home.label.newfolder	「新規フォルダ」ボタンに表示されるラベル	「新規フォルダ」
home.tooltip.newapp	新規アプリケーションに関してユーザーに表示される情報メッセージ	「新しいアプリケーションの作成」
home.label.newapp	「新しいアプリケーション」ボタンに表示されるラベル	「新しいアプリケーション」
home.label.appname	アプリケーション名。	「名前」
home.label.appdesc	アプリケーションの説明	「説明」
home.label.appstatus	アプリケーションのステータス。	「ステータス」
home.image.application	アプリケーションのアイコン	sites/default /images/app.gif
home.label.view	「表示」ボタンのラベル	「表示」
home.label.rename	「名前の変更」ボタンのラベル	「名前の変更」
home.label.move	「移動」ボタンのラベル	「移動」
home.label.copy	「コピー」ボタンのラベル	「コピー」
home.label.delete	「削除」ボタンのラベル	「削除」
home.label.deploy	「デプロイ」ボタンのラベル	「デプロイ」
home.label.viewlog	「ログの表示」ボタンのラベル	「ログの表示」

Java Beans

表 6-10 Java Beans

名前	タイプ	説明
foldersList	Java.util.ArrayList	現行フォルダにあるユーザーのすべてのサブフォルダのリスト
servicesList	Java.util.ArrayList	特定のフォルダにあるユーザーのすべてのサービスのリスト

JSP ページ : testAppInfoBox.jsp

このページは、ユーザー・アプリケーションのテストに関する指示が必要な場所に組み込まれます。home.jsp ページと samples.jsp ページに格納されています。

表 6-11 アプリケーション・テスト情報ボックスのリソース

名前	説明	例
home.text.test.boxtitle	テスト指示ボックスのタイトル	「テスト・アプリケーション」
home.text.test.begin	指示テキスト	
home.text.test.accessinfo	指示テキスト	「ボイスを使用して接続します : 1-800-000-000」
home.text.test.end	指示テキスト	「**「ログの表示」**ボタンをクリックしてリアルタイム・デバッグ情報を表示します」

7

Wireless Customization Portal

この章では、Wireless Customization Portal を使用して、ブラウザからポータルをカスタマイズする方法について説明します。項ごとに様々なトピックを記載しています。各項の内容は、次のとおりです。

- OracleAS Wireless Customization の概要
- Wireless Customization へのログイン
- ユーザー・プロファイルの管理
- アプリケーションのカスタマイズ
- デバイスの管理
- ロケーション・マークの管理
- 連絡ルールの管理
- Netscape 4.7 以前を使用してローカライズされた言語での UTF-8 のページの表示
- Customization Portal の特性変更

OracleAS Wireless Customization の概要

Oracle Application Server Wireless Customization を使用すると、Wireless ポータルをカスタマイズしたり、Wireless モバイル・サービスにアクセスできます。また、フォルダ、ブックマーク、短縮名、デバイス、ロケーション・マーク、連絡ルールなどのリポジトリ・オブジェクトの作成、ユーザー・ビューの表示と作成、通知対応アプリケーションのサブスクライブ、J2ME メディア・コンテンツのダウンロードも可能です。

Wireless Customization へのログイン

Wireless Customization を使用するには、最初に次の手順でログインする必要があります。

次の URL でログイン・ページにアクセスします。

http://hostname:port/mobile/login.uix

または

http://hostname:port/mobile/

たとえば、次のように入力します。

http://hostname:7777/mobile/login.uix

または

http://hostname:7777/mobile/

注意： 7777 は、OracleAS Wireless のデフォルトのポート番号です。ポート番号の範囲は 7777 ～ 7877 です。正しいポート番号を使用していることを確認するには、[Oracle home]/install/portlist.ini に保存されている OracleAS Wireless のポート番号をチェックします。ポートの使用方法の詳細は、Oracle のインストレーション・マニュアルおよび『Oracle Application Server Wireless 管理者ガイド』を参照してください。

URL を入力すると、OracleAS Wireless Customization のログイン・ページが表示されます。このページには「ログイン」ボタンと「ヘルプ」ボタンがあります。表 7-1 に、各ボタンの説明を示します。

表 7-1 ログイン画面のボタン

ボタン	説明
ログイン	このボタンをクリックし、正しいユーザー名とパスワードを入力してログインします。
ヘルプ	このページでは、このボタンは使用できません。

ユーザー名を入力し、次にパスワードを入力します。管理者の場合は、ユーザー名として「orcladmin」を入力して**「ログイン」**をクリックします（パスワードはインストール時に設定されますが、ユーザー・マネージャを使用して変更できます）。

誤ったユーザー名またはパスワードを入力した場合はログインできません。

正常にログインすると「ようこそ」ページが表示されます。このページには、ユーザーのアカウント番号や Oracle Application Server Wireless アプリケーションへのアクセスに使用するアドレスが表示されます。

表 7-2 に、Wireless Customization「ようこそ」ページの要素を示します。

表 7-2「ようこそ」ページの要素

要素	内容
ホーム	この画面には、音声、Web ブラウザおよび双方向メッセージに関するアクセス情報が表示されます。
ユーザー・プロファイル	ユーザーが基本的なユーザー情報を設定または編集する画面が表示されます。
アプリケーション	アプリケーション・ツリー画面が表示されます。フォルダ、ブックマーク、通知サブスクリプションおよび短縮名を管理できます。フォルダ内のアプリケーションの並び順を変更したり、メディア・アプリケーションのコンテンツをダウンロードすることもできます。
デバイス	「デバイス」画面が表示されます。この画面には、ユーザーに現在関連付けられているデバイスがリスト表示されます。このページでは、リストされているデバイスのデフォルト・ステータスを作成、削除、編集または変更できます。
ロケーション・マーク	「ロケーション・マーク」画面が表示されます。この画面には、ユーザーに現在属しているロケーションがリスト表示されます。この画面では、ロケーション・マークのデフォルト・ステータスを作成、削除、編集または変更できます。また、ロケーション認識ポリシーに認証を設定することによって、ロケーションのプライバシを設定することもできます。
連絡ルール	「連絡ルール」画面が表示されます。この画面には、現行の連絡ルールがリスト表示されます。このページを使用すると、現行の連絡ルールを作成、削除、編集または設定できます。

Wireless Customization にログインした後は、次のボタンにアクセスできます。表 7-3 に、各ボタンの説明を示します。

表 7-3 OracleAS Wireless Customization のボタン

ボタン	説明
ログアウト	このボタンをクリックして、Wireless Customization からログアウトします。
ヘルプ	このボタンをクリックすると、ヘルプ・トピックのリストが表示されます。

新規ユーザーとしての Wireless Customization へのアクセス

電話番号と PIN が未登録の場合は、モバイル・デバイスの登録ページへの移動を指示する OracleAS Wireless のプロンプトが表示されます。この登録ページで電話番号と PIN を指定し、ワイヤレス・アクセスと音声アクセスのアカウント番号を登録します。この登録を完了すると、詳細設定ページへの移動を指示する OracleAS Wireless のプロンプトが表示されます。このページには、「ホーム」、「ユーザー・プロファイル」、「アプリケーション」、「デバイス」、「ロケーション・マーク」および「連絡ルール」というサイド・ナビゲーション・タブがあります。

登録済ユーザーとしての Wireless Customization へのアクセス

アカウントが初期化されて登録済ユーザーとなると、OracleAS Wireless では、Wireless Customization への次回ログイン時には詳細設定ページに移動するように、プロンプトが表示されます。詳細ページは「ホーム」タブにデフォルト設定されています。

ユーザー・プロファイルの管理

OracleAS Wireless Customization では、ユーザー・プロファイルを編集できます。

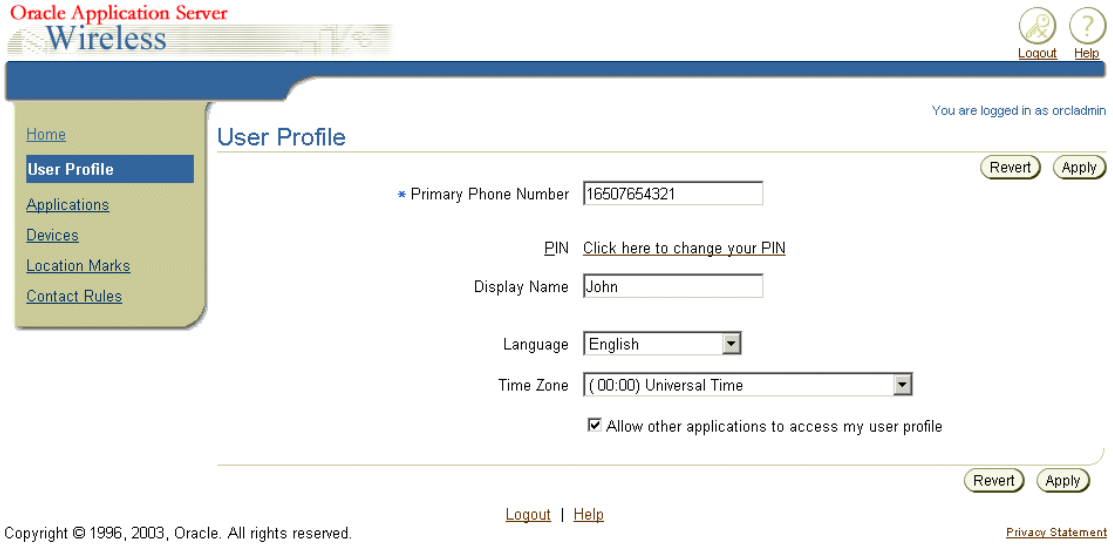
プロファイルを編集するには、最初に「ユーザー・プロファイル」タブを選択します。「ユーザー・プロファイル」画面（図 7-1）に、選択したユーザーの構成情報が表示されます。表 7-4 に、「ユーザー・プロファイル」画面の要素を示します。

表 7-4「ユーザー・プロファイル」画面の要素

パラメータ	説明
電話番号	このパラメータに登録した値は、音声アクセスのアカウント番号としても使用されます。
表示名	ユーザーの表示名。表示名を入力しないと、ユーザー名が表示名として使用されます。
パスワードを変更するには、ここをクリックしてください	パスワードと PIN を変更するページへのリンク。
言語	表示言語のドロップダウン・リスト。これは必須フィールドです。Netscape 4.7（またはそれ以前）を使用してローカライズされた言語で UTF-8 のページを表示する詳細は、7-47 ページの「Netscape 4.7 以前を使用してローカライズされた言語での UTF-8 のページの表示」を参照してください。
タイム・ゾーン	ユーザーのロケールに対するタイムゾーンのドロップダウン・リスト。通知は、Wireless サーバー自体のタイムゾーンではなく、ユーザーが選択したタイムゾーンに対して、生成および配信されます。これは必須フィールドです。
外部アプリケーションへの ID の開示	このチェックボックスによって、ユーザー ID をサード・パーティ・アプリケーションに開示できます。

必要に応じてユーザーの構成パラメータを変更し、**「適用」**をクリックします。

図 7-1「ユーザー・プロファイル」画面



アプリケーションのカスタマイズ

Wireless Customization Portal には、フォルダ、ブックマーク、非同期アプリケーション、J2ME アプリケーションおよび通知アプリケーションのアプリケーション・タイプが含まれています。

「アプリケーション」画面（図 7-2）のデフォルトの表示では、トップレベル・フォルダが表示されます。ここに表示されるアプリケーションはすべて、開発者（およびその開発者が所属するグループ）がアクセスできるアプリケーションです。表 7-5 に、「アプリケーション」画面の要素を示します。

表 7-5「アプリケーション」画面の要素

要素	説明
フォーカス	このオプションを選択すると、選択したフォルダをルート・フォルダとして表示できます。複数のレベルで様々なフォルダを保持している場合は、このオプションで表示を 1 つのフォルダに分離します。
名前	フォルダの名前。
タイプ	オブジェクトのタイプ。フォルダに加え、ブックマーク、非同期アプリケーション、J2ME アプリケーションおよび通知アプリケーション対応のサービスを含めた Wireless Customization のアプリケーションのタイプがあります。アイコンのリストが表示され、異なるタイプのアプリケーションが識別されます。
参照可能	このチェックボックスを選択して、選択した表示プロファイルに対してフォルダを参照可能にします。
アクション	アクション・リンクのリスト。アプリケーションは、アプリケーション・タイプごとに固有で、「並び替え」、「削除」、「編集」、「サブスクライブ」、「カスタマイズ」および「ダウンロード」が含まれています。

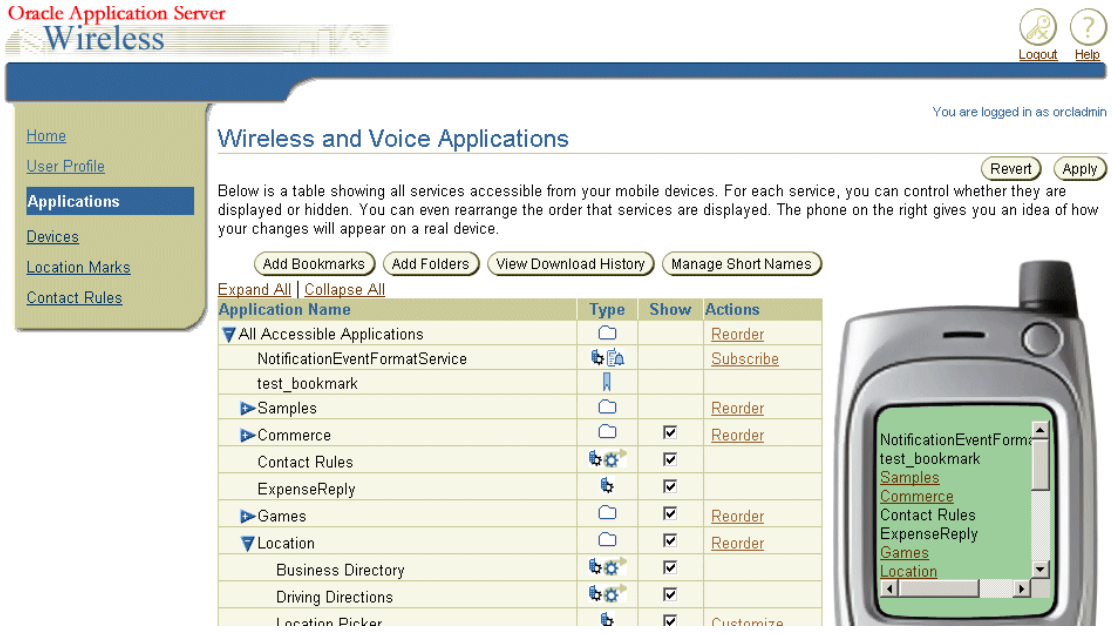
デフォルトでは、トップレベル・フォルダは閉じています。フォルダの内容を表示するには、**「すべて拡張」**をクリックします。フォルダをトップレベル表示に縮小するには、**「すべて縮小」**をクリックします。

Wireless Customization では、管理者はすべてのアプリケーションを変更できます。エンド・ユーザーなど、管理者ロールを付与されていないユーザーが変更できるのは、各自のアプリケーション（つまり、ユーザーが属しているユーザー・グループのアプリケーション）のみです。コンテンツ・マネージャは、アプリケーションの公開時に、アプリケーションをユーザー・グループに割り当てます。コンテンツ・マネージャを使用してアプリケーションをユーザー・グループに公開する方法は、『Oracle Application Server Wireless 管理者ガイド』を参照してください。

デフォルトでは、Wireless には「管理者」、「ゲスト」および「ユーザー」のユーザー・グループがあります。これらのユーザー・グループを構成するには、`ProvisioningHook` を使用します。新規ユーザーは、「ユーザー」グループに自動的に割り当てられ、コンテンツ・

マネージャが「ユーザー」グループに割り当てたアプリケーションすべてに対するデフォルトの権限を保持します。「アプリケーション」ページでは、ユーザーに対してグループ・メンバーシップを表示しません。また、ユーザーのグループに属している他のユーザーも表示しません。

図 7-2「アプリケーション」画面（一部のみ）



注意： デバイス・シミュレータには、アプリケーションが実際のモバイル・デバイスに表示されるとおりに表示されます。フォルダは作業リンクとしてレンダリングされるため、そのフォルダの子ノードの次のページにナビゲートできます。

フォルダの管理

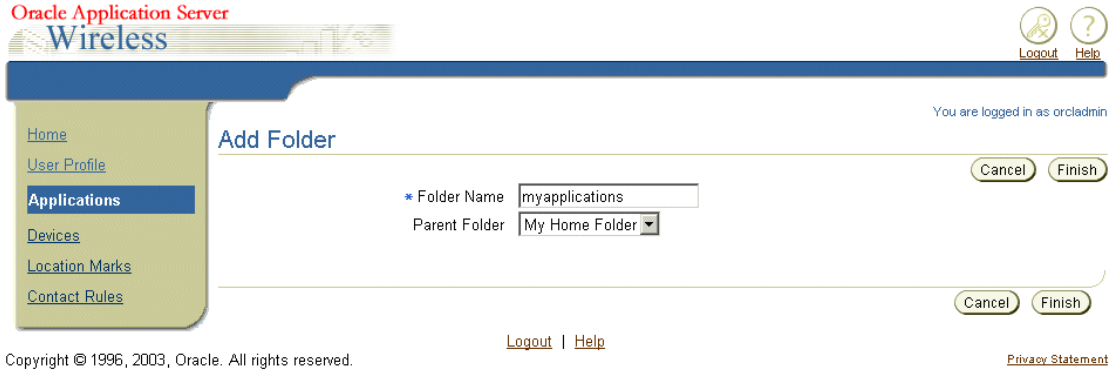
Wireless Customization では、ユーザーが所有するサブフォルダを作成、編集および削除できます。

サブフォルダの作成

サブフォルダを作成することによって、「アプリケーション」ページのフォルダを編成できます。

サブフォルダを作成するには、最初に、アプリケーション・ツリー表の上部にある**「フォルダの作成」**ボタンをクリックします。「フォルダの作成」画面（図 7-3）が表示されます。

図 7-3「フォルダの作成」画面



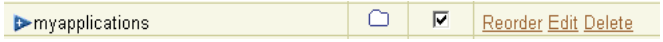
この画面で、サブフォルダの名前（これは必須フィールドです）を入力し、次に「親フォルダ」フィールドのドロップダウン・リストからフォルダの位置（この位置は、ユーザーのホーム・フォルダか、またはユーザーのホーム・フォルダのサブフォルダです）を選択します。**「終了」**をクリックします。「アプリケーション」ページが再表示され、現行のフォルダの適切な位置にフォルダが表示されます。**「取消」**をクリックすると、入力したすべての値が消去され、「アプリケーション」ページに戻ります。

フォルダの編集

「フォルダの編集」画面（図 7-5）では、フォルダの名前を変更できます。

フォルダを編集するには、最初に「アプリケーション」画面でフォルダを選択し、次に**「編集」**（図 7-4 のように、フォルダの横に「並び替え」と「削除」アクションとともにハイパーリンクで表示されます）をクリックします。

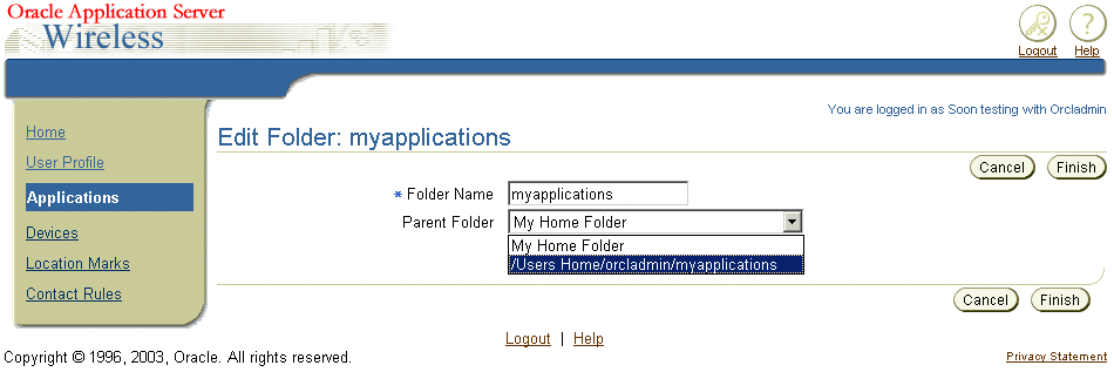
図 7-4 フォルダの編集



「編集」ページに、選択したフォルダに設定した値が表示されます。

必要な値（フォルダ・パラメータの詳細は、7-9 ページの「サブフォルダの作成」を参照してください）を変更し、**「終了」**をクリックします。「アプリケーション」ページが再表示され、現行のフォルダの適切な位置にフォルダが表示されます。**「取消」**をクリックすると、フォルダが以前の値になり、「アプリケーション」ページに戻ります。

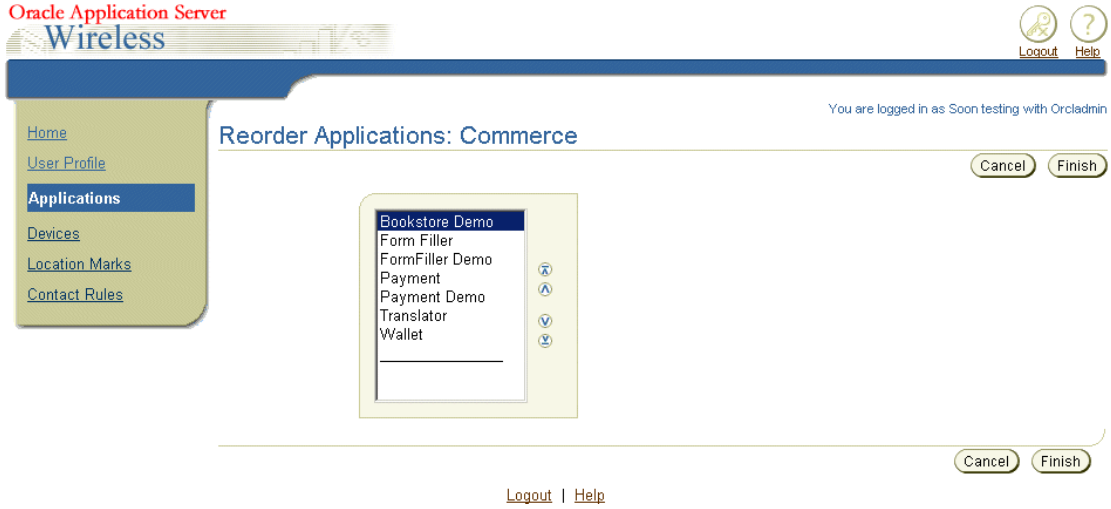
図 7-5「フォルダの編集」画面



フォルダの表示順序の並び替え

フォルダの表示順序を並び替えるには、「アプリケーション」画面でフォルダを選択し、次に**「並び替え」**オプション（図 7-6）をクリックします。「並び替え」画面が表示されます。

図 7-6「並び替え」画面

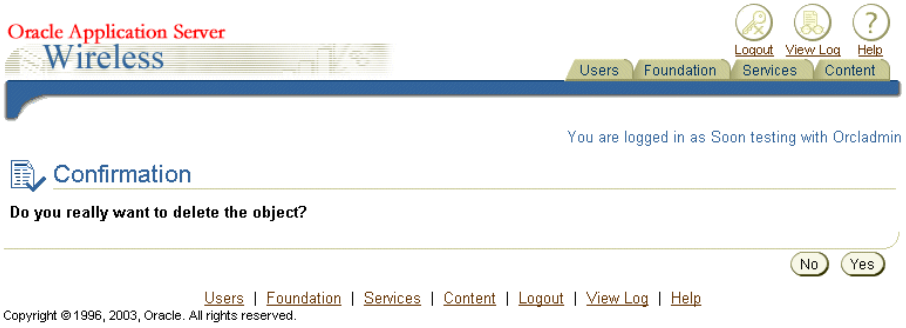


矢印を使用して、フォルダの位置を変更するか、または最上部または最下部に移動します。順序の設定を適用して「アプリケーション」ページに戻るには、**「終了」**をクリックします。**「取消」**をクリックすると、フォルダの配置が以前の状態に戻ります。

フォルダの削除

「アプリケーション」ページでフォルダを選択してから**「削除」**をクリックすることによって、フォルダを削除します。「確認」ページ（図 7-7）が表示されます。削除を承認するには**「はい」**を、取り消すには**「いいえ」**をクリックします。

図 7-7「確認」ページ



ブックマークの管理

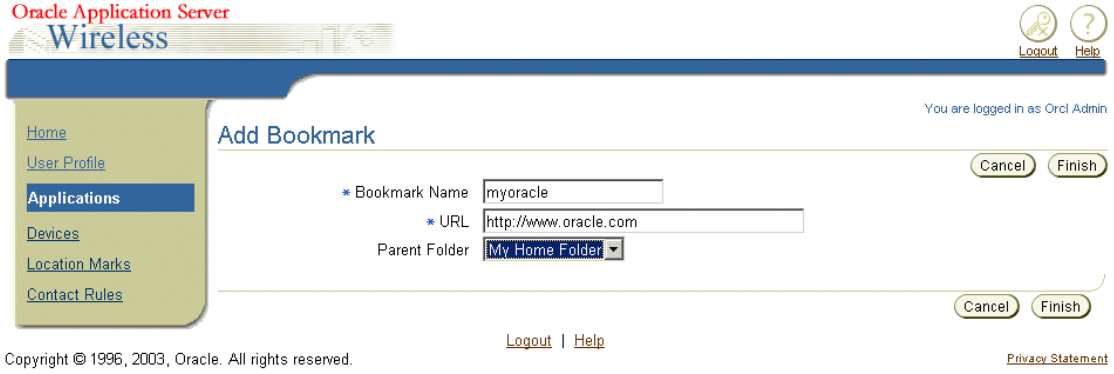
ブックマークは、サイトにすばやくアクセスできる外部 URL へのリンクです。

Wireless Customization では、ブックマークを作成、編集および削除できます。

ブックマークの作成

ブックマークを作成するには、最初に、「アプリケーション」画面で**「ブックマークの追加」**をクリックします。「ブックマークの追加」画面（図 7-8）が表示されます。

図 7-8「ブックマークの追加」画面



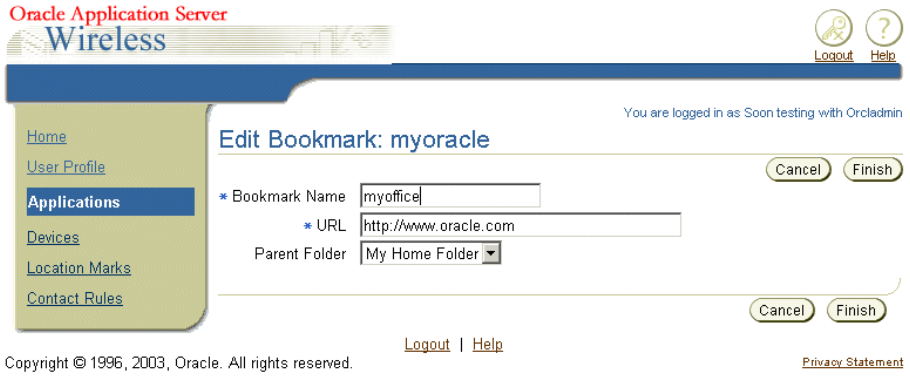
「ブックマークの追加」画面で、ブックマーク名（必須値）を入力し、次に新しいブックマークの URL（www.oracle.com など）を入力します。「親フォルダ」フィールドのドロップダウン・ボックスを使用して、新しいブックマークの位置を割り当てます。位置は、ユーザーのホーム・フォルダか、またはユーザーのホーム・フォルダのサブフォルダのいずれかです。**「終了」**をクリックします。「アプリケーション」画面の適切なフォルダの下に新しいブックマークが表示されます。**「取消」**をクリックすると、すべての値が消去され、「アプリケーション」画面に戻ります。

ブックマークの編集

ブックマークは、異なる URL を選択するか、名前を変更するか、または別のフォルダに配置することで変更できます。

ブックマークを編集するには、「アプリケーション」画面でブックマークを選択（またはフォルダを選択してブックマークにドリルダウン）し、次に**「編集」**をクリックします。フィールドに選択したブックマークの値が移入された状態の「ブックマークの編集」画面（図 7-9）が表示されます。

図 7-9「ブックマークの編集」画面



必要に応じてブックマークの値を変更します。ブックマークのパラメータに間する詳細は、7-13 ページの「ブックマークの作成」を参照してください。変更内容をコミットするには、**「終了」**をクリックします。**「取消」**をクリックすると、パラメータが以前の状態に戻ります。

ブックマークの削除

ブックマークを削除するには、「アプリケーション」画面でブックマークを選択（またはフォルダを選択してブックマークにドリルダウン）し、次に**「削除」**をクリックします。削除の確認ページが表示されます。削除を承認するには**「はい」**を、取り消すには**「いいえ」**をクリックします。

短縮名の管理

短縮名によって、ユーザーは非同期アプリケーションにアクセスするためのコマンドを指定できます。1 つ以上のコマンドとアプリケーションのシステム短縮名をまとめてグループ化し、1 つの短縮名で表すことができます。たとえば、システム短縮名「stk」の株価アプリケーションに対して、カスタマイズした短縮名「s」を割り当てることができます。短縮名には、システム短縮名と値リストを割り当てることもできます。たとえば、「stk」の後に「orcl」の値が続くシステム短縮名（stk orcl）に対して、短縮名「s」を割り当てることができます。また、システム短縮名とその短縮名に適した値のリストを結合した短縮名（stk orcl; weather sj,sf など）も作成できます。

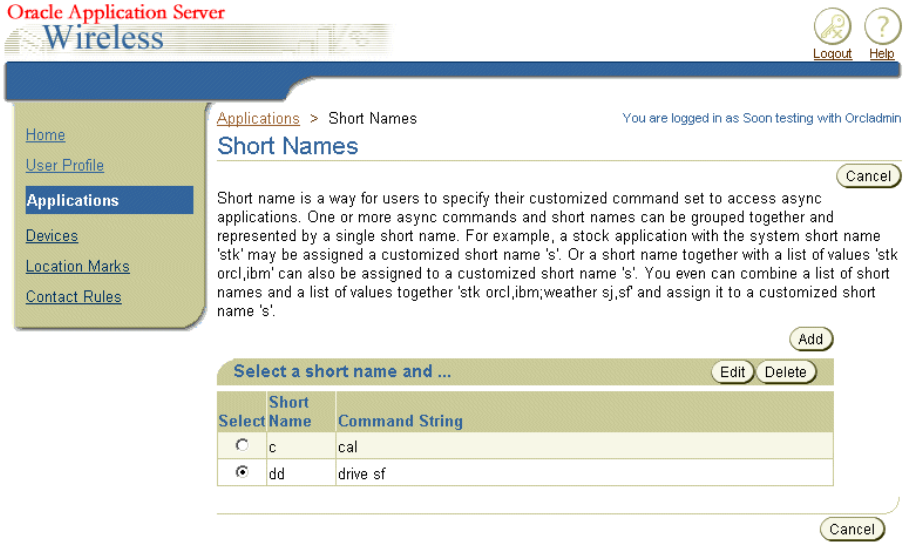
短縮名は、SMS、電子メール、Instant Messaging、双方向ポケットベルなど、双方向メッセージ・デバイスで使用されます。カスタマイズした短縮名をメッセージの件名または本体に入力することで、そのメッセージを双方向メッセージング・サーバーのアクセス・アドレス（通常は電子メール・アドレス）に送信できます。受信したサーバーは、短縮名コマンド文字列で発行されたリクエストに応答するメッセージを使用して返信します。

Wireless Customization では、**「短縮名の管理」**ボタンをクリックして、短縮名を作成、編集および削除できます。

短縮名の作成

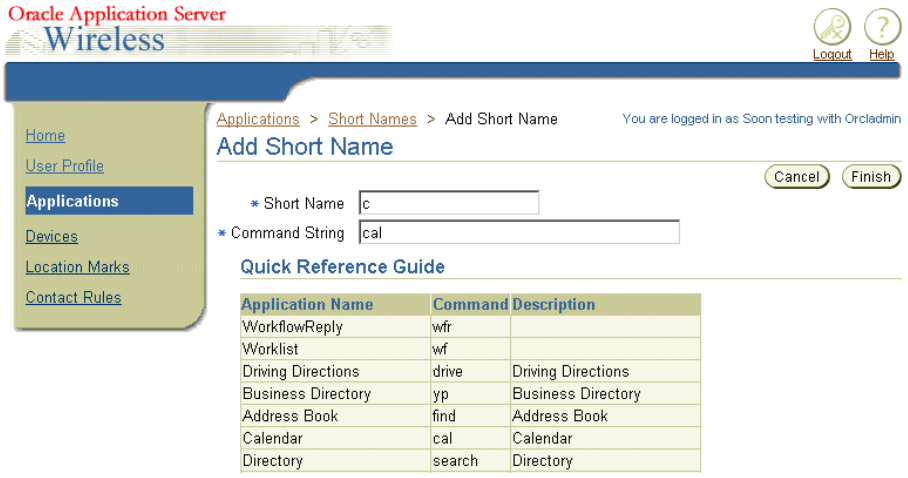
短縮名を作成するには、最初に、「アプリケーション」ページの**「短縮名の管理」**ボタンをクリックします。「短縮名」画面（図 7-10）に、短縮名がリスト表示されます。

図 7-10「短縮名」画面



「追加」をクリックすると、「短縮名の追加」画面（図 7-11）が表示されます。この画面には、アクセス可能なすべての非同期アプリケーションのクイック・リファレンス表が表示されます。

図 7-11「短縮名の追加」画面



この画面を使用して、短縮名を入力し、次にコマンド文字列を入力します。短縮名の追加を完了するには**「終了」**をクリックします。**「取消」**をクリックすると、この画面で入力したすべての値が消去されます。

短縮名の編集

短縮名は、名前を変更するか、またはその短縮名のコマンド文字列を変更することで編集できます。

短縮名を編集するには、最初に、「アプリケーション」画面の**「短縮名の管理」**ボタンをクリックします。表示された画面の短縮名から該当する短縮名を選択し、**「編集」**をクリックします。「短縮名の編集」画面に、選択した短縮名の値セットが表示されます。値を編集します。変更内容をコミットするには**「終了」**をクリックします。**「取消」**をクリックすると、短縮名の値が当初の状態に戻ります。

短縮名の削除

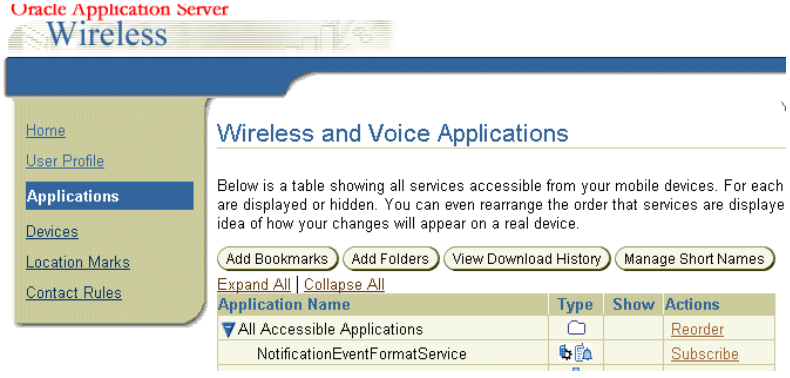
短縮名を削除するには、**「短縮名の管理」**ボタンをクリックします。表示された画面の表から該当する短縮名を選択し、**「削除」**をクリックします。「確認」画面が表示されます。削除を承認するには「はい」を、取り消すには「いいえ」を選択します。

通知サブスクリプションの管理

通知アプリケーションは、事前に定義した条件を使用して通知（アラート・メッセージ）を配信します。これらの条件は、値、時間またはロケーションに基づいて設定できます。たとえば、通知をトリガーする値の条件として、「オラクル社の株価が特定の値に達したときに株価情報を送信してください」という条件を設定できます。「平日の午後 3 時に株価指数を送信する」など、時間の条件を指定することもできます。値や時間の条件に加え、「配送トラックのドライバが顧客のサイトに到着したら通知してください」などのロケーションに基づく通知を定義できます。ロケーション・ベースの詳細は、第 11 章「通知エンジン」を参照してください。

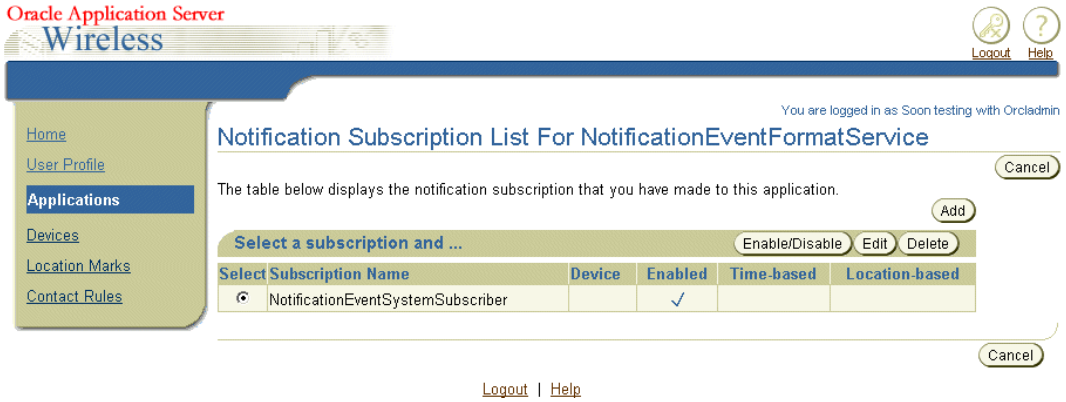
アプリケーション表では、「サブスクライブ」アクション・リンクが、通知に対応しているアプリケーションの「アクション」列に表示されます。図 7-12 では、このリンクが NotificationEventFormatService というアプリケーションに表示されています。

図 7-12 通知対応アプリケーション



「サブスクライブ」アクション・リンクをクリックすると、詳細ページ（図 7-13）が表示されます。このページには、このアプリケーションのサブスクリプションすべてをリストしたサブスクリプション表が含まれています。

図 7-13 通知アプリケーションの詳細ページ



このページでは、サブスクリプションを作成、削除、有効化または無効化できます。表には、「有効」、「時間ベース」、「ロケーション・ベース」など、サブスクリプションのステータスも表示されます。

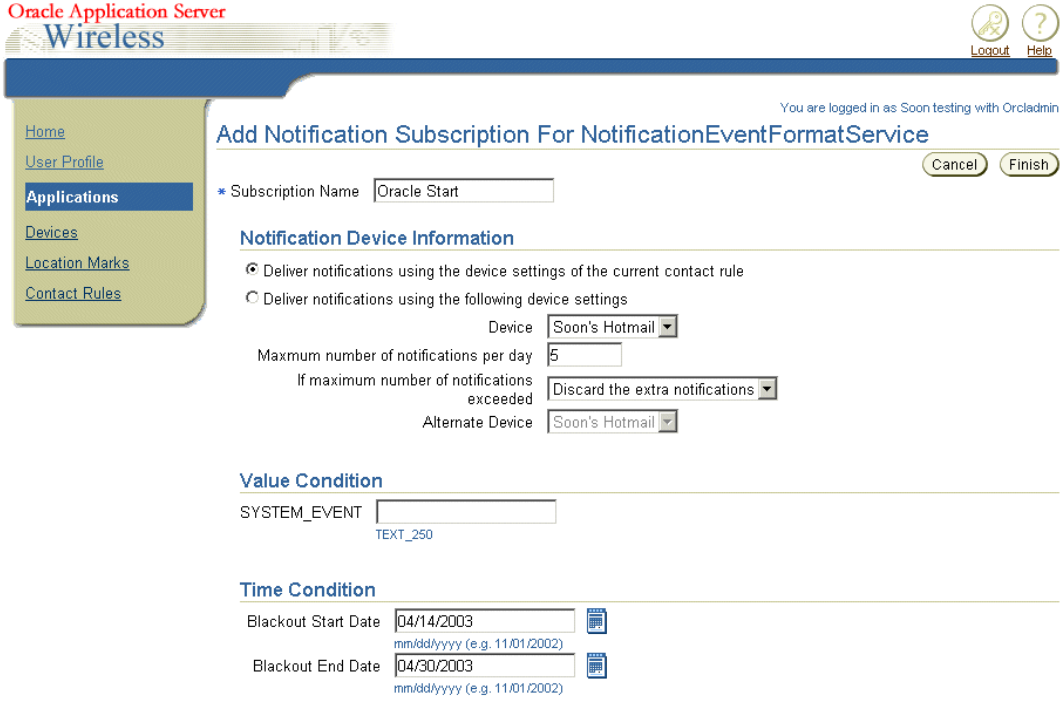
新しい通知サブスクリプションの追加

新しいサブスクリプションを追加する手順は、次のとおりです。

1. サブスクリプションを追加するには、アプリケーション表の**「サブスクライブ」**リンクをクリックします。詳細画面で**「追加」**をクリックします。選択した通知の「通知サブスクリプションの追加」画面（図 7-14）が表示されます。
2. サブスクリプション名を入力します。これは必須フィールドです。
3. メッセージ配信の通知デバイス作業環境を設定します。作業環境の設定には、次のようなオプションがあります。
   - 現在アクティブな連絡ルールの定義に基づいて、通知の配信を指定できます。
   - 主要な通知デバイスとそのデバイスで 1 日当たりに受信する最大通知数を指定できます。
   - 通知数が主要な通知デバイスに設定した最大数に達した場合に、追加のメッセージを処理する代替手段も指定できます。超過したメッセージは、すべて廃棄するか、または代替デバイスに転送できます。
4. 条件のすべてに値を入力します。通知アプリケーションが値ベースの場合は、「値の条件」セクションに値の条件リストが表示されます。

5. 通知アプリケーションが時間ベースの場合は、「時間の条件」セクションに時間の条件が表示されます。時間の条件には、ブラックアウト期間や、通知を一定の期間または特定の時間にトリガーするように設定するオプションが含まれます。トリガー条件設定の詳細は、第 11 章「通知エンジン」を参照してください。
6. 通知アプリケーションがロケーション・ベースの場合は、「ロケーション認識条件」セクションにロケーション認識条件が表示されます。監視するターゲットとそのターゲットの移動を指定できます。たとえば、「John がオラクル本社にいたら通知を送信してください」（この場合、オラクル本社はロケーション管理ツールで定義したリージョン・オブジェクト）などの条件を設定できます。
7. 入力した内容を保存するには、**「終了」**をクリックします。**「取消」**をクリックすると、入力した内容が消去されます。

図 7-14「通知サブスクリプションの追加」画面



通知サブスクリプションの編集

「通知サブスクリプション」画面では、選択したサブスクリプションのトリガー条件を変更するか、または通知配信ルールを変更することで、サブスクリプションを編集できます。

サブスクリプションを編集するには、アプリケーション表の**「サブスクライブ」**アクション・リンクをクリックして、詳細ページにアクセスします。サブスクリプション表から、変更するサブスクリプションを選択して、**「編集」**ボタンをクリックします。フィールドに選択した通知サブスクリプションの値が移入された状態の「編集」画面が表示されます（これらのフィールドの詳細は、7-18 ページの「新しい通知サブスクリプションの追加」を参照してください）。必要に応じて値を編集します。変更内容を保存するには、**「終了」**をクリックします。**「取消」**をクリックすると、パラメータが以前の値に戻ります。

通知サブスクリプションの削除

通知サブスクリプションを削除するには、詳細ページで該当する通知サブスクリプションを選択して、**「削除」**をクリックします。「確認」画面が表示されます。削除を承認するには**「はい」**を、取り消すには**「いいえ」**をクリックするように指示が表示されます。**「はい」**を選択します。

デバイスの管理

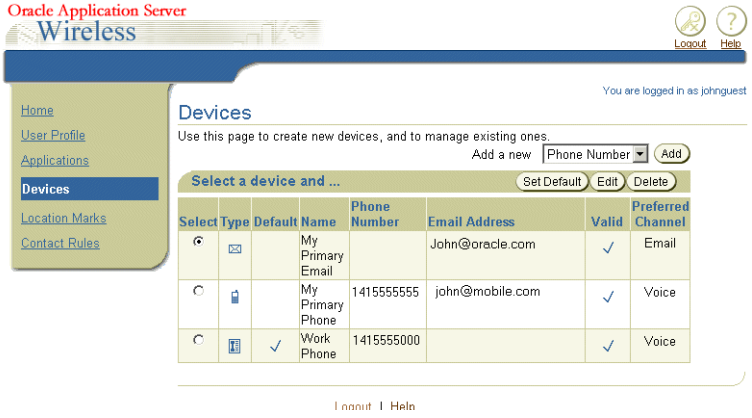
OracleAS Wireless では、デバイス・オブジェクトによって、複数のデバイスのアドレスを単一のエンティティにグループ化できます。たとえば、同一デバイスの様々なデバイス・アドレスをグループ化できます。このグループには、複数のユーザー・エージェントを指定したり、複数のプロトコルを使用できます。これらのプロトコル（またはチャネル）はそれぞれ異なるアドレスや ID を保持できます。ただし、そのすべては同一の物理エンティティから生じます。

デバイスは、通知サブスクリプションと連絡ルールの管理の両方に使用されます。通知を受信し、連絡ルールを設定する対象は、有効なデバイスのみです。

「デバイス」タブをクリックすると、「デバイス」画面（図 7-15）が表示されます。この画面では、デバイスを作成、編集および削除できます。デバイスは、電話、FAX、電子メールおよびモバイル・デバイスの 4 つのタイプに分類されます。電話は音声のみをサポートするデバイスです。FAX は FAX メッセージの受信のみを行うデバイスです。電子メールは主に電子メール・アカウント用で、モバイル・デバイスはマルチチャネル・モバイル・デバイス用のデバイスです。

デバイス表には、それぞれのデバイス・タイプごとに異なるデバイス・アイコンが表示されます。

図 7-15 デバイスの参照

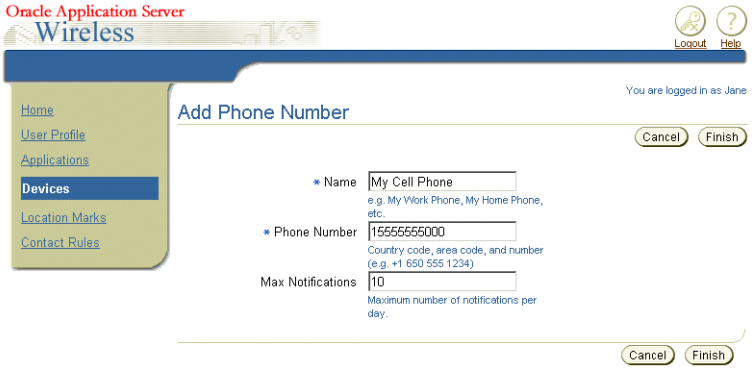


デバイス表の**「追加」**ボタンの横にあるドロップダウン選択リストでデバイスのタイプを選択することで、デバイスを作成できます。

新しい電話の作成

電話を作成するには、ドロップダウン・リストから「電話」を選択し、次に**「追加」**ボタンをクリックします。「電話番号の追加」画面（図 7-16）が表示されます。

図 7-16 電話番号の追加



電話の追加を完了するには、表 7-6 のパラメータを定義してから、**「終了」**をクリックします **「取消」**をクリックすると、入力した値が消去され、参照ページに戻ります。

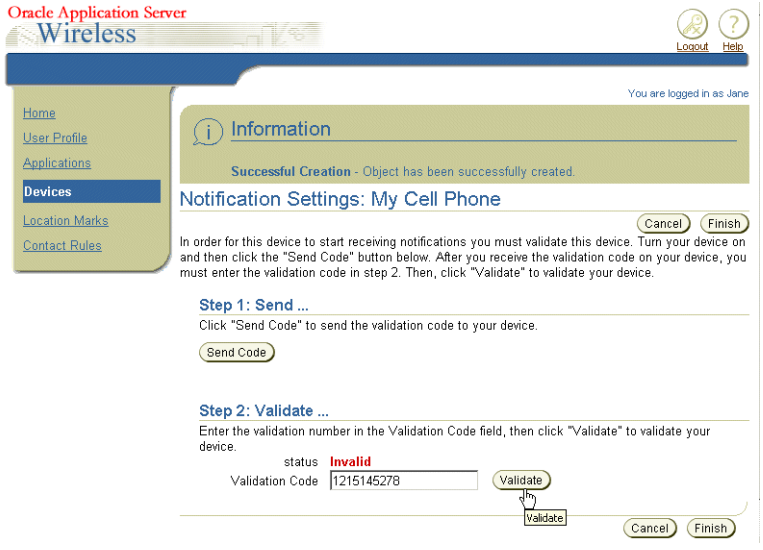
表 7-6「電話番号の追加」ページのパラメータ

パラメータ	値
名前	電話の名前を入力します。たとえば、「自宅の電話」のように入力します。
番号	当該電話の電話番号を入力します。たとえば、「1-555-555-5555」のように入力します。
最大通知数	この電話で 1 日当たりに受信する最大通知数を入力します。このデバイスで受信する通知数に制限を設定しない場合は、このフィールドに値を入力しません。

電話の検証

電話を作成すると、デバイスの検証（図 7-17）を指示する OracleAS Wireless のプロンプトが表示されます。

図 7-17 電話の検証



注意： 検証プロセスを開始する前に、電話の電源を入れてください。

電話を検証するには、**「送信」**をクリックして検証コードを電話に送信します。このコードを受信したら、画面にこのコードを入力して**「検証」**をクリックします。デバイスが正常に検証された場合は、「ステータス」列に「有効」が表示されます。

電話の編集

電話は、名前、デバイスの番号、または受信する通知数を変更することによって更新できます。電話を更新するには、該当する電話を選択して**「編集」**をクリックします。フィールドに選択したデバイスの設定値が移入された状態の「編集」画面が表示されます。これらの値の詳細は、7-22 ページの「新しい電話の作成」を参照してください。値を変更した後は、デバイスを再度検証する必要があります。デバイスの検証の詳細は、7-22 ページの「電話の検証」を参照してください。変更内容を保存するには、**「終了」**をクリックします。**「取消」**をクリックすると、電話のパラメータが以前の値に戻ります。

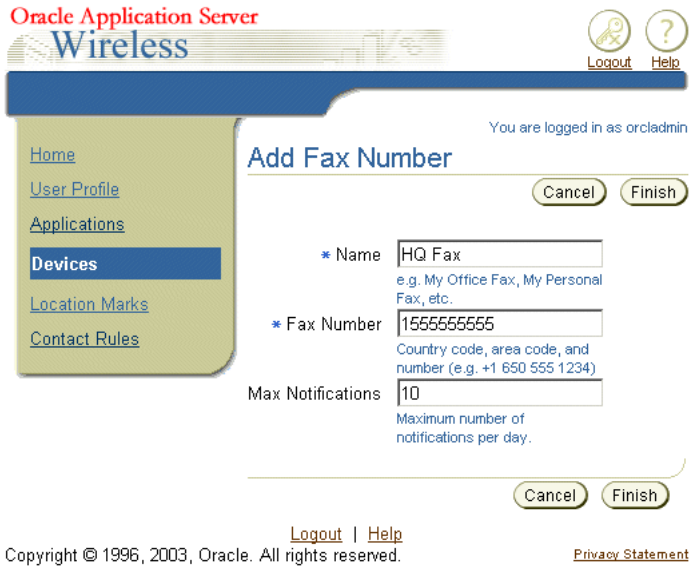
電話の削除

電話を削除するには、「デバイス・リスト」から該当する電話を選択し、**「削除」**をクリックします。「確認」画面が表示されます。削除を承認するには**「はい」**を、取り消すには**「いいえ」**を選択します。

新しい FAX の作成

FAX を作成するには、ドロップダウン・リストから「FAX」を選択し、次に**「追加」**ボタンをクリックします。「FAX 番号の追加」画面（図 7-18）が表示されます。

図 7-18 FAX 番号の追加



FAX の追加を完了するには、表 7-7 のパラメータを定義してから、**「終了」**をクリックします **「取消」**をクリックすると、入力した値が消去され、参照ページに戻ります。

表 7-7「FAX 番号の追加」画面のパラメータ

パラメータ	値
名前	FAX の名前を入力します。たとえば、「自宅の FAX」のように入力します。
番号	FAX の番号を入力します。たとえば、「1-555-555-5555」のように入力します。
最大通知数	この FAX で 1 日当たりに受信する最大通知数を入力します。このデバイスで受信する通知数に制限を設定しない場合は、このフィールドに値を入力しません。

FAX の検証

FAX を作成すると、デバイスの検証を指示する OracleAS Wireless のプロンプトが表示されます。

注意： 検証プロセスを開始する前に、FAX の電源を入れてください。

FAX を検証するには、**「送信」**をクリックして検証コードを FAX に送信します。このコードを受信したら、画面にこのコードを入力して**「検証」**をクリックします。デバイスが正常に検証された場合は、「ステータス」列に「有効」が表示されます。

FAX の編集

FAX は、名前、番号、または受信する通知数を変更することによって更新できます。FAX を更新するには、該当する FAX を選択して**「編集」**をクリックします。フィールドに選択したデバイスに固有の値が移入された状態の「編集」画面が表示されます。これらの値の詳細は、7-24 ページの「新しい FAX の作成」を参照してください。必要な変更を実行後、デバイスを検証します。デバイスの検証の詳細は、7-25 ページの「FAX の検証」を参照してください。変更内容を保存するには、**「終了」**をクリックします。**「取消」**をクリックすると、FAX のパラメータが以前の値に戻ります。

FAX の削除

FAX を削除するには、「デバイス・リスト」から該当する FAX を選択し、**「削除」**をクリックします。「確認」画面が表示されます。削除を承認するには**「はい」**を、取り消すには**「いいえ」**を選択します。

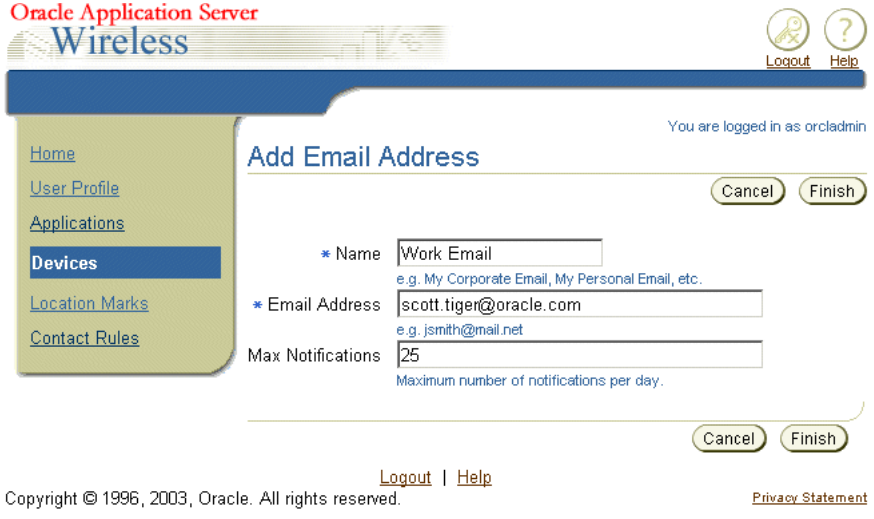
電子メール・デバイスの作成

電子メールを作成するには、ドロップダウン・リストから「電子メール」を選択し、次に**「追加」**をクリックします。「電子メール・アドレスの追加」画面（図 7-19）が表示されます。電子メール・デバイスの追加を完了するには、表 7-8 の値を定義する必要があります。

表 7-8「電子メール・アドレスの追加」画面のパラメータ

パラメータ	値
名前	電子メールの名前を入力します。たとえば、「自宅の電子メール」のように入力します。
アドレス	電子メール・アドレスを入力します。たとえば、「myAccount@somewhere.com」のように入力します。
最大通知	この電子メールで 1 日当たりに受信する最大通知数を入力します。受信する電子メール・メッセージ数に制限を設定しない場合は、このフィールドを空白のままにします。

図 7-19 電子メール・アドレスの追加



電子メール・デバイスの検証

電子メールを作成すると、デバイスの検証を指示する OracleAS Wireless のプロンプトが表示されます。

注意： 検証プロセスを開始する前に、電子メールを使用可能な状態にしてください。

電子メールを検証するには、**「送信」**をクリックして検証コードを電子メールに送信します。このコードを受信したら、画面にこのコードを入力して**「検証」**をクリックします。デバイスが正常に検証された場合は、「ステータス」列に「有効」が表示されます。

電子メール・デバイスの編集

電子メール・デバイスは、名前、番号、または受信する通知数を変更することによって更新できます。電子メールを更新するには、該当する電子メールを選択して**「編集」**をクリックします。フィールドに選択したデバイスに固有の値が移入された状態の「編集」画面が表示されます。これらの値の詳細は、7-26 ページの「電子メール・デバイスの作成」を参照してください。必要な変更を実行後、デバイスを検証します。デバイスの検証の詳細は、7-27 ページの「電子メール・デバイスの検証」を参照してください。変更内容を保存するには、**「終了」**をクリックします。**「取消」**をクリックすると、電子メールのパラメータが以前の値に戻ります。

電子メール・デバイスの削除

電子メールを削除するには、「デバイス・リスト」から該当する電子メールを選択し、**「削除」**をクリックします。「確認」画面が表示されます。削除を承認するには**「はい」**を、取り消すには「いいえ」を選択します。

新しいモバイル・デバイスの作成

新しいモバイル・デバイスを作成するには、ドロップダウン・リストから「モバイル機器」を選択し、次に**「追加」**をクリックします。「モバイル機器の追加」画面（図 7-20）が表示されます。モバイル・デバイスの追加を完了するには、表 7-9 の値を定義する必要があります。パラメータを定義した後、**「終了」**をクリックします。**「取消」**をクリックすると、入力した値が消去され、参照ページに戻ります。

表 7-9「モバイル機器の追加」ページのパラメータ

パラメータ	値
名前	モバイル・デバイスの名前を入力します。たとえば、「自宅の電子メール」のように入力します。
アドレス	モバイル・デバイスのアドレスを入力します。たとえば、「myAccount@somewhere.com」のように入力します。
最大通知	このモバイル・デバイスで 1 日当たりに受信する最大通知数を入力します。このデバイスで受信するメッセージ数に制限を設定しない場合は、このフィールドを空白のままにします。
デバイス・メーカー	ドロップダウン・リストからメーカーを選択します。デフォルト設定は「不明」です。
デバイス・モデル	リストからデバイスの機種を選択します。デフォルト設定は「不明」です。
優先チャネル	デバイスの優先チャネルを選択します。 チャネルに適切な書式でアドレスを入力します。たとえば、音声チャネルのアドレスは電話番号、電子メール・チャネルのアドレスは電子メール・アカウントです。IM チャネルのアドレスは、リストから選択した Messenger Network によって異なります。固有のアドレス書式が不明な場合は、ISP にお問合せください。

図 7-20 モバイル機器の追加（一部のみ）

モバイル・デバイスの検証

モバイル・デバイスを作成すると、デバイスの検証を指示する OracleAS Wireless のプロンプトが表示されます。

注意： 検証プロセスを開始する前に、モバイル・デバイスの電源を入れてください。

モバイル・デバイスを検証するには、**「送信」**をクリックして検証コードをモバイル・デバイスに送信します。このコードを受信したら、画面にこのコードを入力して**「検証」**をクリックします。デバイスが正常に検証された場合は、「ステータス」列に「有効」が表示されます。

モバイル・デバイスの編集

モバイル・デバイスは、名前、番号、または受信する通知数を変更することによって更新できます。モバイル・デバイスを更新するには、該当するモバイル・デバイスを選択して**「編集」**をクリックします。フィールドに選択したデバイスに固有の値が移入された状態の「編集」画面が表示されます。これらの値の詳細は、7-28 ページの「新しいモバイル・デバイスの作成」を参照してください。必要な変更を実行後、デバイスを検証します。デバイスの検証の詳細は、7-29 ページの「モバイル・デバイスの検証」を参照してください。変更内容を保存するには、**「終了」**をクリックします。**「取消」**をクリックすると、モバイル・デバイスのパラメータが以前の値に戻ります。

モバイル・デバイスの削除

モバイル・デバイスを削除するには、「デバイス・リスト」から該当するモバイル・デバイスを選択し、**「削除」**をクリックします。「確認」画面が表示されます。削除を承認するには**「はい」**を、取り消すには「いいえ」を選択します。

デフォルトのデバイスの設定

デフォルトのデバイスを設定するには、「デバイス・リスト」から該当するデバイスを選択し、**「デフォルトを設定」**をクリックします。

ロケーション・マークの管理

ロケーション・マークは、ユーザーの自宅、オフィス、勤務先関連所在地などのユーザー定義のロケーションです。エンド・ユーザーは、各自のロケーション認識アプリケーションにこれらのロケーションを入力できます。レストラン検索アプリケーションなどのロケーション認識アプリケーションでは、目的地までの運転方向で参考になる地点を提供するために、ユーザーの自宅所在地などのエンド・ユーザーの現在地を使用できます。セキュリティとプライバシを確保するために、ユーザーは自分のロケーションにアクセスできるアプリケーションを制御できます。

ロケーション・マークを作成するために、モバイル・デバイスに長い英数文字列を入力する必要はありません。Wireless Customization では、ユーザーが、ロケーション・マークの基礎となる空間情報を入力および管理できます。これらの情報は Wireless リポジトリに格納されます。ユーザーは、モバイル・デバイスでロケーション・マークを選択して、空間情報にアクセスします。

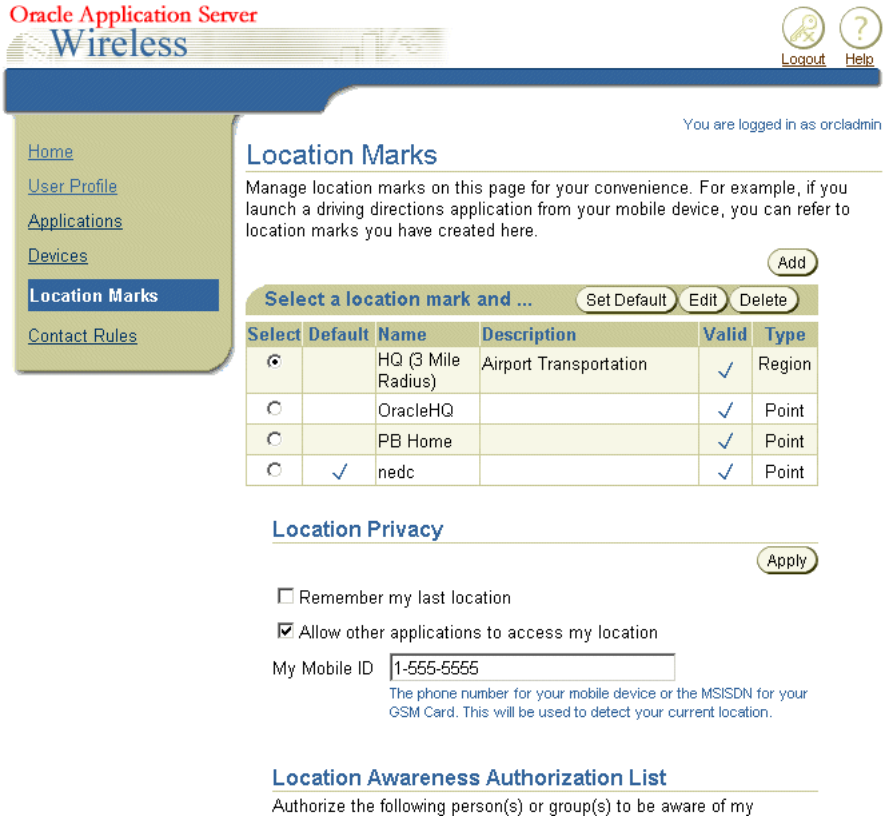
ロケーション・ベースの通知サブスクリプションでは、領域タイプのロケーション・マークを使用できます。たとえば、オフィス所在地から半径 3 マイル以内にある勤務先所在地に対して、領域タイプのロケーション・マークを作成できます。たとえば、オフィスから空港までの乗り物をクライアントのために手配する場合は、「空港のリムジン・バスがオフィスから 3 マイル以内に入ったときに通知してください」などの通知をサブスクライブすることになります。このように指定すると、リムジン・バスがオフィスから 3 マイル以内に入る都度、通知メッセージが届きます。

注意： ロケーション・マークは、地点タイプのロケーションまたは領域タイプのロケーションとして作成できます。領域タイプのロケーションは、地点と半径で定義する円の地域として表現できます。また、レッドウッド・シティ、カリフォルニア、またはカリフォルニア州全体など、システム定義の地図領域としても表現できます。

ジオコーディング機能は、ベンダーによるジオコーディングされたデータにサーバーがアクセスしないと機能しません。ロケーション・マークは、経度と緯度のジオメトリ値がない場合でも作成できます。このタイプのロケーション・マークは、ロケーション情報の目的のみに使用でき、ジオメトリ情報が必要なアプリケーションでは使用できません。

ロケーション・マークを作成、編集および削除する機能にアクセスするには、「ロケーション・マーク」タブをクリックします。参照画面（図 7-21）に、現行のロケーション・マークの一覧表が表示されます。

図 7-21「ロケーション・マーク」画面（一部のみ）

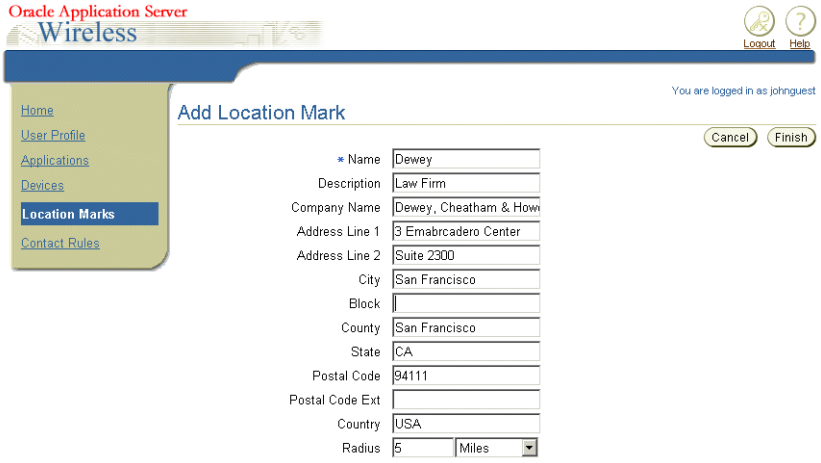


ロケーション・マークの作成

ロケーション・マークを作成するには、最初に**「追加」**ボタンをクリックします。「ロケーション・マークの追加」画面（図 7-22）が表示されます。ロケーション・マークを作成するには、次の各パラメータを定義する必要があります。

1. 「ロケーション・マーク名」フィールドに意味のある名前を入力します。これは必須フィールドです。
2. ロケーション・マークに関する意味のある説明（「勤務先」や「自宅」など）を入力します。
3. ロケーション・マークのラベルを入力します。
4. 「会社名」フィールドに会社名を入力します。
5. 「住所行 1」に番地情報を入力します。たとえば、「123」のように入力します。
6. 「住所行 2」に街路情報を入力します。たとえば、「Main Street」のように入力します。
7. 「都道府県」および「郡」フィールドに都道府県および郡（必要な場合）の情報を入力します。
8. 「郵便番号 1」フィールドに郵便番号を入力します。これは、米国の 5 桁の郵便番号や他の郵便番号の場合があります。
9. 「郵便番号 2」フィールドに拡張郵便番号（必要な場合）を入力します。
10. 「国」フィールドに国名を入力します。
11. 領域タイプのロケーション・マークを作成する場合は、距離の範囲値と単位（km またはマイル）を入力します。
12. **「適用」**をクリックして、ジオコーディング・プロセスをトリガーします。

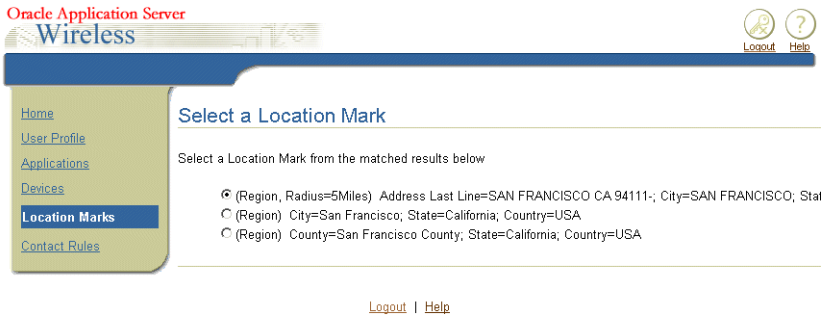
図 7-22 ロケーション・マークの作成



ジオコーディング・プロセスに障害が発生した場合は、警告ページが表示されます。ジオコーディング情報なしでロケーション・マークを保存するかどうかを指定します。ロケーションを保存するには**「保存」**をクリックします。**「戻る」**をクリックすると、他の値を入力するための作成ページに戻ります。

ロケーション・マークの基準に一致する結果が複数見つかった場合は、選択するページ（図 7-23）が表示され、その中から最適なものを選択します。結果には、対応する市区町村や都道府県に一致するロケーション・マークなど、領域タイプのロケーション・マークを含めることができます。

図 7-23 ロケーション・マークの選択



保存するロケーション・マークを選択してから**「保存」**をクリックし、選択したロケーション・マークを保存します。

「取消」をクリックすると、ロケーション・マークは破棄されます。

ロケーション・マークの編集

ロケーション・マークを編集する手順は、次のとおりです。

ロケーション・マーク・リスト画面から、該当するロケーション・マークを選択して**「編集」**ボタンをクリックします。フィールドに選択したロケーション・マークの設定値が移入された状態の「ロケーション・マークの編集」画面（図 7-24）が表示されます。必要に応じてフィールドを編集します。ロケーション・マークの値の入力方法の詳細は、7-33 ページの「ロケーション・マークの作成」を参照してください。**「適用」**をクリックして、ジオコーディング・プロセスをトリガーします。

図 7-24 ロケーション・マークの編集

ロケーション・マークのデフォルト・ステータスの変更

ロケーション・マークのデフォルト・ステータスを変更するには、ロケーション・マーク・リスト画面で、新しいデフォルトのロケーション・マークを選択します。**「デフォルトを設定」**をクリックします。ロケーション・マーク・リストの「デフォルト」列のステータスが true に変わります。デフォルトにできるロケーション・マークは 1 つのみです。

ロケーション・マークの削除

ロケーション・マークを削除するには、該当するロケーション・マークを選択して**「削除」**をクリックします。「確認」画面が表示されます。削除を承認するには「はい」を選択します。削除を取り消すには**「いいえ」**をクリックします。

ロケーション・プライバシ作業環境の設定

ロケーション・プライバシを設定するには、ロケーション・マーク・リスト画面にリストされている次のオプションから選択します。

- 必要な場合は、「最終マイ・ロケーションを記憶」を選択して、ロケーション・マークをキャッシュします。
- ロケーションを開示するには「他のアプリケーションからのマイ・ロケーションへのアクセスを許可」をチェックします。
- ユーザーのポジショニング ID として使用するモバイル ID も設定できます。このモバイル ID によって、モバイル・ポジショニング・サーバーはユーザーの現在位置を検出できます。ユーザーの現在位置をロケーション認識アプリケーションに公開する場合は、この値を設定します。

「ロケーション・プライバシ」セクションで**「適用」**をクリックし、ロケーション・プライバシ作業環境設定を保存します。

ロケーション認識認証の管理

ローケーション認識認証を使用すると、ユーザーの現在位置を別のユーザーが指定した期間検出できます。

ロケーション認識認証は、ユーザーの位置や移動を監視するために他のユーザーがロケーション・ベースの通知をサブスクライブする場合に使用されます。たとえば、今日午前 8 時から午後 5 時までの居場所について、上司の監視を許可する場合は、時間基準に基づいて認証ルールを作成して、そのルールを有効にできます。すると、上司は、「従業員がオフィスから 3 マイル移動するたびに通知してください」などの位置に関する通知をサブスクライブできます。

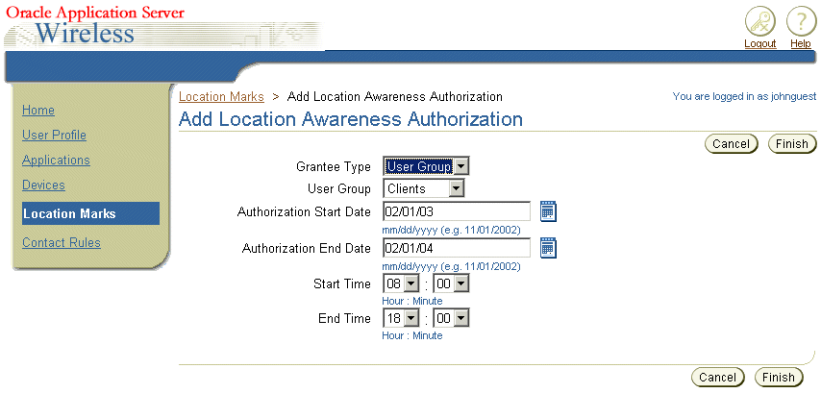
「ロケーション・マーク」画面には、ロケーションの監視を許可されているユーザーの一覧表が含まれています。ロケーション・プライバシと認証ルールは、ロケーション・ベースのアプリケーションで使用されます。ロケーション・ルールによって、ユーザーは、自分のロケーションをロケーション・ベースのアプリケーションに開示できるかどうか（および開示する時間）を制御できます。認証ルールは、作成、編集、削除、有効化または無効化できます。認証ルールに使用するロケーション認識ユーザー・グループも管理できます。

ロケーション認識認証の割当て

ロケーション認識認証を作成する手順は、次のとおりです。

「ロケーション認識認証リスト」画面（図 7-25）で、**「追加」**ボタンをクリックします。「ユーザー」または「ユーザー・グループ」いずれかの権限受領者タイプを選択してから、権限受領者の値を指定します。権限受領者のタイプに「ユーザー」を選択した場合はユーザー名を入力し、「ユーザー・グループ」を選択した場合はグループ選択リストから該当するグループを選択します。認証期間を指定します。変更内容を保存するには、**「終了」**をクリックします。**「取消」**をクリックすると、すべての値が消去されます。

図 7-25 ロケーション認識の追加

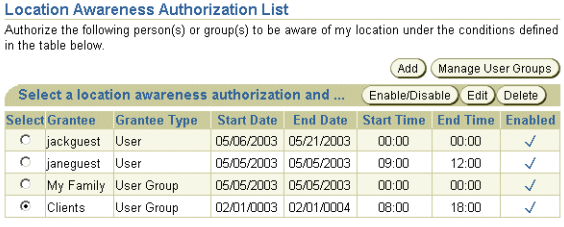


ロケーション認識認証の変更

「ロケーション認識認証リスト」画面（図 7-26）から、該当する認証オブジェクトを選択して**「編集」**ボタンをクリックします。フィールドに選択した認証オブジェクトの設定値が移入された状態の「編集」画面が表示されます。

必要に応じて値を編集します。これらの値の詳細は、7-36 ページの「ロケーション認識認証の割当て」を参照してください。

図 7-26 ロケーション認識認証リスト



Select	Grantee	Grantee Type	Start Date	End Date	Start Time	End Time	Enabled
○	jackguest	User	05/06/2003	05/21/2003	00:00	00:00	✓
○	janeguest	User	05/05/2003	05/05/2003	09:00	12:00	✓
○	My Family	User Group	05/05/2003	05/05/2003	00:00	00:00	✓
◉	Clients	User Group	02/01/0003	02/01/0004	08:00	18:00	✓

ロケーション認識のユーザー・グループの管理

ロケーション認識認証で使用するユーザー・グループは、ユーザー・オブジェクトのコレクションです。『Oracle Application Server Wireless 管理者ガイド』で説明されている、アプリケーションのアクセス制御リスト（ACL）で使用するユーザー・グループの概念とは明確に異なります。ロケーション認識のユーザー・グループは、ロケーション認識認証のターゲット・タイプです。このターゲット・タイプによって、単独のユーザーではなく、複数のユーザーが指定されているグループに対して許可を割り当てることができます。ユーザー・グループは、認可ポリシーを容易に管理できるロケーション認識認証を管理する際の 1 つの権限受領者オブジェクトのタイプです。たとえば、各ユーザーに同じ認可ポリシーを個別に作成するのではなく、10 人のユーザーを含んでいるユーザー・グループに単一の認可ポリシーを作成できます。図 7-26 のユーザー・グループ「My Family」は、個々のユーザーである「jackguest」と「janeguest」で構成されおり、両者ともに「ユーザー」権限受領者タイプが割り当てられています。

「ロケーション認識認証」セクションの**「ユーザー・グループの管理」**ボタンをクリックすると、ユーザー・グループ・リスト・ページが表示されます。この画面では、ユーザー・グループを作成、編集または削除できます。

ユーザー・グループの作成

ユーザー・グループを作成するには、ユーザー・グループ・リスト画面で**「追加」**ボタンをクリックします。「ロケーション認識認証の追加」画面（図 7-25）が表示されます。ユーザー・グループの名前（これは必須フィールド）を入力してから、このグループに挿入するユーザーのリストを入力します。ユーザーは Wireless Customization の有効なユーザーであることが必要です。有効なユーザーでない場合はユーザー・グループを作成できません。**「終了」**をクリックして、ユーザー・グループを作成します。**「取消」**をクリックすると、すべての値が消去されます。

ユーザー・グループの編集

ユーザー・グループを編集するには、更新するユーザー・グループを表から選択します。フィールドに選択したユーザー・グループの設定値が移入された状態の「編集」画面が表示されます。必要に応じて値を編集します。詳細は、7-38 ページの「ユーザー・グループの作成」を参照してください。変更内容を保存するには、**「適用」**をクリックします。**「取消」**をクリックすると、値が以前の状態に戻ります。

ユーザー・グループの削除

該当するユーザー・グループを 1 つ選択し、**「削除」**ボタンをクリックします。「確認」ページが表示されます。選択したユーザー・グループ・オブジェクトを削除するには**「はい」**をクリックします。削除を取り消すには**「いいえ」**をクリックします。

連絡ルールの管理

連絡ルールは、電話とメッセージの受信方法を示します。たとえば、すべてを通知を携帯電話で受信するように会議の連絡ルールを設定できます。Oracle Application Server Wireless では、会議中か外出中かなど、現在の状況が把握され、ユーザーが希望する通知方法でいつでもデバイスを使用できます。

連絡ルールの各設定画面（Oracle Application Server Wireless Customization Portal からアクセス）を使用して、最初に連絡ルールの名前を指定し（「支部」など）、次にその連絡ルールに適した通信デバイスを追加することで、連絡ルールを作成します。通信デバイスの作成時には、デバイスの番号またはアドレス、およびニックネーム（「自分の携帯電話」など）を入力します（Oracle Application Server Wireless Customization Portal を使用したデバイスの作成と管理の詳細は、7-43 ページの「Web ベースのユーザー・インタフェースからの連絡ルールの選択」を参照してください）。連絡ルールには、次の通信方法からデバイスを選択できます。

- 音声
- FAX
- 電子メール
- メッセージ

たとえば、「支部の電話」というデバイスで電話を受信し、「本社の電子メール」デバイスで電子メールを受信し、「支部の FAX」デバイスで FAX を受信できるように、「支部」の連絡ルールを作成できます。連絡ルールを作成するときは、通知の受信に適した方法を指定します。この「支部」連絡ルールの場合は、通知を「本社の電子メール」に送信される電子メール・メッセージとして受信するように選択できます。

Oracle Application Server Wireless には、「Available」と「Unavailable」の 2 種類の事前定義済連絡ルールが用意されています。これらの連絡ルールは、編集して名前を変更できます。ただし、各連絡ルール名は一意であることが必要です。つまり、2 つの連絡ルールに同じ名前は指定できません。また、1 つの連絡ルールを別のユーザーと共有することもできません。

連絡ルールを作成および管理する機能には、Oracle Application Server Wireless Customization Portal のホーム・ページで「連絡ルール」メニューを選択してアクセスします。

Customization Portal での連絡ルール

連絡ルールは、Oracle Application Server Wireless Customization Portal の「連絡ルール」ページで管理できます。

- **カレントに設定**: 選択した連絡ルールをアクティブな連絡ルールとして設定するには、**「カレントに設定」**ボタンをクリックします。
- **編集**: 連絡ルールを編集するには、変更する連絡ルールのラジオ・ボタンを選択してから**「編集」**ボタンをクリックします。
- **削除**: 連絡ルールを削除するには、削除する連絡ルールのラジオ・ボタンを選択してから**「削除」**ボタンをクリックします。
- **追加**: **「追加」**ボタンをクリックして新規連絡ルールを追加します。

連絡ルールの追加

連絡ルールを追加するには、「連絡ルール」ページの**「追加」**ボタンをクリックします。次のフィールドに情報を入力します。

- **連絡ルール**: 連絡ルールの名前を入力します（「自分のデスク」など）。

デバイス設定

- **電話**: 電話に対する通信デバイスを入力します（「自分の携帯電話」など）。電話で通知を受信しない場合は、「電話しないでください」を選択できます。選択肢には、音声機能がある通信デバイスのみがリストされます。
- **電子メール**: 電子メールに対する通信デバイスを入力します（「自分の電子メール」など）。電子メールで通知を受信しない場合は、「電子メールを送信しないでください」を選択できます。選択肢には、電子メール機能がある通信デバイスのみがリストされます。
- **FAX**: FAX を受信する通信デバイスを入力します（「自分の FAX」など）。FAX で通知を受信しない場合は、「FAX を送らないでください」を選択できます。選択肢には、FAX 機能がある通信デバイスのみがリストされます。
- **メッセージ**: ショート・メッセージやインスタント・メッセージなどのメッセージ・データを受信する通信デバイスを入力します（「自分の SMS」など）。この配信方法で通知を受信しない場合は、「メッセージを送信しないでください」を選択できます。選択肢には、このようなメッセージ機能がある通信デバイスのみがリストされます。

通知設定

- **通知チャネル**: 通知を受信するための配信方法を入力します。
- **アクティブ開始時刻**: 通知の配信を開始する時刻を入力します。
- **アクティブ終了時刻**: 終了時刻を入力します。この時刻以降は通知が配信されません。

- **通知日**：アクティブ時間が有効となる週の日（毎日、平日または週末）を入力します。
- **アクティブ時間間隔外**：アクティブ時間外にメッセージが生成された場合に、メッセージの配信をすべて廃棄するか、またはメッセージの配信を遅延するかを指定します。

通知手段に適切な通信デバイスが選択されていることを確認してください。通信デバイスが不適切な場合は通知を受信できません。たとえば、電話で通知を受信する場合は、「電話」配信方法に通信デバイスが選択されていることを確認します。

> **注意：** 通信デバイスを選択するには、その前に通信デバイスを作成しておく必要があります。

情報を入力した後は、**「終了」**をクリックします。

連絡ルールの編集

連絡ルールを編集するには、変更する連絡ルールのラジオ・ボタンをクリックします。次のフィールドに情報を入力します。

連絡ルール：連絡ルールの名前（「自分のデスク」など）を入力するか、または変更します。

デバイス設定

- **電話**：電話に対する通信デバイス（「自分の携帯電話」など）を入力するか、または変更します。電話で通知を受信しない場合は、「電話しないでください」を選択できます。選択肢には、音声機能がある通信デバイスのみがリストされます。
- **電子メール**：電子メールに対する通信デバイス（「自分の電子メール」など）を入力するか、または変更します。電子メールで通知を受信しない場合は、「電子メールを送信しないでください」を選択できます。選択肢には、電子メール機能がある通信デバイスのみがリストされます。
- **FAX**: FAX を受信する通信デバイス（「自分の FAX」など）を入力するか、または変更します。FAX で通知を受信しない場合は、「FAX を送らないでください」を選択できます。選択肢には、FAX 機能がある通信デバイスのみがリストされます。
- **メッセージ**：ショート・メッセージやインスタント・メッセージなどのメッセージ・データを受信する通信デバイス（「自分の SMS」など）を入力するか、または変更します。この配信方法で通知を受信しない場合は、「メッセージを送信しないでください」を選択できます。選択肢には、このようなメッセージ機能がある通信デバイスのみがリストされます。

通知設定

- **通知チャネル**: 必要に応じて、通知を受信するための配信方法を変更します。
- **アクティブ開始時刻**: 通知の配信を開始する時刻を変更します。
- **アクティブ終了時刻**: 終了時刻を変更します。この時刻以降は通知が配信されません。
- **通知日**: アクティブ時間が有効となる週の日（毎日、平日または週末）を変更します。
- **アクティブ時間間隔外**: アクティブ時間外にメッセージが生成された場合に、メッセージの配信をすべて廃棄するか、またはメッセージの配信を遅延するかを指定します。

通知手段に適切な通信デバイスが選択されていることを確認してください。通信デバイスが不適切な場合は通知を受信できません。たとえば、電話で通知を受信する場合は、「電話」配信方法に通信デバイスが選択されていることを確認します。

注意： 通信デバイスを選択するには、その前に通信デバイスを作成しておく必要があります。

連絡ルールの削除

連絡ルールを削除するには、連絡ルール表で該当する連絡ルールのラジオ・ボタンを選択してから、**「削除」**をクリックします。

アクティブな連絡ルールの選択

アクティブにする連絡ルールを選択できます。たとえば、支部にいる場合は、「支部」連絡ルールをアクティブに設定すると、通信とすべての通知について、その連絡ルールの設定内容が有効になります。

アクティブな連絡ルールの設定は、Oracle Application Server Wireless Customization Portal の「連絡ルール」ページから、アクティブに設定する連絡ルールのラジオ・ボタンを選択し、**「カレントに設定」**ボタンをクリックすることで変更できます。

デバイスからの連絡ルールの選択

連絡ルールは、Oracle Collaboration Suite のホーム・ページなどの Web ベースのインタフェースや登録されている通信デバイスから選択できます。

Web ベースのユーザー・インタフェースからの連絡ルールの選択

連絡ルールは、Oracle Collaboration Suite のホーム・ページで、連絡先ドロップダウン・リスト（図 7-27）から選択して**「変更」**をクリックするか、または「拡張」ページで連絡ルールを選択してから**「カレントに設定」**をクリックすることで変更できます。

図 7-27 Oracle Collaboration Suite からの連絡ルールの選択

デバイスからの連絡ルールの選択

連絡ルールは、様々なデバイスからも選択できます。これは、OracleAS Wireless の XML では、Oracle Application Server Wireless アプリケーションから様々なデバイス固有のマークアップ言語への XML 変換が可能なためです。その結果、WAP 対応デバイスや一般の電話から、連絡ルールを選択できます。また、非同期対応のアプリケーションでは、SMS や電子メールなどの非同期メッセージ・アプリケーションを備えたデバイスから連絡ルールを選択できます（ただし、インターネット・アクセスはありません）。これらのデバイスから連絡ルールを変更するには、メッセージを、システム管理者が設定した非同期 SMS や電子メール・アドレスに送信します。

WAP 対応の携帯電話などのデバイスを使用して、表示されたリストから連絡ルールを選択します。連絡ルールを変更すると、OracleAS Wireless によって、それまでのルールの設定（連絡手段の制御）から新しい連絡ルールの設定に切り替わります。

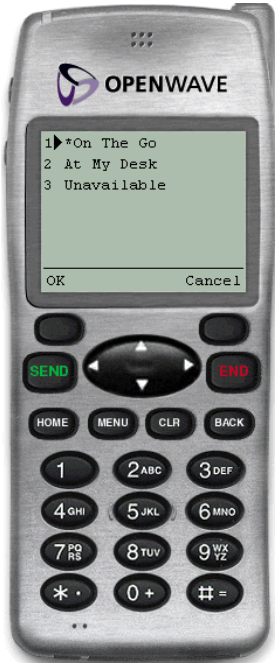
次の内容について各項で説明します。

- デバイスからの連絡ルールの選択
- SMS ベースまたは電子メール・ベースのデバイスからの連絡ルールの選択
- 音声アプリケーションを使用した連絡ルールの選択

デバイスからの連絡ルールの選択

携帯電話などのモバイル・デバイスの場合、連絡ルールはリスト形式で表示され、現行の連絡ルールにはアスタリスク（*）が付きます。図 7-28 の例では、「外出中」が現行の連絡ルールです。デバイスのナビゲーション・キーを使用して「OK」を選択し、新しい連絡ルールを選択します。

図 7-28 デバイスからの連絡ルールの選択



新しい連絡ルールを示す「確認」画面（図 7-29）が表示されます。**「OK」**をクリックするとメイン・メニューに戻ります。

図 7-29 確認ページ（デバイスから）

SMS ベースまたは電子メール・ベースのデバイスからの連絡ルールの選択

非同期アプリケーションを使用しているデバイスから、非同期 SMS アドレスまたは電子メール・アドレスへのメッセージとしてコマンドを送信することで、連絡ルールを設定できます。連絡ルールは、複数のメッセージを使用して次のように設定できます。

方法 1 この方法では、次の 3 つのメッセージを個別に送信して、連絡ルールを変更します。

メッセージ 1: メッセージの件名行または本文に「cr」と入力します。すると、携帯電話の番号と PIN 番号の入力を指示するメッセージが着信します（詳細は、7-40 ページの「Customization Portal での連絡ルール」を参照してください）。

メッセージ 2: メッセージの件名行または本文に、携帯電話の番号と PIN 番号を入力します。電子メールの本文でこの情報を送信する場合は、これらの番号を同じ行に入力する必要があります。すると、連絡ルールの番号付きリストとともにメッセージが着信します。

メッセージ 3: 新しい連絡ルールの番号をメッセージの件名行または本文に入力します。たとえば、番号付きリストから「2. At My Desk」を選択する場合は、「2」を入力します。する

と、連絡ルールの変更を確認するメッセージが着信します。受信した後は、メイン・メニューに戻ることができます。

方法 2 この方法を使用すると、cr コマンドと正確な連絡ルール名を組み合せて、2 つのメッセージを個別に送信することで、連絡ルールを変更できます。

メッセージ 1: メッセージの件名行または本文にある連絡ルールの名前の前に、「cr」を入力します。たとえば、「cr "At My Desk"」のように入力します。連絡ルール名に空白が使用されている場合は、名前全体を引用符（"）で囲む必要があります。連絡ルール名では、大 / 小文字も区別されます。このメッセージを送信すると、使用しているユーザー名とパスワードによる返信を指示するメッセージが着信します。

メッセージ 2: メッセージの件名行または本文に、携帯電話の番号と PIN 番号を入力します。電子メールの本文でこの情報を送信する場合は、これらの番号を同じ行に入力する必要があります。このメッセージを送信すると、連絡ルールの変更を確認する返信を受け取ります。受信した後は、メイン・メニューに戻ることができます。

方法 3 cr コマンド、連絡ルール名、およびユーザー名とパスワードを、まとめてメッセージの件名行または本文に指定し、単一のメッセージ送信で連絡ルールを変更することもできます。たとえば、次のようにメッセージの件名行または本文に情報をすべて入力して、新しい連絡ルールを選択できます。

cr "At My Desk"; 16505555000 12345

注意： セミコロン（;）を使用して、cr と連絡ルール名のコマンド部分から、ユーザー名とパスワード部分を区切ります。

このメッセージを送信すると、連絡ルールの変更を確認する返信を受け取ります。受信した後は、メイン・メニューに戻ることができます。

注意： 連絡ルール名に空白が使用されている場合は、連絡ルール名全体を引用符（"）で囲む必要があります。連絡ルール名では、大 / 小文字も区別されます。

音声アプリケーションを使用した連絡ルールの選択

ダイヤル後、次の手順を実行します。

1. 携帯電話の番号を入力します。詳細は、7-40 ページの「Customization Portal での連絡ルール」を参照してください。
2. PIN を入力します。プロンプトに従って PIN を確認します。
3. 「連絡ルール」と告げて、連絡ルール・アプリケーションを起動します。最初に現在の連絡ルール、次に使用可能な連絡ルールのリストが告知されます。
4. 新しい連絡ルール名を告げます。たとえば、「At My Desk」のように告げます。システムからの返信で変更を確認し、メイン・メニューに戻ります。

Netscape 4.7 以前を使用してローカライズされた言語での UTF-8 のページの表示

Netscape 4.7 以前のバージョンを使用している場合は、一部の言語が正しく表示されない可能性があります。場合によっては、文字がボックスで表示されることがあります。この問題を修正するには、次の手順で Netscape 作業環境を設定します。

1. Netscape のツールバーから、**「編集」**を選択します。
2. ドロップダウン・メニューから**「設定」**を選択します。「設定」ダイアログが表示されます。
3. 「カテゴリ」ツリーから**「フォント」**を選択すると、「フォント」ダイアログが表示されます。
4. 「フォント」ダイアログで、「文字コード セット」ドロップダウン・リストから**「Unicode」**を選択します。
5. 「プロポーショナル フォント」および「固定ピッチ フォント」ドロップダウン・リストから、優先言語をサポートするフォントを選択します。たとえば、優先言語に中国語を選択している場合は、MS Song を選択して中国語でページを表示できます。

Customization Portal の特性変更

OracleAS Wireless の Customization Portal は、Customization インタフェースのフレームワークと、そのフレームワークのサンプル実装を兼ねています。このフレームワークは、UI ベースの XML ページ（UIX）ファイル、JavaBean モジュール、JavaScript、および XSL スタイルシートや HTML ファイルなどの静的要素で構成されています。フレームワークのもう 1 つの要素は、カスタマイズされたページ・プラグインです。既存のフレームワークに基づいて Customization Portal の特性を変更するか、またはユーザー独自のサービス・カスタマイズをプラグインするか静的イメージを置換して、フレームワーク自体を再構築できます。

次の項では、Customization Portal を構成する要素、プラグイン・ページのフレームワーク、およびファイル・ネーミング規則と使用するディレクトリ構造について説明します。

ページ・ネーミング規則

UIX は、Web アプリケーションを構築するための拡張可能な J2EE ベースのフレームワークです。このフレームワークは、Model-View-Controller（MVC）設計パターンに基づいています。UIX ページでは、ページのレイアウトやスタイルも含めたユーザー・インタフェースなどの View レイヤーを定義します。UIX ファイルに関係するプログラミングはなく、変更はコンパイルなしでデプロイできます。Model レイヤーと Controller レイヤーは、すべて JavaBean ファイルにあります。

各 UIX ファイルには、ページ・イベントや動的データ取得を処理するコントローラ Java ファイルが 1 つあります。各モデル・オブジェクトには、これに対応して、Model API レイヤーと直接インタフェースする Java ファイルがあり、UIX ページのキャッシュを処理する別のラッパー Java ファイルがあります。たとえば、各デバイス管理ページには、コントローラ・ファイル DeviceHandler.java が 1 つ、データ・モデル・ファイル DeviceDataObject.java が 1 つ、およびラッパー・ファイル UIXDevice.java が 1 つあり、デバイス管理 UIX ページのオブジェクトのキャッシュを処理します。

- オブジェクトの要約リストのメイン UIX ページでは、ページ名に複数形の名前が使用されます。例 : Devices.uix
- 編集用の詳細 UIX ページは、Edit で開始します。例 : EditMobile.uix
- 作成用の詳細 UIX ページは、Add で開始します。例 : AddMobile.uix
- コントローラ Java ファイルでは、名前に ...Handler.java が使用されます。
  例 : DeviceHandler.java
- モデル Java ファイルでは、名前に ...DataObject.java が使用されます。
  例 : DeviceDataObject.java

UIX ページの構造

各 Customization Portal の UIX ページは、一連の UIX コンポーネントで構成されています。

- 特性設定
- ナビゲーション
- グローバル・ボタン
- ページ・コンテンツ領域
- フッター

表 7-10 に、UIX コンポーネントを示します。

表 7-10 UIX コンポーネント

UIX コンポーネント	説明
フォームおよびページ・レイアウト	ヘッダーとフッターを設定し、その他のコンテンツ用にページの残り部分を予約します。このコンポーネントには、タブ・バー、ナビゲーション・トレース、行レイアウトおよびボタンの各要素が含まれます。
ヘッダー	会社名とグローバル・ボタン。
ナビゲーション・トレース	ナビゲーション・キューと表示名の要素が表示されます。
フッター	グローバル・ボタン・リンクと著作権情報。
ページ・コンテンツ	メイン・コンテンツ、フォーム項目およびページ・ボタンが含まれています。

ディレクトリ構造

Customization Portal の特性を変更するには、Customization Portal を生成する UIX ファイルを変更します。OracleAS Wireless をインストールすると、これらのファイルは次の構造を持つ $ORACLE_HOME/OC4J_Wireless/j2ee/applications/mobile/mobile-web ディレクトリに格納されます。

表 7-11 に、Portal ディレクトリの内容を示します。

表 7-11 Portal ディレクトリの内容

ディレクトリ	内容
customization	コンテナ UIX ファイル。コンテナ・ファイルには、ブラウザが直接アクセスします。
images	Customization Portal 全体で使用されるイメージ。
customization/templates	UIX テンプレート・ファイル。これらのファイルは、コンテナ UIX ファイルまたは他のテンプレート・ファイルによってインクルードされます。
cabo	JavaBean スタイルシート、イメージおよび JavaScript。
cabo/images/cache	JavaBean の生成済イメージ。
cabo/jsps	UIX によって使用される Java サーバー・ページ。
cabo/styles/cache	生成済スタイルシート。

Customization Portal の外観のカスタマイズ

Portal のページは様々な方法でカスタマイズでき、ロゴ、バナーおよびアイコンの外観を変更できます。また、希望する外観と操作方法を実現するために、独自の UIX または JSP を作成することもできます。

$ORACLE_HOME/OC4J_Wireless/j2ee/applications/mobile/mobile-web/customization/templates ディレクトリの base.uit、basicFlow.uit および advancedFlow.uit の各ファイルの静的文字列を変更することによって、Customization Portal の外観をカスタマイズできます。この静的文字列でコールされるファイル名を変更して、バナー・イメージ、ロゴ・イメージおよびツール・ヒント・テキストを変更できます。ラベルは、次のディレクトリに配置されているリソース・ファイル customization.properties にあります。

$ORACLE_HOME/wireless/server/classes/messages/oracle/panama/webtool/common/resources

UI ラベルは、リソース・ファイル内の対応する文字列値を置換することによって変更できます。

UIX の論理ページは、ユーザー・インタフェース・ノードと呼ばれる階層構造の一連のコンポーネントで構成されています。一部のノードは、ボタン、イメージ、表、テキスト・フィールドなどの表示可能なコンポーネントを定義します。一方、他のノードのレイアウトや外観を編成し、その動作も管理するノードがあります。

色とフォント

色とフォントをカスタマイズするには、次の XML スタイルシート・ファイルを変更します。

$ORACLE_HOME/uix/cabo/styles/blaf.xss

変更後に、

$ORACLE_HOME/j2ee/OC4J_Wireless/applications/mobile/mobile-web/

cabo/styles/cache ディレクトリを削除してサーバーを再起動します。

新しい色とフォントが Web ページで有効になります。

UIX の変更

base.uit、basicFlow.uit および advancedFlow.uit の各 UIX テンプレート・ファイルは、Customization Portal のページ・テンプレートを生成します。base.uit ファイルは、basicFlow.uit および advancedFlow.uit にインクルードされます。basicFlow.uit または advancedFlow.uit は、他の UIX ファイルにインクルードされます。テンプレート・ファイルを変更すると、そのテンプレート・ファイルを使用しているすべてのページに変更が自動的に継承されます。

base.uit（表 7-12）はロゴを生成します。

表 7-12 base.uit 文字列の使用方法

UIX コンポーネント	属性値	ページ要素
productBranding	image source="images/wireless_logo.gif"	ページ・ロゴ・イメージ。
productBranding	data:shortDesc="comm on.brand.desc@labelBundle"	ページ・ロゴのツール・ヒント・テキスト。

basicFlow.uit（表 7-13）はグローバル・ボタンを生成します。basicFlow.uit は、テンプレート・ページ base.uit をインクルードします。

表 7-13 basicFlow.uit 文字列の使用方法

UIX コンポーネント	属性値	ページ要素
globalButton	Source="images/logout_ena.gif"	グローバル・ボタンのイメージ。
globalButton	destination="/mobile/login.uix?event=logout"	グローバル・ボタンのイベント処理。

advancedFlow.uit（表 7-14）はグローバル・ボタンとサイド・ナビゲーション・タブ・バーを生成します。advancedFlow.uit は、テンプレート・ページ base.uit をインクルードします。services.uix は、ユーザーがアクセスできるアプリケーションの階層ビューを表します。services.uix は、テンプレート・ページ advancedFlow.uit をインクルードします。

表 7-14 advancedFlow.uit 文字列の使用方法

UIX コンポーネント	属性値	ページ要素
globalButton	Source="images/logout_ena.gif"	グローバル・ボタンのイメージ。
globalButton	destination="/mobile/login.uix?event=logout"	グローバル・ボタンのイベント処理。
sideNav	link data:text="common.tab.home@labelBundle"	サイド・ナビゲーション・タブのテキスト。
sideNav	destination="advancedSetup.uix"	サイド・ナビゲーションの移動先ページ。

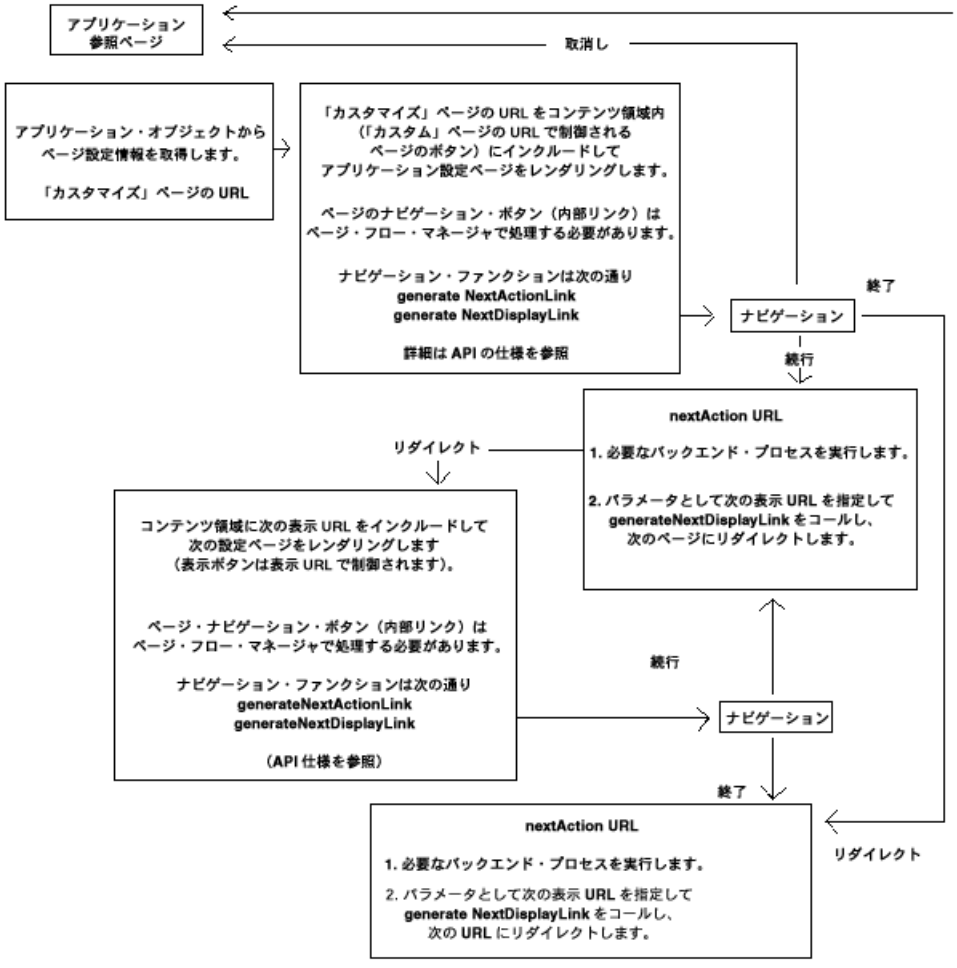
アプリケーションのカスタマイズ・ページのプラグイン・フレームワーク

Customization Portal は、アプリケーション（サービス）のカスタマイズ・ページをプラグインするためのフレームワークを提供します。初期のアプリケーションのカスタマイズ・リンクは、OracleAS WirelessTools で定義する必要があります。

- アプリケーション・ノードにカスタマイズ・リンクが定義されている場合は、「カスタマイズ」ページを起動するために、「カスタマイズ」アクション・リンクがレンダリングされます。
- アプリケーションのカスタマイズ・リンクのコンテンツがレンダリングされるのは、ページ・コンテンツ領域内のみです。この領域では、引き続き `pageFlow` エンジンによって、ページのヘッダー、ナビゲーションおよびフッターが制御およびレンダリングされます。
- `PageFlowManager` は、URL にキーと値を追加して複数ページに回し、HTTP リクエストのページ・フロー値を管理します。
- 「カスタマイズ」ページのすべてのリンクは、`generateNextActionLink` をコールして、生成されたアクション・リンクの URL をフェッチする必要があります。
- リダイレクトされたすべてのページは、`generateNextDisplayLink` をコールして、生成された次のページ・リンクの URL をフェッチする必要があります。
- プラグイン・ページは、Customization Portal の主なページをレンダリングするサーバーとは異なるサーバーからレンダリングされる可能性があるため、すべてのイメージ・ソースは、完全修飾された URL パス（http://server:port/component/images/file.gif など）を使用する必要があります。
- セッション・レベルのキャッシュは、現在の `PageFlowManager` フレームワークの実装ではサポートされません。したがって、中間キャッシュ・オブジェクトでは、リポジトリにオブジェクトを一時的に格納するなど、オブジェクトをキャッシュするための代替手段の使用が必要な場合があります。

図 7-30 に、カスタマイズ・プロセスを示します。

図 7-30 アプリケーションのカスタマイズの流れ



プラグイン・ページでのアプリケーションのカスタマイズ

pluginService.uix は、プラグイン・カスタマイズ・ページが含まれたメイン・ページです。プラグイン URL の値は、アプリケーション（サービス）オブジェクトから取得され、必要なページ・フロー・パラメータを URL に連結する `generatePluginLink` API を使用して構築されます。連結したパラメータには、アプリケーション・オブジェクト ID、ユーザー・オブジェクト ID、GUID およびメイン・ページのページ・フロー情報が含まれています。

Customization Portal に対するマルチバイト・コード体系の設定

Customization Portal は、リポジトリにある PAPZ デバイス内の設定から、サイトのテキストのコード体系を取得します。デフォルトのコード体系は UTF-8 で、これは西欧言語とともに一部のアジア言語を処理できます。Personalization Portal では、各ページのコンテンツがロジカル・デバイスで指定されたコード体系を使用して設定されます。デフォルトのコード体系を変更するには、Foundation Manager の「デバイス」参照画面にリストされている**「PAPZ」**をクリックし、特定言語の IANA 規格に従ってコード体系を変更します。

UI ラベルは customization_LANGUAGE（存在する場合は _COUNTRY）からロードされます。たとえば、customization_fr_CA.properties は、次のディレクトリにあります。

$ORACLE_HOME/OC4J_
Wireless/j2ee/server/classes/messages/oracle/panama/webtool/customization

ログインする前に、OracleAS Wireless のロケール設定によりロケールが決定されます。ログイン後は、ユーザーのロケール作業環境によりロケール設定が決定されます。

第 III 部

Wireless アプリケーションの開発

第 III 部では、Oracle Application Server Wireless を使用した Wireless アプリケーションの開発について説明します。

8

モバイル・ブラウザおよび音声アプリケーションの作成

項ごとに様々なトピックを記載しています。各項の内容は、次のとおりです。

概要

ワイヤレス技術の進歩によって、様々な機能を備えた各種のモバイル・デバイスが開発され、そのフォーム要素も多岐にわたります。これに加えて、各種デバイスでは、HTML、WML、cHTML、XHTML など様々なマークアップ言語を使用してモバイル・アプリケーションが作成されています。OracleAS Wireless によって、ブラウザ・ベースの音声アプリケーションおよびメッセージ・アプリケーションを作成する単一の開発モデルと環境が提供されるため、開発者は、デバイス固有のマークアップ言語に関係なく開発を行うことができます。このマルチチャネル・ソリューションを使用することによって、開発者はこれまでより短期間で開発手順を習得できます。また、OracleAS Wireless を使用して開発されたアプリケーションが将来にわたって引き続き使用可能で、マークアップ言語の変更や拡張に伴う新しいデバイス上でも機能することが保証されます。

OracleAS Wireless では、開発者に対して次の 3 つの開発オプションを提供します。

- XHTML+XForms
- XHTML MP
- OracleAS Wireless XML

次の表に、各開発モデルを使用して開発可能なアプリケーションのタイプを示します。

表 8-1 様々なモデルを使用して作成可能なアプリケーションのタイプ

アプリケーションのタイプ / 開発言語	XHTML+XForms	XHTML MP	OracleAS Wireless XML
ブラウザ・ベース	X	X	X
音声	X		X
通知（SMS、電子メール、MMS、IM）	X		X
非同期	X		X

注意： 音声または SMS/Instant Messaging のアクセス・チャネルでは、XHTML MP はサポートされていません。

この章では、OracleAS Wireless でサポートされる次のマルチチャネル作成モデル、およびそれぞれのモデルでサポートされる機能について説明します。

- 音声およびビジュアル・メディア（SMS および Instant Messaging インタフェースを含む）用の、XForms と CSS を使用する XHTML
- モバイル・ブラウザを使用するビジュアル・メディア用の XHTML Mobile Profile
- Wireless XML: 音声およびビジュアル・メディア（SMS および Instant Messaging インタフェースを含む）用に OracleAS Wireless で定義された抽象表現指向の XML 言語

MobileXML と XHTML/XForms の選択

開発者は、様々なチャネルやデバイスに Wireless コンテンツを提供する必要があります。アクセス・チャネルとデバイスはエンド・ユーザーが選択するため、アプリケーションまたは文書のコンテンツは、すべてのユーザーがアクセス可能であることが必要です。PC ブラウザ・チャネルでは、通常、文書のコンテンツは Hypertext Markup Language（HTML）などのマークアップ言語を使用して表現され、HTML ブラウザを介してエンド・ユーザーに配信されます。ハードウェアの制限や帯域幅の狭さのため、HTML タグをモバイル・デバイスでサポートするのは困難です。また、多くのマークアップ言語はコンテンツとスタイルが文書内に混在しているため、ブラウザでは、表現やスタイルをコンテンツと区別するのは困難です。たとえば、`<font>` タグを使用して、文書内の重要なセクションをマークするとします。これは、色をサポートしているビジュアル・メディアでは適切ですが、色をサポートしていないデバイス上では役立ちません。

デバイスに依存しない方法で文書をコーディングして表示するには、マルチチャネル作成モデルが必要です。Oracle Application Server Wireless は、この問題を解決するために、多くのチャネルおよびデバイス（メッセージ、音声、マイクロ・ブラウザ、PDA ブラウザなど）で使用可能な XML マークアップの MobileXML を提供します。デバイスに依存しない XML の重要性が認識されるに従って、標準規格を確立する努力が払われました。その結果、World Wide Web Consortium（W3C）によって開発されたのが eXtensible Hypertext Markup Language（XHTML）です。XHTML は、MobileXML にかわる標準規格のソリューションです。

XHTML 1.0 規格は HTML の拡張で、XML 1.0 アプリケーションと呼ばれます。XHTML を使用すると、デスクトップ、PDA、音声電話および携帯電話の間で基本的なコンテンツを共有できます。さらに、XHTML 要素を抽象モジュールのコレクションにグループ化し、各モジュールで特定の機能を提供できます。カスケード・スタイルシート（CSS）および XForms は、XHTML とともに使用します。

カスケード・スタイルシート（CSS）は、レンダリング・スタイルと XML 文書の構造を分離します。つまり、CSS では、フォント、空白などを使用する文書の表示方法を示します。CSS を使用して、XML 文書または XHTML 文書のマークアップ部分または要素に表現スタイルを関連付けるため、コンテンツ文書を変更する必要はありません。CSS には、CSS1 と CSS2 の 2 つのレベルがあります。CSS2 は CSS の第 2 世代で、メディア固有のスタイルをサポートするために CSS1 に追加されました。CSS2 は、印刷、画面、音声、携帯情報端末など

のメディアをサポートしています。CSS Mobile Profile はモバイル・デバイスに適した CSS2 のサブセットで、CSS のオーラル・プロパティによって音声レンダリングを制御します。

HTML では、フォームを使用してユーザー・インタフェースを表示し、ユーザーからの入力を受け入れていました。また、HTML フォームでは、データと表現が分離されていませんでした。このため、入力項目の妥当性チェックなどの基本タスクについても、スクリプト・テクノロジを使用する必要がありました。XForms は、HTML フォームのこのような問題に対処するために W3C が制定したテクノロジです。XForms は XML で記述され、XHTML や他のマークアップ言語と統合できます。XForms を使用すると、次のことができます。

- デバイスに依存しないユーザー・インタフェース・コントロールを作成します。
- データとデータ定義を区別します。
- 宣言構文を使用して、一般的に実行されるアクションをサポートします。
- データおよびユーザー・インタフェースのスタイルに関する情報をブラウザに提供します。
- 収集されたデータの有効性を検査します。

XHTML/XFORMS/CSS は強力で標準規格であるため、MobileXML のかわりに使用する必要があります。このリリースでは、XHTML、XForms および CSS の開発に対応して、次のアーキテクチャの変更が実装されています。

- HTTP アダプタは、MobileXML 文書に加えて、XHTML、XForms および CSS の文書も処理します。
- XForms プロセッサが XForms を実装します。
- XForms キャッシュには、実行時に XForms 文書が格納されます。
- XHTML+XForms モデルをサポートするために、CSS 注釈付き XHTML 文書に基づく新しいトランスフォーマ・セットを使用します。このリリースでは、XHTML を格納文書として使用して XForms を使用する場合、その新しい文書モデルを XHTML+XForms モデルと呼びます。
- XForms リクエスト・ブローカは、データまたはイベントを解析して XForms プロセッサにフィードします。データまたはイベントは、OracleAS Wireless に送信するフォーム・データに基づきます。

マルチチャネルの概要

アプリケーションは、通常、コンテンツ（つまり情報）を表現し、それらをユーザーに表示する必要があります。ユーザーは、コンテンツを表示できる複数のモードまたはデバイスからその情報にアクセスできます。アクセス・モードおよびデバイスの特性は、ユーザー・コミュニティが選択します。したがって、作成者は、アプリケーションをすべてのユーザーに対してアクセス可能にすることによって、開発時間を短縮できます。すべてのユーザーに対してアクセス可能とは、様々なアクセス・チャネルでコンテンツが使用可能で、ユーザーは複数の対話モードを使用してアプリケーションと対話できることを意味します。

XHTML+XForms

この項では、OracleAS Wireless の XHTML+XForms+CSS によってサポートされる機能について説明します。この項の内容は次のとおりです。

概要

CSS は文書の表現スタイルを抽象化します。また、XForms は抽象フォーム・モデルを提供し、XHTML は文書の構造およびセマンティクスを提供します。これらのテクノロジには、それぞれ固有の機能があります。これらを組み合せることによって、デバイスに依存しない方法で文書を作成できます。

作成者は、通常、マークアップ言語を使用して文書のコンテンツをエンコードします。一般的に、文書のコンテンツはユーザー・エージェント（ブラウザ）に表示されます。今日の多くのマークアップ言語はコンテンツとスタイルが 1 つの文書内に混在しているため、ユーザー・エージェントでは、表現（またはスタイル）と、文書に含まれているコンテンツ（情報）を区別するのは困難です。HTML3.2 における `<font>` タグの使用方法は、その一般的な例です。HTML 作成者は、通常、色コード（font タグ）を使用して、文書内の重要なセクションをマークします。これは、色をサポートしているビジュアル・メディアを使用して表示する場合には有効ですが、色をサポートしていないデバイス上、またはコンテンツを音声でレンダリングする（テキストが音声インタフェースを介して発声される）場合には役立ちません。

デバイスまたはアクセス・モードに依存しない方法で文書を作成するには、マルチチャネル作成モデルが必要です。マルチチャネル作成モデルでは、文書内のコンテンツ（情報）と表現スタイルを分離する必要があります。

メモリーやプロセッサの制約があるため、モバイル・デバイスでサポートされている表現スタイルは大幅なばらつきがあります。さらに、アクセス方法も異なるため、インタフェースのモードによって表現スタイルが影響を受けます。

テクノロジ開発の背景

XHTML

HTML4 は、World Wide Web（WWW）の公開言語の 1 つとして、広く使用されてきました。HTML4 および CSS によって、コンテンツと表現を正しく分離できます。HTML4 は、モバイル・デバイスではハードウェアの制限や帯域幅の狭さのためにサポートが困難なセマンティクスを豊富にサポートしています。ただし、HTML4 のサブセットと拡張機能は、モバイル・デバイスでサポートできます。

HTML4 を拡張するために、World Wide Web Consortium（W3C）は XHTML（http://www.w3.org/TR/xhtml1/）を公開しました。XHTML は、HTML を XML で再構築した言語です。XML による再構築によって、特定の機能を提供する抽象モジュールを HTML 内で識別できるようになりました。

XHTML をさらに拡張するために、World Wide Web Consortium（W3C）は XHTML モジュール化（http://www.w3.org/TR/xhtml-modularization/）を公開しました。これは、XHTML を、特定タイプの機能を提供する抽象モジュールのコレクションに分解したものです。XHTML モジュール化では、XHTML を定義済モジュールのグループとして定義するフレームワークをサポートします。各モジュールでは、機能および文書構造を定義します。XHTML モジュール化によって、XHTML のモジュールを相互に組み合せて XHTML のサブセットまたはスーパーセットを作成できます。

また、ベンダーは、XHTML モジュール化を使用して、リソースに制約があるデバイスでサポート可能な XHTML のサブセットを定義できます。これにより、モバイル・デバイスで XHTML を使用できるようになりました。XHTML モジュール化によって、重要な仕様である XHTML Basic（W3C）および XHTML Mobile Profile（OMA）が定義されました。これらの仕様によって、モバイル・デバイスで XHTML をサポートできるようになりました。

カスケード・スタイルシート（CSS）

作成者は、カスケード・スタイルシート（CSS）テクノロジを使用して、レンダリング・スタイルと XML 文書の構造を分離できます。CSS では、スタイルシートを使用してマークアップ部分（XML 文書の要素）に表現スタイルを関連付けるため、コンテンツ文書を変更する必要はありません。これによって、文書作成者は、表現スタイルと文書のコンテンツ・モデルが混在しない文書を作成できます。

マルチチャネル用の CSS テクノロジには、@media 規則および CSS Media Queries という 2 つの優れた機能があります。CSS の @media 規則を使用すると、デバイス・メディアに基づいてスタイルを指定できます。また、CSS Media Queries を使用すると、デバイス・メディアでサポートするメディア機能に基づいて表現スタイルを制御できます。

- CSS @media 規則

  作成者は、CSS @media 規則（CSS2）を使用して、メディアに依存するスタイルシートを XHTML 文書に関連付けることができます。これを容易にするため、CSS 仕様には、handheld、aural、tty、print、projection、tv など様々な抽象メディア・タイプが定義されています。たとえば、強調テキストは、デスクトップではイタリックとしてレンダ

リングし、携帯情報端末では下線としてレンダリングできます。オーラル（音声）・インタフェースの場合は、強調テキストを強めに発声することができます。次に例を示します。

```
@media screen {
  em {font-style: italic}
}

@media handheld {
  em {text-decoration: underline}
}

@media aural {
  em {speech-rate: slow; pitch: high}
}
```

@media 規則およびメディア・タイプに関する CSS 構文の詳細は、http://www.w3.org/TR/REC-CSS2 で CSS2 仕様を参照してください。

- CSS3 Media Queries

  CSS @media 規則は強力な機能ですが、デバイスの機能に基づいてコンテンツをレンダリングしたり表示することはできません。たとえば、CSS @media 規則を使用する場合、色をサポートする携帯情報端末と色をサポートしない携帯情報端末とでは、表現スタイルを区別できません。

  CSS3 Media Queries によって、CSS2 に定義されている @media 規則が拡張され、メディア・タイプ（screen、handheld、print、tv など）、および特定のデバイスの機能とプロパティに基づいてスタイルを選択できます。次に例を示します。

```
@media screen, handheld and (color) {
  em {color: red}
}

@media handheld and (color: 0) {
  em {text-decoration: underline}
}
```

  メディア問合せ構文の詳細は、http://www.w3.org/TR/css3-mediaqueries/ で CSS3 Media Queries 仕様を参照してください。

OracleAS Wireless は CSS3 Media Queries をサポートしており、その `media` 属性（MXML 名前空間に定義されます）の構文を使用します。OracleAS Wireless で定義された `media` 属性によって提供される簡略な表記規則を使用して、メディアに基づいて `display:none` プロパティを制御できます。`media` 属性の使用方法の詳細は、8-47 ページの「メディアに応じたコンテンツのスタイルおよび埋込み」を参照してください。

W3C は、リソースに制約がある各種デバイスに適した CSS2 テクノロジのサブセットを定義しています。CSS Mobile Profile および CSS TV Profile は、いずれも CSS2 仕様のサブセット

です。CSS Mobile Profile は、モバイル・デバイスに適した CSS2 テクノロジのサブセットを定義します。OracleAS Wireless は、CSS Mobile Profile をサポートしています。

XForms

XHTML および CSS によって、コンテンツの表現とコンテンツを正しく分離できます。HTML で定義するフォーム・モデルは、実際にはあまり抽象的ではありません。つまり、HTML フォームでは、フォーム・コントロールの表示プロパティとその対象が分離されていません。また、文書の相互作用モデルがないため、作成者は、入力項目の妥当性チェックなどの基本タスクについてもスクリプト・テクノロジを使用する必要がありました。これらの要因によって、デスクトップ環境においても、HTML アプリケーションはブラウザ固有のものでした。

XForms は、HTML フォームのこのような問題に対処するために W3C が制定した新しいテクノロジです。XForms によって優れたフォーム・インタフェースおよび相互作用モデルを定義できるため、スクリプトなどの補助テクノロジは不要になります。XForms の主要な設計目的は次のとおりです。

- 抽象的な UI コントロールをサポートしてデバイスに依存しないこと
- 一般的に実行されるアクションをサポートするための宣言構文
- 対象（データおよび処理ロジック）と表現の分離
- 構造化フォーム・データ（XML を使用）

XForms は、完全な文書モデルを定義するのではなく、フォーム処理に必要なユーザー・インタフェース・コンポーネントのみを定義することを理解してください。XForms は、他の文書コンテンツ情報モデルに追加されたテクノロジであるため、他のマークアップ言語（XHTML など）に含まれる必要があります。XForms は HTML の現行のフォームにかわって使用するものとみなされますが、XForms の設計では、XForms を WML などのユーザー・インタフェース規格（または現行の HTML フォーム）とともに使用することを禁止していません。

XML 名前空間の概要

XML が広く採用されるに従って、様々な業界グループや独立したソフトウェア・ベンダー（ISV）によって定義された XML 文書（DTD）が大量に作成されました。そのため、様々な XML テクノロジ間で要素名や属性名の競合が発生し、ソフトウェア・モジュールで要素を識別できなくなる可能性が出てきました。そこで、XML 文書の要素や属性を一意に識別することが必要になりました。この要件に対処するために、XML 名前空間を使用します。

XML 名前空間は、要素および属性を一意に識別するメカニズムです。XML 名前空間を使用すると、要素名および属性名は URI 参照を使用して識別できます。XML 名前空間を使用した要素名および属性名は、修飾名と呼ばれます。要素および属性の修飾名は、名前空間の接頭辞、コロン（:）、ローカル要素名の順で構成されています。接頭辞は、名前空間の URI の短い表記です。

作成者は、XHTML+XForms 文書を記述するために、次の名前空間を理解する必要があります。

- XHTML の名前空間 : 名前空間 URI: http://www.w3.org/1999/xhtml

  HTML 名前空間は、HTML 仕様に定義されている要素および属性を識別します。このマニュアルでは、HTML 要素の名前空間の接頭辞として html を使用します。例 : `<html:div>`

- XForms 名前空間 : 名前空間 URI: http://www.w3.org/2002/xforms/cr

  XForms 名前空間は、XForms 仕様に定義されている要素および属性を識別します。このマニュアルでは、XForms 要素の名前空間の接頭辞として XForms または xf を使用します。例 : `<xforms:input>` または `<xf:input>`

- XML Events: 名前空間 URI: http://www.w3.org/2001/xml-events

  XML Events 名前空間は、XML Events 仕様に定義されている属性を識別します。このマニュアルでは、XML Events 属性の名前空間の接頭辞として ev を使用します。例 : `<xforms:action ev:event="DOMActivate"/>`

- XML Schema: 名前空間 URI: http://www.w3.org/2001/XMLSchema

  XML Schema 名前空間は、XML Schema 仕様に定義されている要素および属性を識別します。

  注意 : XML Schema は、XML Schema に定義されているすべてのデータ型を識別します。

  このマニュアルでは、XML Schema データ型の名前空間の接頭辞として xsd または xs を使用します。例 : `<xforms:bind type="xsd:positiveInteger"/>`

- MXML: 名前空間 URI: http://xmlns.oracle.com/2002/MobileXML

  OracleAS Wireless は、XHTML+XForms 文書で使用できる拡張要素を定義します。この拡張要素によって、マルチチャネル配信用の XHTML と XForms の機能が拡張されます。このマニュアルでは、MXML 要素の名前空間の接頭辞として mxml を使用します。例 : `<mxml:uiobject ....>`

XPath の概要

XPath は World Wide Web Consortium（W3C）で定義された標準式言語で、XML 文書のフラグメントを処理および操作するための構文を提供します。また、XPath は、`string`、`number` および `boolean` データ型を使用したその他の関数および式をサポートします。

XPath の式および関数 XForms 文書を作成するときに広く使用されている XPath 式の 1 つがパス式です。パス式は、XML 文書内のノード（要素、属性、テキスト、コメント、コンテンツ部分）を識別します。XPath のパス式の構文は、UNIX などのオペレーティング・システムでサポートされているファイル・システムでの、ディレクトリやファイルのナビゲーション構文に似ています。ファイル・システムにおけるディレクトリやファイルのナビゲーションは現行ディレクトリに対して相対的ですが、XPath のパス式は、XML 文書で現在選択されているノード（コンテキスト・ノードと呼びます）に対して相対的です。また、XPath はフルパス式（スラッシュ記号 [/] で始まるパス式）もサポートしています。

次に例を示します。

```
<orders>
   <order>
      <partid item="1">123</partid>
      <quantity>2</quantity>
   </order>
   <order>
      <partid item="2">456</partid>
      <quantity>3</quantity>
   </order>
</orders>
```

XPath 式の `/orders` では、ドキュメント・ルート要素を選択します。

XPath 式の `/orders/order` では、すべての発注要素（ノード集合と呼ばれます）を選択します。

この例では、ノード集合に 2 つの発注要素（ノード）が含まれています。

XPath 式の `/orders/order[position() = 2]` では、2 番目の発注要素を選択します。

XPath 式の `/orders/order/partid/@item` では、すべての `item` 属性（ノード集合）を選択します。

この例では、2 つの item 属性ノードが選択されています。

XPath 式の `/orders/order/partid[@item = 1]` では、すべての partid ノード（この例では属性が 1）を選択します。

また、XPath では、XML ノードで操作できるコア関数ライブラリを定義します。XPath 関数が戻したり受け入れるデータ型は、ノード集合、数値（10 進数）、文字列およびブールです。

XForms の概要

XForms によって、フォーム処理を複数の異なる論理モジュールに区分できます。次のモジュールに区分できます。

- XForms モデル
- XForms 処理ロジック
- XForms のユーザー・インタフェース・コンポーネント

XForms モデル XForms モデルは、フォーム・データの構造、およびフォーム・データに関連付けられたプロパティで構成されます。フォーム・データの構造は、インスタンス文書と呼ばれる XML 文書で表現されます。インスタンス文書内の各データ・ノード（XML 要素）は、インスタンス・データ・ノードと呼ばれます。これは、XPath 仕様で定義されたノードで、XML 文書の要素、属性またはテキストのデータ・ノードがあります。

次の例は、XForms のインスタンス文書を示します。インスタンス文書は XForms の `<instance>` 要素に含まれ、有効な XML 文書であることが必要です。また、インスタンス文書は、単一ルートの文書であることが必要です。つまり、XForms のインスタンス文書の子ノードは 1 つのみで、次の例では、名前空間の接頭辞として my を使用した `http://my.org` 名前空間の `<example>` 要素がこれに該当します。`datanode1` 要素、`datanode2` 要素および `nodeattr` 属性は、インスタンス・データ・ノードまたはインスタンス項目と呼ばれます。

```
<xforms:model xmlns:xforms="http://www.w3.org/2002/xforms/cr">
   <xforms:instance>

      <!-- Data instance -->

      <my:example xmlns:my="http://my.org">
        <my:datanode1 nodeattr="2">120</my:datanode1>
        <my:datanode2>200</my:datanode2>
      </my:example>

      <!-- End Data Instance -->

   </xforms:instance>
</xforms:model>
```

作成者は、XForms を使用して、このインスタンス項目のデータのプロパティを追加定義できます。このプロパティは、モデル項目プロパティと呼ばれます。モデル項目プロパティを使用すると、インスタンス項目の値を制限または制御でき、インスタンス項目を必須にする場合や関連付ける場合を定義できます。インスタンス項目に対して定義できるプロパティは次のとおりです。

- type: type プロパティは、インスタンス項目の基本データ型を制限します。たとえば、インスタンス項目を、文字列、10 進数、整数、日付、時刻などに制限できます。XForms では XML Schema データ型を使用して、インスタンス項目のデータ型を定義します。
- calculate: calculate プロパティを使用すると、他のインスタンス項目に基づいてインスタンス項目の値を計算できます。たとえば、通貨換算アプリケーションでは、換算レートに基づいて通貨金額を換算できます。calculate プロパティの値には、XPath 式を指定します。
- constraint: constraint プロパティは、インスタンス項目に妥当性制約を定義します。妥当性制約は他のインスタンス項目の値に基づいて定義できます。たとえば、ショッピング・カート・アプリケーションでは、ショッピング・カートに複数の品目がある場合のみ、送料無料の規則を有効な選択肢として適用できます。constraint プロパティの値には、XPath 式（ブール式として評価されます）を指定します。
- readonly: readonly プロパティは、読取り専用制約をインスタンス項目に関連付けます。読取り専用制約は、他のインスタンス項目の値に基づいて定義できます。readonly プロパティの値には、XPath 式（ブール式として評価されます）を指定します。
- required: required プロパティは、必須制約をインスタンス項目に関連付けます。必須制約は、他のインスタンス項目の値に基づいて定義できます。required プロパティの値には、XPath 式（ブール式として評価されます）を指定します。
- relevant: relevant プロパティは、コンテキストに基づいてインスタンス項目を関連付けることができます。インスタンス項目の関連付けは、他のインスタンス項目の値に基づいて定義できます。relevant プロパティの値には、XPath 式（ブール式として評価されます）を指定します。

モデル項目プロパティは、XForms の instance 要素内のインスタンス項目またはノードで直接宣言されず、XForms の model 要素内の XForms インスタンス外で宣言されます。モデル項目プロパティは、XForms の `<bind>` 要素を使用して定義し、インスタンス項目に関連付けます。XForms モデルの各 `<bind>` 要素では、インスタンス項目を対応するモデル項目プロパティに関連付けます。`<bind>` 要素には 7 つの属性があります。各モデル項目プロパティ用の属性が 1 つずつあり、モデル項目プロパティに関連付けられたインスタンス項目を識別するバインド式用の属性が 1 つあります。

次の例は、XForms の `<bind>` 要素を示します。この例の `<bind>` 要素では、モデル項目プロパティの type および required をインスタンス項目の `my:datanode2` に関連付け、type プロパティのみ `nodeattr`（`datanode1` の属性）に関連付けます。

注意： `xsd:positiveInterger` は、xsd: が XML Schema データ型の名前空間の接頭辞で、XML Schema データ型が正の整数であることを示します。

```
<xforms:model xmlns:xforms="http://www.w3.org/2002/xforms/cr">
   <xforms:instance>

       <!-- Data Instance document here --->

   </xforms:instance>
   <!-- Bind Elements begin -->
   <xforms:bind id="b1" nodeset="/my:example/my:datanode2"
              type="xsd:positiveInteger"
              required="/my:example/my:datanode1 &gt; 1"/>
   <xforms:bind id="b2" nodeset="/my:example/my:datanode1/@nodeattr"
               type="xsd:positiveInteger" />
   <!--Bind Elements end -->

</xforms:model>
```

注意： XForms の model 要素には、UI コントロール（input、select、textarea など）は含まれません。XForms の model 要素には、フォームで必要なデータ（XML 文書として）、および XML インスタンス文書の各インスタンス項目に対するモデル項目プロパティのみが含まれます。XForms の model 要素で使用される他の要素には、XForms の submission 要素（<submission>）、XML Schema 要素（<xsd:schema>）および XForms アクション要素があります。

XForms 処理ロジック

ユーザーがフォームと対話している間、XForms 文書は中間的な様々な状態を遷移します。XForms 処理ロジックは、文書がいつ、どのような状態を遷移するかを定義します。XForms のすべての処理および状態の遷移は、イベントおよびイベント・ハンドラ（アクション）に関連付けて定義されます。XForms 処理ロジックは、XForms プロセッサがイベントをスローする時期、およびそのイベントに関連付けられたデフォルト動作を指定します。

XForms のイベント・メカニズムは、DOM レベル 2（DOM2）の Events 仕様に準拠しています。DOM2 Events では、ディスパッチされたすべてのイベントに定義済ターゲット・ノード（ドキュメント・ツリーのノード）があります。ディスパッチされたイベントは、イベント獲得フェーズの後に、ドキュメント・ルート・ノードからターゲット・ノードに達します。ターゲット・ノードに達した後、そのイベントは、オプションでバブル・フェーズに進むことができます。さらに、各イベントにはデフォルト・アクションが定義されており、獲得フェーズおよびバブル・フェーズの後に XForms プロセッサによって実行されます。文書の作成者は、必要に応じて、このデフォルト・アクションと取り消すことができます。文書の作成者は、イベント・リスナーを使用して、イベント・ハンドラ（アクションとも呼ばれます）を文書のノードに登録できます。イベント・ハンドラは、イベント獲得フェーズまたはイベント・バブル・フェーズのいずれかで登録して実行できます。DOM2 Events の詳細は、DOM レベル 2 の Events 仕様（http://www.w3.org/TR/DOM-Level-2-Events/）を参照してください。

文書の作成者は、XML Events で定義された宣言構文を使用してイベント・リスナーを登録できます。XML のイベント構文である XML Events は、要素をイベント・オブザーバとして登録可能にする属性、および要素をアクション・ハンドラに設定できる属性を定義します。XML Events の詳細は、XML Events 仕様（http://www.w3.org/TR/xml-events/）を参照してください。

この項で前述したとおり、XForms 処理ロジックは、イベントのデフォルト・セット、および文書の作成者が文書の相互作用モデルを宣言で指定できるアクションを定義します。この宣言モデルによって、スクリプト・テクノロジによるユーザー相互作用のサポートが不要になります。XForms 処理ロジックは、フォーム初期化、フォーム相互作用、プロセッサ通知およびフォーム / プロセッサ・エラーのカテゴリに分類されるイベントを定義します。

次の例は、イベント・ハンドラ（xforms:send アクション）を DOMActivate イベントに登録する方法を示します。この例では、send 要素で ev:observer 属性を使用して、trigger 要素を DOMActivate イベントのオブザーバ（リスナー）として宣言します。DOMActivate が trigger 要素に達すると、プロセッサは trigger 要素が特定のイベントをリスニングしていることを検出します。また、プロセッサは、send 要素が trigger 要素でイベント・ハンドラとして定義されたハンドラであることも検出します。プロセッサは、指定されたイベントに対応して定義済のアクションを実行します。この例では、send（submit）アクションを実行します。

```
<xforms:trigger id="SubmitButton"
xmlns:xforms="http://www.w3.org/2002/xforms/cr">
   <xforms:label>Submit</xforms:label>


   <!-- Event observer is the trigger and event action is the submit -->
   <xforms:send ev:observer="SubmitButton" ev:event="DOMActivate"
               submission="ExpenseSave"/>

</xforms:trigger>
```

注意： XForms モデルおよび XForms 処理ロジックでは、いずれも特定のユーザー・インタフェース・コントロールは宣言されません。この明確な分離によって、XForms モデルおよび処理ロジックは、ホスト言語（WML、HTML フォームなど）を使用して提供されるユーザー・インタフェース・コンポーネントとともに使用できます。

XForms のユーザー・インタフェース・コンポーネント

XForms は、モデルおよび処理ロジックに加えて、ユーザー・インタフェース・コントロールの抽象セットを定義します。ユーザー・インタフェース・コンポーネントは UI ウィジェットの目的を定義し、ウィジェットの表示方法は指定しません。XForms で定義する `select1` はその 1 つの例です。`select1` はこのコントロールの目的を示します。つまり、リストやラジオ・ボタン・コントロールとして項目を表示するのではなく、1 つの項目を選択するために使用することを示します。このユーザー・インタフェース・コントロールは UI ウィジェットの目的のみ定義するため、その表示方法は、ターゲット・デバイスの制約に基づいて決定できます。

XForms のユーザー・インタフェース・コンポーネントには、獲得済または獲得されるデータは格納されません。ユーザー・インタフェース・コンポーネントは、コンポーネントがバインドされているインスタンス・データ・ノード（インスタンス項目）のデータに影響を与えるだけです。ユーザー・インタフェース・コントロールは、コントロールに定義されるバインド属性を使用して、インスタンス項目にバインドします。このコンポーネントで定義されるバインド属性は、`bind`、`model`、`ref` および `nodeset` です。ユーザー・インタフェース・コントロールは、`model` と `ref` 属性、または `nodeset` 属性を使用して、インスタンス項目に直接バインドできます。または、`bind` 属性を使用して、コントロールをインスタンス項目に間接的にバインドできます（`bind` 属性は XForms の bind 要素の id を取り、この id によってインスタンス項目にバインドします）。

次の例は、インスタンス項目にバインドするユーザー・インタフェース・コントロール（`input`）を示します。この例では、`model` 属性と `ref` 属性を使用して、input コントロールをインスタンス項目にバインドします。

```
<xforms:input model="first_model" ref="/my:example/my:datanode1">
    <xforms:label>Input Label</xforms:label>
</xforms:input>
```

次の例も、インスタンス項目にバインドするユーザー・インタフェース・コントロール（`input`）を示します。この例では、`bind` 属性を使用して、input コントロールをインスタンス項目にバインドします。

```
<xforms:input bind="b1">
    <xforms:label>Input Label</xforms:label>
</xforms:input>
```

XForms と XPath

XForms では、すべてのバインド式およびモデル項目プロパティで XPath を使用します。

- モデル・バインド式

  モデル・バインド式は、モデル項目プロパティをインスタンス項目に関連付ける式です。モデル項目式は XPath 式で、XForms の <bind> 要素の nodeset 属性で使用します。

- UI バインド式

  UI バインド式は、UI フォーム・コントロールをインスタンス項目に関連付ける式です。UI バインド式は XPath 式で、UI コントロールの ref または nodeset 属性で使用します。

- 計算式

  計算式は、モデル項目プロパティに値を提供するために、実行時に評価できる式です。計算式は、XForms の <bind> 要素の calculate、constraint、relevant、required および readonly 属性で使用します。

XForms のインスタンス文書は、名前空間によって識別できる場合とできない場合があります。インスタンス文書が名前空間に属していない場合、すべてのバインド式（XPath 式）では、XPath 式に名前空間の接頭辞を使用する必要があります。

次の例では、インスタンス文書が名前空間 http://my.org に属しています。<bind> または UI コントロール内の XPath 式では、パス式に接頭辞として名前空間の接頭辞を追加する必要があります。

```
<xforms:model xmlns:xforms="http://www.w3.org/2002/xforms/cr">
   <xforms:instance>

      <my:example xmlns:my="http://my.org">
         <!-- Child Nodes -->

      </my:example>

   </xforms:instance>

   <!--
     The nodeset attribute use the "my:" prefix to address the

     <example> element in the instance. Also the "my:" prefix must

     be defined in the bind element or in one of the ancestors nodes

     of the bind element
    -->
   <bind nodeset="/my:example" ...>

</xforms:model>
```

XForms のホスト言語としての XHTML

前項で説明したとおり、XForms は、フォーム処理に必要なコンポーネントを提供するだけです。XForms は、完全な文書モデルを定義しません。World Wide Web Consortium（W3C）が作業を進めている次世代の XHTML である XHTML 2.0 では、XForms は XHTML に統合される予定です。現在、XHTML2.0 はワーキング・ドラフトで、XForms がどのように XHTML 2.0 に統合されるかはまだ明確に指定されていません。

OracleAS Wireless は、XForms の格納文書として XHTML を使用する、XForms 処理をサポートします。XHTML+XForms モデルと呼ばれるこの新しい文書モデルでは、XForms の `<model>` 要素は XHTML 文書の `<head>` セクションに属し、フォーム・コントロールおよび Forms のユーザー・インタフェース・コンポーネントは XHTML 文書の `<body>` に属します。OracleAS Wireless は、XHTML Basic およびいくつかの追加モジュールに基づいて XHTML モジュール（要素）をサポートしますが、XHTML Basic の Forms モジュールは XForms に置換されます。

XForms のフォーム・コントロール（input、select、textarea など）は、XHTML インライン・コンテンツとして扱われます。つまり、フォーム・コントロールは、インライン要素（span、strong など）を特定の制限下で使用できる場合に使用できます。コンテンツ・モデルの詳細は、付録 A「サポートされる XHTML モジュール」を参照してください。

XForms のユーザー・インタフェース・コントロール（group、switch/case、repeat など）は、XHTML ブロック・コンテンツ・モデルとして扱われます。つまり、ユーザー・インタフェース・コントロールは、ブロック要素（div など）を特定の制限下で使用できる場合に使用できます。コンテンツ・モデルの詳細は、付録 A「サポートされる XHTML モジュール」を参照してください。

文書のコンテンツ・タイプおよびプロファイル属性の設定

OracleAS Wireless でリモート文書を XHTML+XForms 文書として識別するには、レスポンス文書の MIME メディア・タイプ（content-type）を `application/vnd.oracle.xhtml+xforms` に設定する必要があります。

文書が OracleAS Wireless で定義された XHTML+XForms コンテンツ・モデルに準拠することを OracleAS Wireless Server が判別できるようにするには、準拠するすべての文書で、`<html>` 要素の `profile` パラメータを `http://xmlns.oracle.com/ias/dtds/xhtml+xforms/0.9.0/1.0` に設定する必要があります。

次の例は、正しいコンテンツ・タイプとプロファイル属性を設定した JSP ページを示します。

```
<?xml version = "1.0"?>

<%@ page contentType="application/vnd.oracle.xhtml+xforms"%>

<html xmlns="http://www.w3.org/1999/xhtml"

      xmlns:my="abc"

      xmlns:ev="http://www.w3.org/2001/xml-events"

      xmlns:xforms="http://www.w3.org/2002/xforms/cr"

      profile="http://xmlns.oracle.com/ias/dtds/xhtml+xforms/0.9.0/1.0">
   <head>
      ....
   </head>
   <body>
      ....
   </body>
</html>
```

XHTML と XForms を使用した Hello World アプリケーション

Hello World アプリケーションの概要および基本要件

- Hello World アプリケーションの概要

  Hello World アプリケーションは、input コントロールをユーザーに対して表示します。ユーザーは、input コントロールに値としてテキスト（文字列）データを入力できます。ページを送信すると、Hello World メッセージおよびユーザーが入力した値がレスポンス・ページに表示されます。

- サーバーの要件

  Hello World アプリケーションでは、JSP ページをホスティングして処理できるアプリケーション・サーバーが必要です。

  > **注意：** このチュートリアルでは、JSP を CGI プログラミング環境として使用します。したがって、作成者は、使い慣れている CGI プログラミング環境を使用できます。

アプリケーションまたは文書の作成者は、XML、XHTML および CSS を理解している必要があります。また、このマニュアルのこれまでの説明を読み、XForms および XPath のテクノロジについて基礎知識を得る必要があります。

Hello World アプリケーションの作成

1. XHTML 文書は、<xml> 宣言から始まる必要があります。最初に、次の文字セットを文書に追加します。

   ```
   <?xml version = "1.0"?>
   ```

2. <xml> 宣言の直後に、<html> 要素をドキュメント・ルートとして追加します。<html> 要素の開始タグと終了タグを追加します。デフォルトの名前空間宣言が含まれていることを確認します（<html> 要素の xmlns 属性でデフォルトの名前空間を HTML に設定します）。また、profile 属性を <html> 要素に追加します。

   ```
   <?xml version = "1.0"?>
   <html xmlns="http://www.w3.org/1999/xhtml"
         profile="http://xmlns.oracle.com/ias/dtds/xhtml+xforms/0.9.0/1.0">
</html>
   ```

3. XHTML 文書の <head> セクションと <body> セクションを追加します。

   ```
   <?xml version = "1.0"?>
   <html xmlns="http://www.w3.org/1999/xhtml"
         profile="http://xmlns.oracle.com/ias/dtds/xhtml+xforms/0.9.0/1.0">
     <head>
   ```

```
      <title>My First Hello World Application</title>
    </head>
    <body>
      <h1>My First XForms Page</h1>
      <div>
        <p>Welcome to my first XForms Page.</p>
      </div>
    </body>
</html>
```

4. XHTML 文書にスタイルを追加します （<style> 要素では、type 属性を text/css に設定する必要があります）。

```
  <?xml version = "1.0"?>
  <html xmlns="http://www.w3.org/1999/xhtml"
        profile="http://xmlns.oracle.com/ias/dtds/xhtml+xforms/0.9.0/1.0">
    <head>
      <title>My First Hello World Application</title>
      <style type="text/css">
        body {color : #000000}
        h1   {font-family : sans-serif;
              color : blue}
      </style>
    </head>
    <body>
      <h1>My First XForms Page</h1>
      <div>
        <p>Welcome to my first XForms Page.</p>
      </div>
    </body>
</html>
```

5. XForms をこの XHTML 文書に追加します。最初に、XForms および XML Events の名前空間宣言を <html> 要素に追加します。また、XML データ（インスタンス）文書の名前空間も追加します。この例では、インスタンス・データは http://example.org 名前空間（名前空間の接頭辞として mydata を使用）にあると想定しています。

```
<?xml version = "1.0"?>

<html xmlns="http://www.w3.org/1999/xhtml"

      xmlns:xforms="http://www.w3.org/2002/xforms/cr"
        xmlns:mydata="http://example.org"
        xmlns:ev="http://www.w3.org/2001/xml-events"
      profile="http://xmlns.oracle.com/ias/dtds/xhtml+xforms/0.9.0/1.0">
```

6. XForms の model 要素を head セクションに追加します。<model> には、フォーム・データを定義する <instance> が含まれます。

```
<head>

   <title>My First Hello World Application</title>

   <style type="text/css">
      ....
   </style>
   <xforms:model id="m1">
       <xforms:instance>
           <mydata:example>
              <mydata:name/>
           </mydata:example>
       </xforms:instance>
</xforms:model>
               </head>
```

7. アクセスするユーザーの名前を入力するための UI コントロールを追加します。これを行うには、xforms:input コントロールを <body> のコンテンツに追加します。input コントロールは、model 属性と ref 属性を使用して、xforms:instance に定義されたインスタンス項目（name）にバインドします。

注意： ref 属性では、インスタンス文書（ref="mydata:example/mydata:name"）の name 要素のパス式を含む XPath 式を使用します。

```
  <body>

    <h1>My First XForms Page</h1>

    <div>

     <p>Welcome to my first XForms Page.</p>
     <p>
       <xforms:input model="m1" ref="/mydata:example/mydata:name">
         <xforms:label>Hello Visitor, Please Enter your name</xforms:label>
         <xforms:help>Enter you name</xforms:help>
         <xforms:hint>Please Enter you name</xforms:hint>
       </xforms:input>
     </p>

    </div>
</body>
```

8. 最後に、ユーザーが入力したデータを送信するために、データの送信先である送信ページを定義します。送信をアクティブ化するために、送信トリガーを定義する必要があります。次の例に示すように、送信ページは、XForms の model 要素内で <submission> 要素を使用して定義されます。<submit> 要素は、送信をトリガーする DOMActivate イベントに関連付けられた UI コントロールです。

   XForms の送信は、デフォルトで、インスタンス文書を XML 文書として送信します。送信データを正規の URL パラメータとして受け取るには、method 属性の値を get に設定し、separator 属性の値を &amp に設定する必要があります。次の例は、完全な XHTML+XForms 文書を示します。

```
<?xml version = "1.0"?>

<html xmlns="http://www.w3.org/1999/xhtml"

      xmlns:xforms="http://www.w3.org/2002/xforms/cr"
      xmlns:mydata="http://example.org"
      xmlns:ev="http://www.w3.org/2001/xml-events"
      profile="http://xmlns.oracle.com/ias/dtds/xhtml+xforms/0.9.0/1.0">
  <head>

     <title>My First Hello World Application</title>
    <style type="text/css">
        body {color : #000000}
        h1   {font-family : sans-serif;
              color : blue}
      </style>
      <xforms:model id="m1">
         <mydata:example>
            <mydata:name/>
         </mydata:example>
         <!-- Submission element -->
         <xforms:submission id="nextpage"
              method="get" action="submit.jsp" separator="&amp;amp;"/>
      </xforms:model>

  </head>
  <body>
    <h1>My First XForms Page</h1>

     <div>

      <p>Welcome to my first XForms Page.</p>

      <p>

       <xforms:input model="m1" ref="/mydata:example/mydata:name">
```

```
            <xforms:label>Hello Visitor, Please Enter your name</xforms:label>

            <xforms:help>Enter you name</xforms:help>

            <xforms:hint>Please Enter you name</xforms:hint>

          </xforms:input>

        </p>
        <p>
           To submit please select the submit trigger
           <xforms:trigger>

             <xforms:label>Submit</xforms:label>
             <xforms:send submission="nextpage" ev:event="DOMActivate"/>

           </xforms:trigger>
         </p>

       </div>
    </body>
</html>
```

「Hello World」ページのデプロイおよび CGI プログラムの提供

これで、XHTML+XForms 文書をデプロイしてテストする準備ができました。作成者は、自分の Web サーバーに文書をデプロイできます。また、デプロイ時に、文書の MIME メディア・タイプ（content-type）が application/vnd.oracle.xhtml+xforms に設定されていることを確認します。MIME メディア・タイプは、プログラムによって設定するか、または Web サーバーの構成ファイルを使用して設定できます。また、作成者は、サンプルの送信ページとして使用するページを提供する必要があります。このページで、「Hello World」ページからのフォーム・データを問合せパラメータとして受け取ります。

OracleAS Wireless および XHTML+XForms+CSS

OracleAS Wireless は、デバイスに依存しない文書作成をサポートする次世代の公開言語として XHTML+XForms+CSS をサポートします。この項では、OracleAS Wireless を使用して、特定のチャネル用に XHTML+XForms+CSS を作成する方法を説明します。

OracleAS Wireless は、幅広いデバイスをサポートするために、XHTML+XForms 文書をレンダリングする 3 つのモードをサポートします。次の 3 つのモードです。

- クライアント・コードをサポートしないデバイス

  限定された機能のみサポートし、ユーザー・エージェント動作を拡張できない組込みのユーザー・エージェント（ブラウザ）をサポートするデバイスです。

  このようなデバイスをサポートするために、OracleAS Wireless はサーバー側の仮想ブラウザに似た動作をします。XHTML+XForms のページはサーバー上にキャッシュされ、XHTML+XForms のビューのみデバイスにレンダリングされます。XForms のすべてのイベントはサーバー側で処理されます。デバイスは、更新された XHTML+XForms 文書のコンテンツをレンダリングするだけです。

  ページ処理（XForms プロセッサ）およびレンダリングされたビューをネットワーク上で配信するモデルは、その 1 例です。このモデルの場合、XForms アプリケーションはクライアントの XForms ベース・ブラウザに比べてインタラクティブではありませんが、処理能力が限定されたデバイス上でアプリケーションを適切に制限できます。

  OracleAS Wireless は、すべての電話機（WAP）および PDA（PocketPC、Palm）でこのモデルの処理およびレンダリングをサポートします。

- スクリプト環境をサポートするデバイス

  スクリプト環境をサポートする組込みのユーザー・エージェント（ブラウザ）をサポートするデバイスです。このモデルの場合、XForms プロセッサは、実行時にサーバー側でスクリプトを生成して、処理ロジックをクライアント側にオフロードします。これによって、環境内で相互作用性が向上します。

  このようなデバイスをサポートするために、OracleAS Wireless はサーバー側の仮想ブラウザに似た動作をします。つまり、XHTML+XForms のページはサーバー上にキャッシュされ、XHTML+XForms のビューのみデバイスにレンダリングされます。サーバーでは、クライアントがレンダリングするビューを生成する一方で、（スクリプトを使用して）クライアント上で特定の基本アクションを実行できるスクリプト・コードも生成します。サーバー側の XForms プロセッサでは、ほとんどのイベントおよびアクション（特に、挿入や削除などの重要なアクション）が処理されます。

  ページ処理（XForms プロセッサ）の一部およびレンダリングされたビューをネットワーク上で配信するモデルも、その 1 例です。このモデルでは、クライアント側のスクリプトの相互作用性が向上すること、および完全な XForms 処理モデルはスクリプト環境でサポートできないことを考慮しています。

  OracleAS Wireless は、音声ゲートウェイを使用するオーラル（音声）レンダリング用にこのモデルをサポートします。通常、音声ゲートウェイと OracleAS Wireless Server

は高速ネットワーク上で接続されるため、OracleAS Wireless Server は大量のスクリプト・データを送信できます。

- システム固有のクライアントをサポートできるデバイス

  システム固有コードを実装でき、ブラウザへのプラグインを可能にするデバイスで、ブラウザをレンダリング・コンテナとして使用します。このモデルの場合、XForms プロセッサはクライアント・デバイスにプラグインとして常駐し、サーバーは前処理ステップを実行します。

  このモデルは、相互作用性が最も高く、OracleAS Wireless による XForms ページのサポートも優れています。このモデルでは、ページ処理（XForms 処理）およびビューのレンダリングがシングル・プロセスのコンテキストで発生するため、XForms 指定の機能をより多くサポートできます。

  OracleAS Wireless は、Win32 環境（ラップトップ・デバイス）用の Microsoft Internet Explorer ブラウザのプラグインを提供します。このソフトウェアの今後のリリースでは、さらに多くのデバイス（WinCE、Pocket PC デバイスなど）をサポートする予定です。

OracleAS Wireless は、リクエストが出されたデバイスに基づいて、リクエストの優先モードを動的に選択します。正しいレンダリング・モードを動的に検索することによって、OracleAS Wireless は様々なデバイス上で XHTML+XForms をサポートできます。XForms ページの開発者は、これらの多様なレンダリング・モデルを理解し、あらゆるモードで主要な機能が使用できるように XForms ページを作成する必要があります。

OracleAS Wireless による XHTML、XForms および CSS のサポート

OracleAS Wireless は、XHTML、XForms および CSS のテクノロジを組み合せて、マルチチャネル作成モデルを提供します。また、モバイル・アプリケーションに適した、これらのテクノロジのサブセットもサポートします。

OracleAS Wireless は、いくつかの追加モジュールを含む XHTML Basic をサポートします。XHTML Basic の Forms モジュールは XForms に置換されます。さらに、OracleAS Wireless によって、ナビゲーション・リスト（XHTML2.0）、MXML Media Attribute モジュールおよび Speech Grammar モジュールが追加されます。サポートされる XHTML モジュールの一覧は、付録 A「サポートされる XHTML モジュール」を参照してください。

OracleAS Wireless は、W3C によって定義された CSS Mobile Profile もサポートします。さらに、OracleAS Wireless によって、追加の CSS プロパティとして、CSS オーラル（音声レンダリング用のスタイル）、CSS Media Queries（メディア機能ベースのスタイル）および Oracle CSS レイアウト拡張機能（XForms のスタイル用）が追加されます。OracleAS Wireless はクライアント・デバイス上でブラウザを使用してレンダリングするため、あらゆるデバイスですべてのプロパティがサポートされるわけではありません。サポートできない場合、OracleAS Wireless は適切な表現スタイルを検索します。サポートされる CSS プロパティの一覧は、付録 D「OracleAS Wireless による CSS のサポート」および 8-47 ページの「メディアに応じたコンテンツのスタイルおよび埋込み」を参照してください。

Oracle Application Server Wireless は、W3C XForms 1.0 Candidate Recommendation（http://www.w3.org/TR/2002/CR-xforms-20021112/）で指定されている機能のサブセットもサポートします。サポートされる機能の一覧は、付録 C「XForms 仕様のサポート」を参照してください。

OracleAS Wireless では、サポートされている XHTML モジュール（追加モジュールを含む）と XForms モジュールを組み合せたスキーマを定義します。OracleAS Wireless でレンダリングされてサポートされるすべての XHTML+XForms 文書は、次の要件を満たす必要があります。

- OracleAS Wireless で定義された XHTML+XForms スキーマに準拠すること。
- MIME メディア・タイプ（content-type）を `application/vnd.oracle.xhtml+xforms` に設定して、OracleAS Wireless に渡されること。
- profile 属性（html 要素）を http://xmlns.oracle.com/ias/dtds/xhtml+xforms/0.9.0/1.0 に設定することによって、プロファイルの準拠を示すこと。

OracleAS Wireless と XML Events のサポート

XForms では、DOM2 Events を使用して処理ロジックをサポートし、XML Events 構文を使用して XML 文書に XML Events を表示します。開発者は、XML Events を使用して、イベント・リスナー（またはオブザーバ）を任意の XML 要素（XForms 要素および XHTML 要素）に連結できますが、OracleAS Wireless では、イベントを連結できるノードを制限します。OracleAS Wireless では、イベント・リスナーは XForms のフォーム・コントロール（`input`、`secret`、`textarea`、`trigger`、`select1`、`select`、`submit`）要素にのみ連結できます。また、OracleAS Wireless は、html body 要素および html のナビゲーション・リスト（`html:nl`）要素へのイベント・リスナーの連結もサポートします。OracleAS Wireless は、他の XForms（`xforms:group`、`xforms:item`、`xforms:itemset` など）および HTML 要素（`p`、`div` など）へのイベントの連結はサポートしません。

ビジュアル・アプリケーションと XHTML+XForms

概要 OracleAS Wireless によって、電話機や PDA などの様々なモバイル・デバイスから Web アプリケーションにアクセスできます。Web アプリケーションには、ブラウザ・インタフェースまたはテキスト・ベースのインタフェース（SMS、Instant Messaging インタフェースなど）を使用してもアクセスできます。OracleAS Wireless は、一般的な WML/HDML/HTML ブラウザ、SMS プロトコルおよび一般的な IM プロトコル（Yahoo、AOL、MSN および Jabber）のほとんどをサポートします。次の各項では、デバイス・フォーム要素に加えてサポートされているネットワーク・インタフェースも異なる多様なビジュアル・デバイスで表示されるアプリケーションの作成方法について説明します。

PDA/ ラップトップ・デバイス用のビジュアル・アプリケーション ほとんどの PDA/ ラップトップ・デバイスには HTML ブラウザがありますが、ブラウザでサポートされている HTML のバージョンは大きく異なります。たとえば、ラップトップと PDA デバイスの主な違いは、CSS プロパティのサポートです。CSS をサポートするため、ほとんどの PDA デバイスは HTML 3.2 ブラウザをサポートしていますが、ほとんどのラップトップは HTML 4.0 ブラウザをサポートしています。

OracleAS Wireless は、CSS 仕様の定義に従って、PDA とラップトップを異なるメディア・タイプとして扱います。つまり、ラップトップ・デバイスのメディア・タイプは screen で、PDA のメディア・タイプは handheld として処理します。開発者は、メディア・タイプとメディア・プロパティ（width、height、color など）を使用して、デバイスに応じたページのコンテンツをレイアウトできます。たとえば、screen（ラップトップ）メディアには 3 列レイアウトを使用し、handheld（PDA）メディアには 2 列レイアウトを使用できます。

もう 1 つの重要な違いは、OracleAS Wireless Client のブラウザへのプラグインを使用してフォーム処理とレンダリングをサポートするラップトップ環境があることです。このレンダリング・モードでは、ラップトップ・デバイスに追加のソフトウェアをインストールする必要があります。

ラップトップ・デバイスの場合、OracleAS Wireless は次の 2 つのレンダリング・モードをサポートします。

- ラップトップでブラウザにレンダリングするサーバー側変換。このモードの場合、サーバーは仮想ブラウザと似た動作をし、サーバー上で XForms ページの状態を維持します。サーバーは、ラップトップ上でブラウザを使用して、サーバーから変換された HTML マークアップをレンダリングします。
- ローカル・クライアント・デバイス上で XForms をサポートする Wireless Client のプラグイン。OracleAS Wireless は、このプラグインを使用して、ほとんどの処理をクライアント側に委譲し、サーバー上ではいくつかの前処理ステップを実行します。このプラグインによって、ブラウザはデバイス上で XHTML+XForms ページをサポートできるため、ほとんどのフォーム処理はデバイス上で実行され、OracleAS Wireless サーバーへのラウンドトリップ回数が減ります。OracleAS Wireless Client は、URL で omc:// プロトコル・スキーマを使用して起動できます。次に例を示します。
  omc://wireless-host.com/ptg/rm.

  注意： このプラグインは、Windows プラットフォーム上の Internet Explorer ブラウザでのみ使用できます。

PDA デバイスの場合、OracleAS Wireless はサーバー側ベースの変換のみサポートし、すべてのフォーム処理はサーバー側で実行されます。OracleAS Wireless は HTML 3.2 タグを使用して、PDA デバイス上でスタイル・プロパティを制御します。これは、すべての CSS プロパティ（Mobile Profile）が PDA デバイス上でサポートされているわけではない（特に、ボックス・プロパティや背景プロパティ）ことを示します（サポートされる CSS プロパティの項を参照）。

電話機用のビジュアル・アプリケーション OracleAS Wireless は、HDML、WML、XHTML MP、CHTML ブラウザなど、多様な電話機用ブラウザをサポートします。使用可能な電話機用ブラウザは、次の 2 種類があります。

- 複数のビュー（カード）を単一文書に保持する、従来の WML/HDML ベースのブラウザ。ユーザーはクライアント上のビュー（カード）のスタック間を移動できます（サーバーへのラウンドトリップはありません）。このようなメディアは paged メディアとして処理され、単一文書を複数のページ・レイアウトで表示できます。ユーザーに表示されるのは一度に 1 ページのみです。
- XHTML MP および CHTML ブラウザは、HTML ブラウザと同様で、すべての文書コンテンツが単一のビューに表示されます。

OracleAS Wireless は、すべての電話機をメディア・タイプ handheld として扱います。さらに、複数のカード・ビューをサポートするデバイスは、paged という特別なメディア機能があるとみなされます。paged メディア機能をサポートするデバイスの場合、開発者は CSS の改ページ・プロパティを使用して、文書を複数のカードに分割する方法を制御できます。次の例は、個別のカード内の各 `div` 要素を示します（CSS Media Queries および mxml:media 属性の詳細は、8-47 ページの「メディアに応じたコンテンツのスタイルおよび埋込み」を参照）。

```
<html xmlns="http..." ....>
    ....
   <style type="text/css">
     @media handheld and (paged) {
         body > div {page-break-after: always}
     }
   </style>
   ....
   <body>
     <div>
        Content on Card 1 on paged media
     </div>
     <div>
        Content on Card 2 on paged media
     </div>
   </body>
</html>
```

注意： paged メディア機能は、整数値を取ります。問合せ式 (paged) は (paged:1) と同じです。問合せ式 (paged:0) は、paged メディア機能をサポートしないデバイスに一致します。

すべての電話機および PDA は handheld とみなされるため、開発者は、paged、grid、min/max-device-width などのメディア機能を問合せ式で使用して、文書のコンテンツのスタイルやレイアウトを指定する必要があります。

非同期参照を使用するビジュアル・アプリケーション OracleAS Wireless は、標準的なページング・インタフェース（SMS）、電子メール・インタフェースまたは Instant Messaging インタフェースを使用して、アプリケーションへのアクセスをサポートします。リクエストおよびレスポンスのサイクルが実際には非同期であるため（つまり、リクエストに対するレスポンスがサーバーへの単一の接続では発生しない）、アプリケーションにアクセスするこのモードは特殊とみなされます。アプリケーションとの相互作用の非同期モードをサポートするデバイスは、メディア・プロパティが async であるとみなされます。このプロパティは、メディア問合せ式で使用できます。次の例では、`mxml:media` 属性で async メディア・プロパティを使用します（詳細は 8-47 ページの「メディアに応じたコンテンツのスタイルおよび埋込み」を参照）。

```
<html xmlns="http..." ....>
    ....
   <body>
     <div mxml:media="all and (async)">
       Content to be  displayed on an
       SMS/IM/Email Interface
     </div>
     <div mxml:media="all and (async: 0)">
        Content to be displayed on a regular
        browser interface
     </div>
   </body>
</html>
```

注意： async メディア機能は、整数値を取ります。問合せ式 (async) は (async:1) と同じです。問合せ式 (async:0) は、通常のブラウザ（同期オンライン参照）インタフェースを使用するデバイスに一致します。

非同期チャネルにはクライアント参照機能がないため、アプリケーションの結果表示が他のチャネルと異なる場合があります。表 8-2 に、非同期固有のセマンティクスで使用するタグを示します。これらのタグは、参照デバイスによって解析が異なります。

表 8-2 XHTML/XFORMS のタグ

XHTML/XFORMS のタグ	セマンティクス
xhtml:a	戻されるページにアンカーの値が出力され、ハイパーリンクに識別用の番号接頭辞が追加されます。ターゲット URL と番号接頭辞はサーバーに格納されるため、ユーザーは選択後に URL を取り出すことができます。
xhtml:abbr	テキストを出力します。
xhtml:acronum	テキストを出力します。

表 8-2 XHTML/XFORMS のタグ（続き）

XHTML/XFORMS のタグ	セマンティクス
xhtml:address	改行付きテキストを出力します。
xhtml:blockquote	テキストを出力します。
xhtml:br	改行を出力します。
xhtml:caption	テキストを出力します。
xhtml:cite	テキストを出力します。
xhtml:code	テキストを出力します。
xhtml:dd	テキストを出力します。
xhtml:dfn	テキストを出力します。
xhtml:div	改行付きテキストを出力します。
xhtml:dt	テキストを出力します。
xhtml:em	テキストを出力します。
xhtml:h1	改行付きテキストを出力します。
xhtml:h2	改行付きテキストを出力します。
xhtml:h3	改行付きテキストを出力します。
xhtml:h4	改行付きテキストを出力します。
xhtml:h5	改行付きテキストを出力します。
xhtml:h6	改行付きテキストを出力します。
xhtml:hr	改行を出力します。
xhtml:kdb	テキストを出力します。
xhtml:label	テキストを出力します。
xhtml:li	インデントされたテキストを出力します。番号接頭辞は、親要素が nl の場合、ハイパーリンク・セレクタとしてテキストの前に追加されます。
xhtml:object	テキストを出力します。
xhtml:p	改行付きテキストを出力します。
xhtml:param	無視（非同期対応デバイスには適用されません）。
xhtml:pre	事前にフォーマットされた改行付きテキストを出力します。
xhtml:q	テキストを出力します。
xhtml:samp	テキストを出力します。

表 8-2 XHTML/XFORMS のタグ（続き）

XHTML/XFORMS のタグ	セマンティクス
xhtml:span	テキストを出力します。
xhtml:strong	テキストを出力します。
xhtml:td	各エントリが区切り文字で区切られたテキストを出力します。デフォルトの区切り文字はカンマ（,）です。
xhtml:tr	改行付きテキストを出力します。
xhtml:var	テキストを出力します。
xforms:alert	テキストを出力します。
xforms:filename	無視（非同期対応デバイスには適用されません）。
xforms:help	テキストを出力します。
xforms:hint	無視（非同期対応デバイスには適用されません）。
xforms:input	入力マーカー [] が最後に付いたラベルを出力します。
xforms:item	項目選択を識別する番号接頭辞付きのインデントされた項目ラベルを出力します。
xforms:itemset	番号接頭辞付きのインデントされた項目ラベルを出力します。
xforms:label	テキストを出力します。
xforms:mediatype	無視（非同期対応デバイスには適用されません）。
xforms:message	テキストを出力します。
xforms:output	テキストを出力します。
xforms:range	入力マーカー [] が最後に付いたラベルを出力します。
xforms:secret	入力マーカー [] が最後に付いたラベルを出力します。

表 8-2 XHTML/XFORMS のタグ（続き）

XHTML/XFORMS のタグ	セマンティクス

xforms:select	入力マーカー [..] が最後に付いたラベルを出力します。複数の項目接頭辞をこのフォーム・コントロールに割り当てることによって、デバイスのユーザーは複数の選択肢を選択できます。たとえば、次の文書を考えてみます。
	
	`<xforms:select ref="my:warehouse" selectUI="listbox">`
	`<xforms:label>Select your favorite sports</xforms:label>`
	`<xforms:item>`
	`<xforms:label>Basketball</forms:label>`
	`<xforms:value>basketball</xforms:value>`
	`</xforms:item>`
	`<xforms:item>`
	`<xforms:label>Football</forms:label>`
	`<xforms:value>football</xforms:value>`
	`</xforms:item>`
	`<xforms:label>Basketball</forms:label>`
	`<xforms:value>basketball</xforms:value>`
	`</xforms:item>`
	`<xforms:item>`
	`<xforms:label>Football</forms:label>`
	`<xforms:value>football</xforms:value>`
	`</xforms:item>`
	`</xforms:select>`
	`...`
	
	これは、次のように変換されます。
	
	`Select your favorite sport [...]`
	`1 Basketball`
	`2 Baseball`
	`3 Football`
	
	ユーザーは、Basketball と Baseball を選択する場合、`1 2` を選択してレスポンスします。

xforms:select1	入力マーカー [] が最後に付いたラベルを出力します。

表 8-2 XHTML/XFORMS のタグ（続き）

XHTML/XFORMS のタグ	セマンティクス

xforms:submit	要素は、次のいずれかのオプションとして表示されます。
	■ 文書内の submit 要素と trigger 要素の合計数が 2 未満の場合、要素に対する入力テキストはありません。それ以外の場合は、次のとおりです。
	■ select1 構成メンバーは、最初のフォーム・コントロールとして作成され、文書内の submit 要素と trigger 要素のラベルが項目オプションになります。submit 要素のラベルは、関連する文書のコンテキスト位置に出力されます。
	たとえば、次の文書を考えてみます。
	`...`
	`<xforms:input ref="my:name">`
	`  <xforms:label>Name:</xforms:label>`
	`<x/forms:input>`
	
	`<xforms:submit submission="form1">`
	`  <xforms:label2">Submit</xforms:label>`
	`<x/forms:submit>`
	
	`<xforms:submit Submission="form2">`
	`  <xforms:label>Reset</xforms:label>`
	`<x/forms:submit>`
	`...`
	
	これは、次のように変換されます。
	
	`Actions []`
	`1 #Submit`
	`2 #Reset`
	`Name: []`
	
	`#Submit`
	`#Reset`

xforms:textarea	入力マーカー [] が最後に付いたラベルを出力します。
xforms:trigger	要素は、次のいずれかの方法で表示されます。 ■ 文書内の submit 要素と trigger 要素の合計数が 2 未満の場合、要素に対する出力テキストはありません。それ以外の場合は、次のとおりです。 ■ select1 構成メンバーは、最初のフォーム・コントロールとして作成され、文書内の submit 要素と trigger 要素のラベルが項目オプションになります。submit 要素のラベルは、関連する文書のコンテキスト位置に出力されます。
xforms:upload	無視（非同期デバイスには適用されません）。

音声アプリケーションと XHTML+XForms

OracleAS Wireless は、XForms アプリケーションへの音声アクセスをサポートします。OracleAS Wireless は、XForms ページを VoiceXML および ECMAScript に変換するため、OracleAS Wireless を使用する音声アクセスでは、VoiceXML 1.0 または 2.0、および ECMAScript をサポートする、適合した音声ゲートウェイが必要です。OracleAS Wireless は、XForms ページを VoiceXML に変換します。OracleAS Wireless による変換処理によって、オーラル表現のための VoiceXML（VoiceXML 実行可能コンテンツ）が生成されます。また、音声ゲートウェイで実行する限定された XForms 処理モデルを実装するための ECMAScript 機能も動的に生成されます。

XForms 文書で主に実行する必要があるのは、XForms アクションの処理です。XForms アクションは、そのタイプに応じて、VoiceXML 実行可能コンテンツによって実行されて動的に ECMAScript が生成されるか、または、アクションの処理要件および VoiceXML と ECMAScript の機能に応じてサーバー上で処理されます。たとえば、<message> や <setvalue> などのアクションは、実行可能コンテンツと ECMAScript を使用してクライアント・ブラウザ上で実行されます。これに対して、<insert> や <delete> などのアクションは、リソース集中型の XML/XPath プロセッサが必要なため、サーバー側のサポートを使用します。

次の XForms アクションは、サーバー上で実行されます。

- insert および delete
- setindex
- send（submit）
- reset
- load

これらの XForms アクションを実行するとき、音声ゲートウェイ（クライアント）は、サーバーへのラウンドトリップを開始し、サーバー上でこれらのアクションが実行されます。実行後の XForms ページの表示 / 状態は、音声ゲートウェイ（クライアント）で再レンダリングされます。一部のインスタンスでは、単一イベントの起動によって複数のアクションが実行されます。この場合、サーバーのサポートが必要なアクションがあると、クライアント上ではいずれのアクションも実行されません。かわりに、すべてのアクションがサーバー上で実行されます。たとえば、<insert> アクションの後に <message> アクションが続く場合、両方のアクションはサーバー上で実行されます。つまり、これらのアクションはグループ化されてサーバー側で実行されます。これによって、サーバーは、処理モデルでの一貫性と効率を維持できます。

特定のアクションをゲートウェイに提供するプロセス、および他のアクションをサーバーに提供するプロセスでは、制限があります。XForms 文書では、オーラル・モードでレンダリングされるときに、次の制限があります。

- UI コントロールとアクションの動的バインドは、オーラル・モードではサポートされません。XForms 文書の UI コントロールは、XForms 文書が様々な状態遷移を経過する間、単一のインスタンス項目にバインドされたままであることが必要です。この制限は、ECMAScript に XML プロセッサがないために発生します。

  注意： サーバーのサポートが必要なアクションを常に指定すると、この制限を回避できます。これによって、サーバーは動的バインドを処理できます。

- XML ノード・コンテキストに依存する XPath 関数をオーラル・モードで使用する場合は、音声ゲートウェイ（クライアント）で関数を常に評価できないため、注意が必要です。この制限も、ECMAScript に XML プロセッサがないために発生します。ノード・コンテキストが必要な XPath 関数の一覧は、付録 A「サポートされる XHTML モジュール」を参照してください。このような XPath 関数を使用するアクションは、サーバー側のサポートを必要とするアクションと組み合せることができます。これによって、関数の計算はサーバーで実行されます。

XForms モデル項目プロパティ、デフォルト値および音声レンダリング `relevant`、`readonly`、`default` 値などのモデル項目プロパティは、フォーム・コントロールがオーラル・モードでレンダリングされる方法に影響を与えます。フォーム・コントロールが `relevant="false()"` または `readonly="true()"` にマークされている場合、または有効なデフォルト値がある場合、音声ゲートウェイでは、オーラル・モードでのフォーム・コントロールの値を要求したり収集しません。これ以外の場合、音声ゲートウェイはコントロールの値を要求して収集します（コントロールが `relevant="true()"` または `readonly="false()"` にマークされているが、デフォルト値が無効な場合も含みます）。

音声アプリケーション用の拡張イベント 音声アプリケーションをサポートするために、OracleAS Wireless では、オーラル・モードでアクティブ化できる一連の音声イベントが定義されています。音声アプリケーション用に定義されたイベントは、次のとおりです。

- `vxml-nomatch`

  このイベントは、音声構文認識で、ユーザーの発声が現在の有効範囲内でリスニングしている発声と一致しない場合にディスパッチされます。

  バブル : あり

  取消し可能 : 可

  コンテキスト情報 : なし

  このイベントのデフォルト処理は、ユーザーが使用可能なオプションのリストを読みあげることです。イベントのターゲットが `<nl>`、`<select>` または `<select1>` の場合は、そのターゲットの項目またはオプションが読みあげられます。それ以外の場合のデフォルト処理は、ゲートウェイでユーザーに対して、発声を繰り返すか、または「help」と言うように要求するローカライズされたメッセージを読みあげることです。

- `vxml-cancel`

  このイベントは、ユーザーが「cancel」と言うとディスパッチされます。

  バブル : あり

  取消し可能 : 可

  コンテキスト情報 : なし

  これは通知イベントです。このイベントに対するデフォルト処理はありません。

- `vxml-exit`

  このイベントは、ユーザーが「exit」と言うとディスパッチされます。

  バブルあり

  取消し可能 : 可

  コンテキスト情報 : なし

  このイベントのデフォルト処理は、ローカライズされたバージョンの「Goodbye」を発声し、音声セッションを終了することです。

- `vxml-error`

  このイベントは、音声ゲートウェイでエラーが発生した場合にディスパッチされます。音声ゲートウェイのエラーは、文書内の構文またはセマンティクスのエラーが発生した場合、またはゲートウェイでサポートされていない電話機能を使用しようとした場合に発生します。

  バブル : あり

  取消し可能 : 可

  コンテキスト情報 : なし

  このイベントのデフォルト処理は、エラーが発生したことを示すローカライズされたメッセージを読みあげ、音声セッションを終了することです。

- `vxml-error-badfetch`

  このイベントは、リソースのフェッチ・リクエストが失敗した場合にディスパッチされます。

  バブル : あり

  取消し可能 : 可

  コンテキスト情報 : なし

  このイベントのデフォルト処理は、フェッチ・リクエストが失敗したことを示すローカライズされたメッセージを読みあげ、音声セッションを終了することです。

- vxml-main-menu

  このイベントは、ユーザーが「main menu」と言うとディスパッチされます（このイベントは、サービス固有のメイン・メニュー文書に戻り、サービスでイベントを処理できるようにするために使用します）。

  バブル : あり

  取消し可能 : 可

  コンテキスト情報 : なし

  これは通知イベントです。このイベントに対するデフォルト処理はありません。

音声アプリケーション用の拡張アクション OracleAS Wireless は、MXML 名前空間でカスタム・アクションを定義することによって、XForms アクションのリストを拡張します。

- mxml:handler

  ビジュアル・アプリケーションの場合は、画面に文書が永続的に表示され、ユーザーはその画面で常に情報を参照できます。音声アプリケーションの場合、音声は一過性であるため、ユーザーはシステムで読みあげられた内容を記憶して正しくレスポンスする必要があります。音声アプリケーションはユーザーの記憶に依存するため、ユーザーに対して詳細なヘルプやヒントのメカニズムが必要になる場合があります。ただし、経験豊富なユーザーは、そのような詳細なヘルプやヒントのメカニズムをわずらわしく感じることがあります。広範なユーザーのニーズを満たすために、通常、音声アプリケーションでは、イベントの発生回数に応じたメッセージを提供します。たとえば、ユーザーが「help」を 2 回リクエストした場合、ユーザーはより詳細なヘルプ・メッセージを探しているとみなすことができます。

  イベントの発生回数に応じたアクションをサポートするために、OracleAS Wireless では、MXML 名前空間に定義された <handler> アクションを使用するアクションのリストが拡張されました。<handler> アクションを使用すると、アクションの発生回数に応じて、<catch> 要素の *count* 属性を使用して定義した様々な処理が可能になります。次に、<mxml:handler> アクションを使用する例を示します。この例では、xforms-help イベントの 1 回目と 3 回目の発生で異なるヘルプ・メッセージを提供します。

```
 <xforms:input>
  <mxml:handler ev:event="xforms-help">
     <mxml:catch count="1">
        <xforms:message>Brief Help Message</xforms:message>
     </mxml:catch>
     <mxml:catch count="3">
        <xforms:message>Expanded Help Message</xforms:message>
     </mxml:catch>
  </mxml:handler>
</xforms:input>
```

注意： 前述の例は、ユーザーが 1 回目にヘルプを起動した場合（count="1"）と 3 回目に起動した場合（count="3"）に、異なるヘルプ・メッセージを提供する方法を示します。ユーザーが 4 回目にヘルプを起動した場合は（count="4"）、count="3" に対して定義されたメッセージが使用されます（count="4" に対するメッセージは定義されていないため）。同様に、2 回目にヘルプが起動された場合は（count="2"）、count="1" に定義されたメッセージが使用されます。

- mxml:disconnect

  OracleAS Wireless は、電話システムで機能する音声ゲートウェイを使用して、音声アクセスをサポートします。ユーザーは、音声ゲートウェイにダイヤルインして、音声アプリケーションにアクセスする必要があります。この環境では、作成者が通話を切断してユーザー・セッションを終了できると有効な場合があります。OracleAS Wireless は、これをサポートするために、MXML 名前空間で拡張アクション <mxml:disconnect> を定義します。

音声アプリケーション用の「Help」、「Hint」および「No-Match」メッセージの指定 音声アプリケーションは、通常、ユーザーが「help」と言った場合、ゲートウェイで認識できない何かを言った場合（no-match）、または長時間何も言わなかった場合（no-input）に UI コントロールやナビゲーション・メニューに対して再生されるメッセージを提供します。さらに、ほとんどの音声アプリケーションは、文書（<html:body>）レベルで、デフォルトの「help」、「no-match」および「no-input」のメッセージを宣言します。

XForms の UI コントロールでは、<help> 要素と <hint> 要素を提供します。デフォルトでは、ユーザーが「help」と言うと <help> 要素が再生され、ユーザーが何も言わなかった場合は（no-input）<hint> 要素が再生されます。

開発者は、xforms-help または xforms-hint イベントを処理し、ヘルプやヒントのデフォルト・アクションを実行しない場合、XML Events の *defaultAction* 属性を使用してデフォルト・アクションが実行されないようにします。次の例では、xforms-help イベントのデフォルト・アクションが発生しないようにします。

```
<xforms:input>
  <xforms:label>Input Control</xforms:label>
  <xforms:help>Default Help Message, not played.</xforms:help>
  <xforms:message ev:event="xforms-help" ev:defaultAction="cancel">
     This help message is played.
  </xforms:message>
</xforms:input>
```

音声アプリケーションはユーザーの発声によって駆動しますが、ユーザーの言った内容を常に認識できるわけではありません。ユーザーの発声が認識できない場合は、vxml-nomatch イベントがスローされます。このイベントに対するデフォルトのレスポンスは、状況に応じたオプション（ある場合）のリストを読みあげることです。ただし、作成者は、次の例に示すように、vxml-nomatch に対する独自のハンドラを指定できます。

```
<xforms:select1>
  <xforms:label>Select Control</xforms:label>
  <xforms:message ev:event="vxml-nomatch" ev:defaultAction="cancel">
      Sorry, I didn't understand what you said.
  </xforms:message>
  ...
</xforms:select1>
```

注意： 文書レベル（<html:body>）で、またはナビゲーション・メニュー（<html:nl>）に対して help/hint/no-match のアクションを指定するには、これらのアクションを <xforms:model> セクションで宣言し、XML Events 属性 observer を使用して <nl> 要素と <body> 要素をオブザーバとして宣言する必要があります。これは、XForms 要素を使用するとアクションを子要素として定義できる一方、XHTML では定義できないためです。次のバージョンの XHTML では、この問題は解決される予定です。

次の例では、<body> 要素をオブザーバとして使用し、xforms-help イベントに対するアクションを宣言します。

```
<html>
  <head>
     ....
     <xforms:model>
         <xforms:message ev:event="xforms-help" ev:observer="ID_of_BODY">
             Document level help.
         </xforms:message>
     </xforms:model>
     .....
  </head>
  <body id="ID_of_BODY">
     .....
  </body>
</html>
```

音声構文の埋込み オーラル・モードでは、ページの内容がユーザーに対して読みあげられ、ユーザーからのデータが音声認識またはキー入力によって収集されます。オーラル・モードでのデータ収集およびコマンド選択を可能にするために、開発者は、音声ゲートウェイが音声認識エンジンを使用してリスニングする、一連の発声またはキー入力（キーパッドを使用）の順序を指定する構文を提供できます。

様々な音声認識機能のベンダーによって複数の構文フォーマットが定義されていますが、通常、それらの構文フォーマットは相互に運用できません。後続の表には、ベンダー固有の構文フォーマットがいくつか列挙されています。W3C は、Speech Recognition Grammar Specification と呼ばれる、XML 構文を使用する標準フォーマットを作成しています。

作成者は、<head> 要素に含まれる <grammar> 要素を使用して、XForms 文書に構文を埋め込むことができます。この構文は、html <object> 要素を使用して、UI コントロール、リンク（アンカー）またはナビゲーション・リスト項目（<nl> 内の ）に関連付けることができます（例は、付録 A「サポートされる XHTML モジュール」を参照）。<object> 要素の *type* 属性で、使用する構文のフォーマットを指定します。これによって、作成者は使用可能な構文フォーマットを使用できます。次の表に、OracleAS Wireless でサポートされる構文フォーマット、および対応する *type* 属性の値を示します。

表 8-3 OracleAS Wireless でサポートされる構文フォーマット

フォーマット	Type（MIME タイプ）
GSL	application/x-gsl
ABNF	application/x-abnf
JSGF	application/x-jsgf
W3C Speech Recognition Grammar Specification（SRGS）の XML Form	application/srgs+xml
組込み構文	application/vnd.oracle.builtin+grammar
Oracle Grammar Subset（OGS）	application/vnd.oracle.srgs+xml

組込み構文および Oracle Grammar Subset（OGS）は、サポートされているすべての音声ゲートウェイで開発者が使用できる構文フォーマットです。他のタイプの構文は、異なる音声ゲートウェイ間でトランスポータブルであることは保証されていません（従来型アプリケーションでのみサポートされます）。また、これらの構文は、文書に直接埋め込むことはできず、外部 URL でのみ参照できます。新しい構文を定義する開発者は、最大限の移植性を実現するためには、Oracle Grammar Subset のフォーマットを使用する必要があります。

組込み構文は、音声サービスによって収集された共通タイプのデータ用の構文です。組込み構文を使用するデータ型は、boolean、date、digits（1、1、0 のように 1 つずつ発声された一連の数字を認識します）、currency、number（110 のように単一の数字として発声された一連の数字を認識します）、phone および time です。フォーム・コントロールがいずれかのデータ型で入力されるように指定するには、そのデータ型の組込み音声および DTMF 構文に対応するオブジェクトをフォーム・コントロールの拡張要素内に配置する必要があります。次の例では、digits 型の一連の数字が入力されます。

```
<xforms:input>
  <xforms:label>Digit String</xforms:label>
  <xforms:extension>
    <object type="application/vnd.oracle.builtin+grammar"
data="builtin:grammar/digits"/>
    <object type="application/vnd.oracle.builtin+grammar"
data="builtin:dtmf/digits"/>
  </xforms:extension>
</xforms:input>
```

他のデータ型の場合は、<object> の *data* 属性の digits が他のデータ型の名前にかわります。フォーム・コントロールに関連付けられた構文がない場合、音声ゲートウェイはフォーム・コントロールで休止します。つまり、ユーザーが発声してもフォーム・コントロールに入力されません。

Oracle Grammar Subset（OGS）は、W3C で定義された SRGS（http://www.w3.org/TR/2002/CR-speech-grammar-20020626）の XML Form のサブセットです。サブセットについては、付録 F「Oracle XML Grammar Subset」を参照してください。OGS に定義されている構文は、文書に埋め込んだり、外部 URI からフェッチできます。どのような場合でも、OGS 構文は、指定の発声クラスを認識する音声ゲートウェイ固有のフォーマットに変換されます。OGS 構文によって、音声ゲートウェイ間で移植可能な音声構文および DTMF 構文が提供されます。

構文は、アンカーまたはナビゲーション・リスト項目とともに使用する必要はありません。アンカーまたはナビゲーション・リスト項目とともに使用された構文がない場合、音声インタフェースは要素のコンテンツをリスニングします。ユーザーがコンテンツを言った場合は、アンカーまたはナビゲーション・リスト項目に従います。次の例では、ユーザーが「continue」と言うと音声インタフェースが nextPage.xhtml をフェッチします。

```
<p>
  Say <a href="nextPage.xhtml">continue</a> to go to the next page.
</p>
```

次の例では、音声インタフェースは、ユーザーが「top」と言った場合は top.xhtml をフェッチし、「next」と言った場合は next.xhtml をフェッチします。

```
<nl>
  <label>Navigation Menu</label>
  <li href="top.xhtml">Top</li>
  <li href="next.xhtml">Next</li>
</nl>
```

構文は、<trigger> 要素または <submit> 要素に関連付ける必要はありません。これらの要素に関連付けられた構文がない場合、音声ゲートウェイは <label> のコンテンツをリスニングします。次の例では、「add」と言うと <trigger> がアクティブ化し、「submit」と言うと <submit> がアクティブ化します。

```
<p>
  ...
  <xforms:trigger ...>
    <xforms:label>Add</xforms:label>
    ...
  </xforms:trigger>
  <xforms:submit ...>
    <xforms:label>Submit</xforms:label>
  </xforms:submit>
</p>
```

文書には、複数のフォーム・コントロールとリンクを含めることができます。ユーザー入力を明確に解析するため、アンカー、ナビゲーション・リスト項目、<trigger> および <submit> には、有効範囲が指定されています。これらの要素の有効範囲は、囲んでいる最小のブロック要素（ 、<p>、<nl>、<xforms:group> など）です。次の例では、最初の <p> で「go on」と言うと <trigger> がアクティブ化し、2 番目の <p> で「go on」と言うと <submit> がアクティブ化します。

```
<p>
  ...
  <xforms:trigger>
    <xforms:label>Go on</xforms:label>
  </xforms:trigger>
</p>
<p>
  ...
  <xforms:submit ...>
    <xforms:label>Go on</xforms:label>
  </xforms:submit>
</p>
```

作成者は、文書内のアンカー、<trigger> などの有効範囲に注意する必要があります。つまり、コマンドまたはナビゲーション・オプションは、リスニングされるポイントでのみ使用可能です。

フォーム・コントロールでユーザー入力をリスニングするとき、開発者は、フォーム・コントロールに modal="true" を設定することによって、そのフォーム・コントロールに関連付けられた構文以外は音声ゲートウェイでリスニングしないように設定できます。

音声スタイル用のオーラル CSS の使用 OracleAS Wireless は、発声速度、音量など表現（音声合成）を制御できる CSS のオーラル・プロパティをサポートします。さらに、OracleAS Wireless では、bargein（_orcl-bargein）、say-as フォーマット（_orcl-sayas-format）などの動作をサポートするために、CSS のオーラル・プロパティが拡張されています。サポートされている CSS のオーラル・プロパティおよび拡張オーラル・プロパティの一覧は、付録 D「OracleAS Wireless による CSS のサポート」を参照してください。

UI オブジェクトを使用した VoiceXML サブダイアログの起動 VoiceXML ゲートウェイから XHTML+XForms にアクセスするとき、MXML 名前空間の <uiobject> 要素を使用して、VoiceXML で記述されたサブダイアログに XHTML+XForms からアクセスできます。<uiobject> 要素は、拡張 UI コントロールです。

たとえば、XHTML+XForms 文書の作成者が、VoiceXML の <record> 要素を使用してオーディオを録音し、それをサーバーに送信して格納するとします。VoiceXML 文書で再利用可能なコンポーネントを起動するために、作成者は、読みあげられるいくつかのメッセージを VoiceXML 文書に静的に埋め込むのではなく、それらのメッセージ、およびオーディオ・データの送信先の URL にオーディオ・データを渡します。

また、VoiceXML 文書では、サーバー上でオーディオが格納されているファイルの名前を戻すとします。さらに、コール元が録音を途中で中断した場合に、その中断を示すイベントを VoiceXML 文書でスローし、XHTML+XForms 文書で処理できるようにします。

XHTML+XForms でこのような起動を実行するには、次のマークアップを使用します。

```
<mxml:uiobject data="record.jsp" type="text/x-vxml">
 <f:label>Invoking record function.</f:label>

 <mxml:uiparam
  valuetype="in"
  name="prompt"
  value="'Please record your message.'"
 />

 <mxml:uiparam
  valuetype="in"
  name="noinputMessage"
  value="'I did not hear anything.  Please try again.'"
 />
```

```
  <mxml:uiparam
   valuetype="submit"
   name="storageURL"
   ref="/data/urls/recordingSubmit"
  />

  <mxml:uiparam
   valuetype="out"
   name="location"
   ref="/data/urls/recordingStorage"
  />

  <mxml:uieventmap in="hangup" out="vxml-cancel"/>

  <xforms:action ev:event="vxml-cancel">
   <xforms:setvalue ref="/data/status">Hung up in recording.</xforms:setvalue>
   <xforms:send submission="exit"/>
  </xforms:action>
 </mxml:uiobject>
```

<mxml:uiobject> は、record.jsp でサブダイアログを起動します。*type* 属性とその値は、<mxml:uiobject> が VoiceXML のサブダイアログを起動していることを示すために必要です。<xforms:label> のコンテンツは、サブダイアログの起動直前に読みあげられます。XHTML+XForms 文書は、VoiceXML のサブダイアログが戻るまで一時停止します。

valuetype=*in* を指定した 2 つの <mxml:uiparam> 要素は、VoiceXML サブダイアログで読みあげられるメッセージに渡されます。name=*prompt* を指定した <mxml:uiparam> 要素は、録音開始直前に読みあげられるプロンプトに渡されます。name=*noinputMessage* を指定した <mxml:uiparam> 要素は、録音開始後、一定の時間内にコール元がレスポンスしなかった場合に読みあげられるメッセージに渡されます。

valuetype=*submit* を指定した <mxml:uiparam> 要素は、オーディオの送信先で、インスタンス・データから取得される URL に渡されます。valuetype=*in* と valuetype=*submit* の違いは、前者のタイプの入力は VoiceXML の <var> 変数としてサブダイアログで使用可能になり、後者のタイプは HTTP を介してサブダイアログに送信されることです。

valuetype=*out* を指定した <mxml:uiparam> 要素は、サブダイアログによって戻されたサーバー・ファイル名を受け取り、そのファイル名を参照インスタンス・データ・ノードに格納します。

<mxml:uieventmap> 要素は、VoiceXML イベントから XForms イベントへのマッピングを指定します。コール元が録音を途中で中断した場合、サブダイアログは、hangup という名前の VoiceXML イベントをスローするとします。このイベントは、スローされると、vxml-cancel イベントのディスパッチに変換されます。このイベントはその後、<xforms:action> 要素で処理されます。この要素は、インスタンス・ノードで発生した内容を記録してサーバーに返信します。

次に、VoiceXML を作成してサブダイアログを実装する JSP を示します。*prompt* および *noinputMessage* の入力は VoiceXML の <var> 変数として使用可能になり、同じ名前で <uiparam> で指定された値を使用します。*storageURL* の入力は、HTTP リクエスト・パラメータとして使用可能になります。

```
<?xml version="1.0"?>
<vxml version="1.0">
 <form>
  <var name="prompt"/>
  <var name="noinputMessage"/>

  <record name="audio">
   <prompt>
    <value expr="prompt"/>
   </prompt>

   <noinput>
    <value expr="noinputMessage"/>
   </noinput>

   <catch event="telephone.disconnect.hangup">
    <return event="hangup"/>
   </catch>
  </record>

  <block>
   <submit
    next="<%= request.getParameter("storageURL") %>"
    method="post"
    namelist="audio"
   />
  </block>
 </form>
</vxml>
```

コール元が録音を途中で中断した場合は、サブダイアログで telephone.disconnect.hangup イベントがスローされます。サブダイアログはこのイベントを捕捉し、XHTML+XForms 文書に戻して、VoiceXML の hangup イベントをスローします。それ以外の場合、録音されたオーディオは、XHTML+XForms 文書で指定された URL に送信されます。サーバーは、そのオーディオをファイルに格納し、次のような VoiceXML 文書を戻します。この文書の *location* 変数にファイル名が設定されており、このファイル名は XHTML+XForms 文書に戻されます。

```
<?xml version="1.0"?>
<vxml version="1.0">
 <form>
  <var name="location" expr="'recording529.wav'"/>
  <block>
   <return namelist="location"/>
  </block>
 </form>
</vxml>
```

注意： 前述の最初の VoiceXML 文書がオーディオをサーバーに送信したとき、VoiceXML サブダイアログのセマンティクスによって、XHTML+XForms 文書（特に、その VoiceXML レンダリング）は破棄されません。XHTML+XForms 文書は、2 番目の VoiceXML 文書の送信およびフェッチが実行されている間、実行が一時停止した状況で音声ゲートウェイにロードされたままです。2 番目の文書から制御が戻されると、XHTML+XForms 文書は実行状況に戻ります。

メディアに応じたコンテンツのスタイルおよび埋込み

スタイルは、文書を表現する上で非常に重要です。実際には、ビジュアルなスタイルと音声によるスタイルがあります。たとえば、開発者は、エラー・メッセージを表現するとき、色をサポートしているビジュアル・デバイスでは赤色で表示し、色をサポートしていないビジュアル・デバイスでは太字で表示し、オーラル・デバイスでは大きめの音量でメッセージを再生できます。メッセージ・テキストはすべてのデバイスで同一ですが、ユーザーに対する表現方法が異なり、デバイスの特性に応じた表現方法を使用することに注意してください。このような要件をサポートするために、OracleAS Wireless では、CSS Media Queries をサポートし、これを使用することを推奨しています。

デバイスに応じて実際のコンテンツをカスタマイズする別のタイプのスタイルがあります。たとえば、ビジュアル・デバイスでは、「Enter Amt.」のように短い表記を使用するのが一般的ですが、音声を使用してアプリケーションにアクセスするときは、同じコンテンツでも「Please say the Amount」と表現した方がわかりやすくなります。この場合、アプリケーションのコンテンツの概念は変わりませんが、よりわかりやすく表現するためにコンテンツを少し変更しています。このようなモデルをサポートするために、OracleAS Wireless では、文書のすべての要素でサポートされる拡張属性 *media* が導入されています。

CSS Media Queries

CSS Media Queries は、W3C によって定義された仕様です（http://www.w3.org/TR/css3-mediaqueries/）。開発者は、CSS Media Queries を使用すると、同じコンテンツでも、デバイス（メディア）およびそのデバイスがサポートする機能（メディア機能）に応じてスタイルを指定できます。

> **注意：** このリリースの OracleAS Wireless では、style 属性で `@media` 文をサポートしていません。

次の例では、@media 文を使用して、メディア・タイプ（デバイス）に応じてコンテンツのスタイルを指定します。

```
<html>
  <head>
    <style type="text/css">
      @media handheld, screen, tty {
         .error {color: red}
      }
      @media aural {
         .error {volume: loud}
      }
    </style>
  </head>
  <body>
    <div>
      <span class="error">
```

```
          This is an error message
      </span>
    </div>
  </body>
</html>
```

次に、CSS Media Queries に定義されている問合せ構文を使用する、より高度な @media 文の例を示します。この例では、エラー・メッセージを表現するとき、色をサポートするデバイスでは赤色で表示し、色をサポートしないデバイスでは下線付きテキストとして表示し、音声モードでは大きめの音量で再生します。

```
<html xmlns="http://www.w3.org/1999/xhtml"
      profile="http://xmlns.oracle.com/ias/dtds/xhtml+xforms/0.9.0/1.0">
  <head>
    <style type="text/css">
      @media handheld and (color), screen and (color), tty and (color) {
         .error {color: red}
      }
      @media handheld and (monochrome), screen and (monochrome), tty and
(monochrome) {
         .error {text-decoration: underline}
      }
      @media aural {
         .error {volume: loud}
      }
    </style>
  </head>
  <body>
    <div>
      <span class="error">
          This is an error message
      </span>
    </div>
  </body>
</html>
```

MXML メディア属性

OracleAS Wireless では、XHTML+XForms 文書のすべての要素でサポートされている拡張属性 media が導入されています。この media 属性は OracleAS Wireless 名前空間（MXML）に定義され、media には、接頭辞として名前空間の接頭辞（mxml:media）を追加する必要があります。

mxml:media は、サポートされているメディア（デバイス）およびメディア機能（デバイス機能）に応じた、コンテンツ・モデルの条件付き表示をサポートします。mxml:media 属性は、HTML 名前空間での style 属性と同様に、レンダリングにのみ影響し、XForms のイベントやアクションなどの処理ロジックには影響しません。mxml:media は、mxml:media の値空間にリストされていないメディアに CSS プロパティ 'style="display: none"' を指定するショートカットとみなすことができます。

> **注意：** mxml:media によって、要素が文書から削除されることはありません。action 要素に mxml:media を指定しても影響ありません。mxml:media は、要素のレンダリングの制御にのみ使用されます。また、関連するアクション・ハンドラを持つイベント・オブザーバである要素に mxml:media を指定しても、イベントは取得され、関連するアクション・ハンドラが実行されます。

MXML メディア属性構文 開発者は、mxml:media を使用する場合、特定のコンテンツがターゲットになるメディア（デバイスおよびデバイス機能）を指定する必要があります。OracleAS Wireless は、mxml:media 属性をサポートするために、CSS3 Media Queries 構文の使用をサポートします。前述したとおり、メディア問合せ構文では、メディアとメディア機能を組み合せることができる、問合せに似た式をサポートします。

次に、メディア・タイプのみ使用して、レンダリングされたコンテンツを制御する簡単な例を示します。この例では、オーラル・デバイスは文をレンダリングしますが、ビジュアル・デバイスは短い文字列をレンダリングします。

```
<div>
  <p mxml:media="handheld, screen, tty">
    Currency Conv. Tbl.
  </p>
  <p mxml:media="aural">
    Here is the currency conversion table
  </p>
</div>
```

次に、メディア機能（デバイスのフォーム要素など）を使用して、レンダリングされたコンテンツを制御するより高度な例を示します。この例では、デバイスの幅が 20em（m 文字または m 文字と同じ幅で 20 文字分を表示可能）以上の場合に「Currency Conversion Table」と表示され、これより狭い場合は短い文字列で「Currency Conv. Tbl」と表示されます。

```
<div>
   <p mxml:media="screen, handheld and (min-device-width: 20em), tty and
(min-device-width: 15em)">
      Currency Conversion Table
   </p>
   <p mxml:media="screen, handheld and (max-device-width: 15em), tty and
(max-device-width: 15em)">
      Currency Conv. Tbl.
   </p>
</div>
```

サポートされるメディアおよびメディア機能の一覧は、付録 B「メディア・タイプとその機能」を参照してください。

XHTML と XForms を使用した高度なサンプル

この項では、高度な例の作成方法を説明し、XForms の様々な側面（モデル、制約、イベントなど）について説明します。また、作成した文書のスタイルを CSS プロパティを使用して指定する方法も説明します。

この項で使用する例の概要

この項で使用する例では、ショッピング・カートに入っている品目の名前、価格および数量をユーザーに表示し、ユーザーはショッピング・カート内の品目の数量を変更できます。ショッピング・カートには、品目の（更新後の）最終小計（品目価格×数量）、および（更新後の）最終合計金額が表示されます。

文書およびコンテンツ・タイプの構造 最初に、XHTML+XForms 文書の構造を作成します。すべての XHTML 文書には、<html>、<head> および <body> の各セクションと適切な属性が必要です。<html> 要素では、適切な名前空間の定義と接頭辞を宣言する必要があります。

```
<?xml version = "1.0"?>

<html xmlns="http://www.w3.org/1999/xhtml"

      xmlns:html="http://www.w3.org/1999/xhtml"
      xmlns:xf="http://www.w3.org/2002/xforms/cr"

      xmlns:ev="http://www.w3.org/2001/xml-events"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:mydata="http://example.org"
```

```
      profile="http://xmlns.oracle.com/ias/dtds/xhtml+xforms/0.9.0/1.0">
    <head>
      <title>Shopping Care Example</title>
    </head>
    <body>
    </body>
  </html>
```

この例では、次の名前空間の定義と接頭辞を宣言しています。

- 接頭辞 html を使用した HTML 名前空間
- 接頭辞 xf を使用した XForms 名前空間
- 接頭辞 ev を使用した XML Events 名前空間
- 接頭辞 mydata を使用した、ショッピング・カート内のデータの名前空間定義

head 要素には、すべての文書で設定する必要がある *profile* 属性が含まれています。

文書を作成した後、正しいコンテンツ・タイプを設定したことを確認することが重要です。これによって、OracleAS Wireless は文書を XHTML+XForms 文書として認識できます。コンテンツ・タイプは、application/vnd.oracle.xhtml+xforms に設定する必要があります。JSP を使用してこの例を作成している場合は、`<@page>` 宣言または `response.setContentType()` メソッドを使用してコンテンツ・タイプを設定できます。

ショッピング・カートのデータと XForms モデル

この例では、主にショッピング・カートのデータを使用します。ショッピング・カートには、複数の品目を入れることができます。各品目について、次の 3 つのデータがあります。

- 品目名
- 品目価格
- 数量

このデータは、XML フォーマットでモデル化できます。次のデータ例では、3 つの品目がショッピング・カートに追加されています。

```
<cart xmlns="http://example.org">
  <item>
    <name>A Book</name>
    <price>10</price>
    <quantity>2</quantity>
    <subtotal/>
  </item>
  <item>
    <name>A Game</name>
    <price>15</price>
    <quantity>3</quantity>
    <subtotal/>
  </item>
  <item>
    <name>A Movie</name>
    <price>20</price>
    <quantity>4</quantity>
    <subtotal/>
  </item>
</cart>
```

この XML データは、xhtml:head セクション内の xforms:model 要素のコンポーネントである instance セクションで、XForms 文書に追加できます。

ショッピング・カートのデータを <head> セクションに入れる手順は、次のとおりです。

1. XForms モデルを <head> セクション（この例では xf: model 要素）で宣言します。
2. インスタンス・データを XForms の model セクション（この例では xf:instance）で宣言します。

3. ショッピング・カートの XML データを、XForms の instance セクションに配置します。

```
<head>

    <title>Shopping Care Example</title>
    <!-- Declare an XForms Model -->
    <xf:model id="model_1">
       <!-- Declare an XForms Instance -->
        <xf:instance>
           <!-- Embed the above XML Data Document -->
           <cart xmlns="http://example.org">
              .....
           </cart>
        </xf:instance>
    </xf:model>
</head>
```

ユーザーに対するデータの表示

文書の基本構造とショッピング・カートのデータは準備できました。データをユーザーに表示するには、ショッピング・カートの最初の行（品目）をユーザーに表示します。

ショッピング・カートの最初の行（品目）を表示する手順は、次のとおりです。

1. 品目の名前、価格および数量を表示できる 3 つのフィールドを作成します。
2. この例では、修飾子 name または price は使用しません。ユーザーは、quantity のみ変更できます。XForms の Output コントロールを使用して名前と価格を表示し、XForms の Input コントロールを使用して数量を表示します。
3. これら 3 つのフィールドを作成した後、各フィールドの値が head セクションの XML データ（インスタンス）から取得されるようにします。これを行うには、XPath を使用して UI コントロールと XML データの間のマッピング（バインド）を作成します。このマッピングは UI バインド式と呼ばれ、UI コントロールの *ref* 属性を使用して実行されます。

   たとえば、*name* 要素をショッピング・カートの最初の行にバインドするには、次の XPath 式を使用します（mydata: は、ショッピング・カートの XML データに対する名前空間の接頭辞です）。

   ```
   ref="/mydata:cart/mydata:item[1]/my:data:name"
   ```

   XML データの price と quantity の値を UI コントロールとバインドする場合も同様で、次の XPath 式を使用する必要があります。

   ```
   ref="/mydata:cart/mydata:item[1]/mydata:price"
   ref="/mydata:cart/mydata:item[1]/mydata:quantity"
   ```

次に、XForms の UI コントロールとそのマッピング（バインド）XPath 式を指定した <body> セクションを示します。

```
<body>
  <div>
     <!-- Display "Item name"
     <xf:output ref="/mydata:cart/mydata:item[1]/mydata:name">
       <xf:label>Item Name</xf:label>
     </xf:output>

     <xf:output ref="/mydata:cart/mydata:item[1]/mydata:price">
       <xf:label>Item Price</xf:label>
     </xf:output>

     <xf:input ref="/mydata:cart/mydata:item[1]/mydata:quantity">
      <xf:label>Quantity</xf:label>
     </xf:input>

  </div>
</body>
```

注意： この文書を使用するには、<body> セクションと <head> セクションをマージし、文書のコンテンツ・タイプが Web サーバー上で正しく設定されていることを確認してください。

この文書を OracleAS Wireless Server で使用すると、すべての UI コントロールは雑然と表示され、整った表示ではありません。スタイルを追加する方法は、後の項で説明します。

繰返し構造の追加

この時点では、ショッピング・カートの最初の行が表示されています。ショッピング・カートの他の行を表示するには、データの 2 番目と 3 番目の行に静的にマップされる UI コントロールを追加する方法があります。4 番目、5 番目の品目を追加する場合、XForms では repeat 構成メンバーを使用します。repeat を使用すると、インスタンス・データに基づいて構造を繰り返すことができます。

ショッピング・カートの例に繰返し構造を追加する手順は、次のとおりです。

1. *nodeset* 属性を指定して <repeat> 要素を定義します。*nodeset* 属性は、インスタンス（XML）データセットのコレクションを選択する XPath 式です。

   ショッピング・カートの例では、nodeset="/mydata:cart/mydata:item" によって、インスタンス（XML）データからすべての品目（行）が選択されます。

2. また、repeat 要素内に、繰返しによって選択される各品目の詳細を選択および表示する UI コントロールを追加します。

   注意： UI コントロールの *ref* 属性で指定する XPath 式には、繰返し（nodeset 属性）によって確立されるコンテキストに対して相対的なパス式が必要です。

次の例は、repeat 要素を使用した <body> セクションで、ショッピング・カートのすべての品目を表示します。

```
<body>
  <xf:repeat nodeset="/mydata:cart/mydata:item">
     <!-- Display "Item name" -->
     <xf:output ref="mydata:name">
       <xf:label>Item Name </xf:label>
     </xf:output>

     <!-- Display "Item Price" -->
     <xf:output ref="mydata:price">
       <xf:label>Item Price </xf:label>
     </xf:output>

     <!-- Display "Item Quantity" -->
     <xf:input ref="mydata:quantity" size="5">
       <xf:label>Quantity </xf:label>
     </xf:input>

  </xf:repeat>
</body>
```

計算フィールドの追加 : 小計および合計

この時点では、ショッピング・カートのインスタンス（XML）データ内のすべての行が表示されています。次に、小計および合計用のフィールドを追加します。小計および合計は、ユーザーが選択した品目の価格と数量に基づいて計算されます。また、ユーザーがショッピング・カート内の品目の数量を変更すると、合計も自動的に更新される必要があります。このため、アプリケーションでは、データが送信された後にバックエンドで計算するのではなく、Forms プロセッサを使用してデータの変更と同時に値を計算するようにします。

インスタンス・データの各 <item/> ノードには、<subtotal/> 子ノードがあります。このノードを使用して、特定の品目の小計を格納します。<subtotal/> ノードでは最初のデータの値はありませんが、XForms プロセッサは、品目の価格と数量に基づいて小計の値を計算し、ノードに入力します。このため、文書では、XForms プロセッサに対して小計の計算方法を指示する必要があります。

XForms は、条件のインスタンス・データへの添付をサポートします。この条件は、モデル項目プロパティと呼ばれます。

> **注意 :** モデル項目プロパティはデータベース表によく似ており、列名を宣言し、NOT NULL などの追加条件や外部キー制約を追加することもできます。XForms プロセッサでサポートされるモデル項目プロパティ（条件）は、type（データ型）、relevant、calculate、readonly、required および constraint です。

モデル項目プロパティ（バインド条件）は、XHTML+XForms 文書の model セクションで <xf:bind> 要素を使用して添付できます。XForms の bind 要素では、XPath 式を使用して条件を関連付けるインスタンス・ノードを識別する、*nodeset* 属性をサポートします。小計については、*calculate* 属性を使用して計算規則も添付する必要があります。

次の例では、XForms の bind 要素で、計算用のモデル項目プロパティ（条件）を subtotal インスタンス項目（データ・ノード）に追加します。

```
<xf:bind nodeset="/mydata:cart/mydata:item/mydata:subtotal"

         calculate="../mydata:price * ../mydata:quantity"/>
```

この bind 要素は、XHTML+XForms 文書の XForms の model セクション内で指定する必要があります。

```
<head>

    <title>Shopping Care Example</title>
    <!-- Declare an XForms Model -->
    <xf:model id="model_1">
       <!-- Declare an XForms Instance -->
        <xf:instance>
           <!-- Embed the above XML Data Document -->
```

```
            <cart xmlns="http://example.org">
               .....
            </cart>
         </xf:instance>

         <!-- Bind Conditions -->
         <xf:bind nodeset="/mydata:cart/mydata:item/mydata:subtotal"
                  calculate="../mydata:price * ../mydata:quantity"/>
    </xf:model>
</head>
```

次に、output コントロールを使用して、小計をユーザーに表示します。

```
<xf:repeat ...>
  <!-- Item Name -->
  ....
  <!-- Item Price -->
  ....
  <!-- Item Quantity -->
  ....
  <!-- Sub Total -->
  <xf:output ref="mydata:subtotal">
     <xf:label>Sub Total </xf:label>
  </xf:output>
</xf:repeat>
```

次に、すべての品目の合計を表示します。合計を表示するために、各品目の小計を合計する集計関数 sum を使用します。また、合計を表示するために、*value* 属性を指定した output コントロールも使用します （この output コントロールは、繰返し構造の外側で指定します)。

```
<xf:repeat ...>
  <!-- Item Name -->
  ....
  <!-- Item Price -->
  ....
  <!-- Item Quantity -->
  ....
  <!-- Sub Total -->
</xf:repeat>
<div>
  <xf:output value="sum(/mydata:cart/mydata:item/mydata:subtotal)">
    <xf:label>Total </xf:label>
  </xf:output>
</div>
```

スタイルの追加

繰返しが表構造（通常、ショッピング・カート・アプリケーションで使用されます）で表示されるように、簡単なスタイルを追加します。このために、次の例に示すように、<style> 要素を <head> セクションに追加します。

```
<head>
  <style type="text/css">
     /*... Style Declarations */
     /*
       Display repeat as tabular
       layout with 4 columns,
       but show the labels of UI
       Control only once
     */
     repeat {_orcl-repeat-labels: once;
             display: grid;
             _orcl-grid-cells: 4}

     /*
       Display input and output
       within a repeat as a cell in
       in the tabular layout
     */
     repeat > input, repeat > output
         {display: grid-cell}

     /*
       Display input and output
       labels as a cells in the
       tabular structure
     */
     repeat > input > label,
     repeat > output > label
          {display: grid-cell;
           _orcl-label-side: top}

     /*
       Display all labels as
       bold and underlined
     */
     label {text-decoration: underline;
            font-weight: bold}
  </style>
</head>
```

更新ボタンの追加とイベントの使用

ショッピング・カートの例では、データ、UI コントロールおよびスタイルを追加しました。アプリケーションでは、ユーザーがショッピング・カートの内容を変更したときに、小計と合計の画面表示を更新するメカニズムを提供することも重要です。使用しやすいデバイス（システム固有のクライアント・スタック、またはスクリプトがサポートされたデバイス）では、画面（またはディスプレイ）の更新はクライアント側で自動的に実行されます。画面の動的な更新をサポートしていないユーザー・エージェント（ブラウザ）の場合は、アプリケーションで、ユーザーが画面（またはディスプレイ）の更新を手動でリクエストできるボタンを提供する必要があります。ボタン（つまりトリガー）によって OracleAS Wireless の中間層へのラウンドトリップが開始され、更新された値を使用してページが再レンダリングされます。

XForms にボタンを追加するには、XForms の trigger コントロールを使用する必要があります。アクションが trigger コントロールに関連付けられるまで、trigger コントロール自体は何も実行しません。また、アクションは、そのアクションを実行できるイベントに関連付けられる必要があります。次に、XForms の trigger コントロールの例を示します。この例では、リフレッシュ・アクションが trigger コントロールに関連付けられ、trigger コントロールがアクティブ化（DOMActivate イベント）するとリフレッシュ・アクションが実行されます。

```
<div>
   <xf:trigger>
      <xf:label>Update</xf:label>
      <xf:refresh model="model_1" ev:event="DOMActivate"/>
    </xf:trigger>
</div>
```

タイプの妥当性チェックの追加

タイプの妥当性チェックを追加して、サンプル・アプリケーションを完成させます。ショッピング・カートの「Quantity」（数量）フィールドの値は数字ですが、ユーザーが負の数字や文字データを入力しても妨げられません。アプリケーション・エラーを回避するために、ユーザーが「Quantity」フィールドに無効な値を入力した場合はユーザーに警告します。「Quantity」フィールドの妥当性チェックは、XForms の bind 要素で実行されます。この要素では、「Quantity」フィールドのスキーマ・データ型を宣言できます。この例では、XML Schema データ型 nonNegativeInteger が使用されます。

```
<xforms:model>
   <xf:bind nodeset="/mydata:cart/mydata:item/mydata:quantity"
type="xsd:nonNegativeInteger"/>
</xforms:model>
Also we have to provide an alert message that is displayed when user enters an
invalid value.
<xf:input ref="mydata:quantity" size="5">
   <xf:label>Quantity </xf:label>
   <xf:alert>Quantity must be greater than or equal to zero</xf:alert>
</xf:input>
```

完全なサンプル・コード

ショッピング・カートの例では、XForms のデータ・モデル、XForms の UI コントロールとインタフェース、イベントとアクション、モデル項目プロパティとスタイルなど、XForms アプリケーションを作成する際の様々な局面を説明しました。次に、ショッピング・カートの例の完全な XForms 文書を示します。

```
<?xml version = "1.0"?>
<html xmlns="http://www.w3.org/1999/xhtml"
       xmlns:html="http://www.w3.org/1999/xhtml"
       xmlns:xf="http://www.w3.org/2002/xforms/cr"
       xmlns:ev="http://www.w3.org/2001/xml-events"
       xmlns:mydata="http://example.org"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       profile="http://xmlns.oracle.com/ias/dtds/xhtml+xforms/0.9.0/1.0">
    <head>
       <title>Shopping Care Example</title>
       <!--
       <style type="text/css">
          repeat {_orcl-repeat-labels: once;
                  _orcl-grid-cells: 4;
                  display: grid}
          input, output {display: grid-cell}
          input > label, output > label
                     {display: grid-cell;
                      _orcl-label-side: top;
                     text-decoration: underline;
                     font-weight: bold}
       </style>
       -->
      <style type="text/css">
        /*... Style Declarations */
        /*
           Display repeat as tabular
           layout with 4 columns,
           but show the labels of UI
           Control only once
        */
        repeat {_orcl-repeat-labels: once;
               display: grid;
               _orcl-grid-cells: 4;
               border-left-style: solid;
               border-left-width: 1px;
               border-top-style: solid;
               border-top-width: 1px;
               border-right-style: solid;
               border-right-width: 1px;
               border-bottom-style: solid;
```

```
            border-bottom-width: 1px}

      /*
         Display input and output
         within a repeat as a cell in
         in the tabular layout
      */
      repeat > input, repeat > output
          {display: grid-cell}

      /*
        Display input and output
        labels as a cells in the
        tabular structure
      */
      repeat > input > label,
      repeat > output > label
          {display: grid-cell;
           _orcl-label-side: top}

      /*
        Display all labels as
        bold and underlined
      */
       label {text-decoration: underline;
              font-weight: bold}
</style>


   <xf:model id="model_1">
      <!-- Declare an XForms Instance -->
      <xf:instance>
        <!-- Embed the above XML Data Document -->
        <cart xmlns="http://example.org">
          <item>
            <name>A Book</name>
            <price>10</price>
            <quantity>2</quantity>
            <subtotal/>
          </item>
          <item>
            <name>A Game</name>
            <price>15</price>
            <quantity>3</quantity>
            <subtotal/>
          </item>
          <item>
```

```
                <name>A Movie</name>
                <price>20</price>
                <quantity>4</quantity>
                <subtotal/>
              </item>
            </cart>
          </xf:instance>
          <xf:bind nodeset="/mydata:cart/mydata:item/mydata:subtotal"
                  calculate="../mydata:price * ../mydata:quantity"/>
          <xf:bind nodeset="/mydata:cart/mydata:item/mydata:quantity"
type="xsd:nonNegativeInteger"/>
       </xf:model>
    </head>
    <body>
      <xf:repeat  nodeset="/mydata:cart/mydata:item">
         <!-- Display "Item name"-->
         <xf:output ref="mydata:name">
           <xf:label>Item Name </xf:label>
         </xf:output>

         <xf:output ref="mydata:price">
           <xf:label>Item Price </xf:label>
         </xf:output>

         <xf:input ref="mydata:quantity" size="4">
          <xf:label>Quantity </xf:label>
          <xf:alert>Quantity must be greater than or equal to zero</xf:alert>
         </xf:input>

         <xf:output ref="mydata:subtotal">
           <xf:label>Sub Total </xf:label>
         </xf:output>

      </xf:repeat>
      <div>
         <xf:output value="sum(/mydata:cart/mydata:item/mydata:subtotal)">
           <xf:label>Total </xf:label>
         </xf:output>
      </div>
      <div>
         <xf:trigger>
            <xf:label>Update</xf:label>
            <xf:refresh model="model_1" ev:event="DOMActivate"/>
          </xf:trigger>
      </div>
    </body>
  </html>
```

XHTML と XForms を使用した高度な音声サンプル

この項では、XHTML+XForms を使用して音声サービスを開発する高度な例を説明します。前項の高度なサンプルに関する詳細は繰り返し説明しませんが、音声に関連するサービスに重点を置いて説明します。

ここでは、請求書のチップを計算し、合計請求金額を人数で割るサービスを例にして説明します。このサービスのインスタンス・データは、次の項目で構成されます。

- 請求書の税抜き金額
- 請求金額を分割する人数
- チップのパーセント

これは、次のインスタンス・データ構造を使用して、XML で表現されます。

```
<tip>
 <amt/>
 <num/>
 <pct/>
</tip>
```

最初は、amt、num および pct の各ノードにデータは格納されていません。このサービスの最初のステップは、これらのデータを収集することです。請求書の税抜き金額は、次のフォーム・コントロールで収集されます。

```
 <xforms:input id="amt" ref="/tip/amt">
  <xforms:label>
   <object data="/audio/howMuch.wav" type="audio/wav">
    How much is the bill?
   </object>
  </xforms:label>

  <xforms:extension>
   <object
    data="builtin:grammar/currency"
    type="application/vnd.oracle.builtin+grammar"
   />

   <object
    data="builtin:dtmf/currency"
    type="application/vnd.oracle.builtin+grammar"
   />
  </xforms:extension>

  <xforms:help>
   Help.  Say the amount of the bill in dollars and cents.
   For example, twenty-five dollars and ten cents.
  </xforms:help>
```

```
 <mxml:handler ev:defaultAction="cancel" ev:event="xforms-hint">
  <mxml:catch count="1">
   <xforms:message>How much is the bill?</xforms:message>
  </mxml:catch>

  <mxml:catch count="5">
   <xforms:load resource="main.jsp"/>
  </mxml:catch>
 </mxml:handler>

 <mxml:handler ev:defaultAction="cancel" ev:event="vxml-nomatch">
  <mxml:catch count="1">
   <xforms:message>
    Please say that again.  How much is the bill?
   </xforms:message>
  </mxml:catch>

  <mxml:catch count="2">
   <xforms:message>
    Say the amount of the bill in dollars and cents.  For
    example, twenty-five dollars and ten cents.
   </xforms:message>
  </mxml:catch>

  <mxml:catch count="5">
   <xforms:load resource="main.jsp"/>
  </mxml:catch>
 </mxml:handler>
</xforms:input>
```

注意： <label> コンテンツでは、プロンプトとして音声の <object> を使用します。audio/wav MIME タイプをサポートしない音声ゲートウェイで文書が解析される場合、または音声ゲートウェイによる音声ファイルのフェッチが失敗した場合、含まれているテキストは TTS エンジンで発声されます。

<xforms:extension> 要素に含まれる <object> は、通貨指定を認識する組込みの VoiceXML 音声と DTMF 構文を参照します。このような組込み構文を使用するには、data 属性で VoiceXML の built-in: URI スキームを使用し、type 属性で組込み構文に対する Oracle 固有の MIME タイプを使用する必要があります。音声ゲートウェイは、このフォーム・コントロールに達すると、コール元が金額を発声するか、または金額がキー入力されるのをリスニングします。戻される値は、通貨の種類を示す 3 文字のコードが先頭に付いた金額です。この例では、通貨の種類を示すコードを USD（米国ドル）とします。したがって、コール元が

「Twenty-three dollars and fifty-eight cents」とレスポンスすると、値「USD23.58」が amt インスタンス・ノードに入れられます。

<xforms:input> の子である help と event handler では、コール元がヘルプを要求した場合、コール元がプロンプトに対してレスポンスしない場合、またはコール元のレスポンスを音声ゲートウェイで認識できない場合に発生するメッセージおよびアクションを定義します。コール元が「help」と言うと、音声ゲートウェイは <xforms:help> 要素のコンテンツを読みあげます。

コール元がプロンプトに対して一定時間レスポンスしないと、xforms-hint イベントがディスパッチされます。このイベントは、拡張アクションの <mxml:handler> と <mxml:catch> とともに処理されます。ゲートウェイでユーザー入力を待ってタイムアウトになった場合、1 ～ 4 回目のタイムアウトでは、count="1" が指定された <catch> のコンテンツが実行され、メッセージが読みあげられます。ゲートウェイで 5 回目のタイムアウトが発生すると、サービスはメイン・メニュー文書に戻ります。

コール元が言った内容または一連のキー入力が金額として認識されない場合は、vxml-nomatch イベントがディスパッチされ、最後の <mxml:handler> 要素で処理されます。入力内容が認識されない場合、1 回目は最初のメッセージが読みあげられます。2 ～ 4 回目は 2 番目のメッセージが読みあげられます。5 回目に入力内容が認識されない場合、サービスはメイン・メニューに戻ります。

2 つある <mxml:handler> 要素には両方とも ev:defaultAction="cancel" 属性が設定され、xforms-hint イベントおよび vxml-nomatch イベント（一般的なメッセージが再生されます）に対するデフォルト・レスポンスが取り消されます。

次のステップでは、請求金額を分割する人数を収集します。これは、次のフォーム・コントロールを使用して実行します。

```
<xforms:input ref="/tip/num">
 <xforms:label>
  <object data="/audio/howMany.wav" type="audio/wav">
   How many are in your party?
  </object>
 </xforms:label>

 <xforms:extension>
  <object
   data="builtin:grammar/number"
   type="application/vnd.oracle.builtin+grammar"
  />

  <object
   data="builtin:dtmf/number"
   type="application/vnd.oracle.builtin+grammar"
  />
 </xforms:extension>
</xforms:input>
```

このフォーム・コントロールは、発声またはキー入力された数字をリスニングし、num インスタンス・ノードに入力してその数字を読みあげます （実際のサービスでは、ヘルプ、レスポンスなし、および認識できないレスポンスのハンドラが提供されます)。

次に、チップのパーセントを収集します。これは、次のフォーム・コントロールを使用して実行されます。

```
<xforms:select ref="/tip/pct">
 <xforms:label>
  <object data="/audio/howBig.wav" type="audio/wav">
   How big a tip would you like to leave?
  </object>
 </xforms:label>

 <xforms:item>
  <xforms:label>small</xforms:label>
  <xforms:value>10</xforms:value>
 </xforms:item>

 <xforms:item>
  <xforms:label>medium</xforms:label>
  <xforms:value>15</xforms:value>
 </xforms:item>

 <xforms:item>
  <xforms:label>large</xforms:label>
  <xforms:value>20</xforms:value>
 </xforms:item>

 <xforms:help>
  Help.  Say small for a ten percent tip, medium for a
  fifteen percent tip, and large for a twenty percent tip.
 </xforms:help>
</xforms:select>
```

このフォーム・コントロールは、リスニングする内容を指定するために、構文の <object> を含む <xforms:extension> 要素は使用しません。音声ゲートウェイは、<xforms:item> 要素の子であるいずれかの <xforms:label> のコンテンツをリスニングします。<xforms:label> のコンテンツが読みあげられると、対応する <xforms:value> のコンテンツが pct インスタンス・ノードに入れられます。<xforms:input> 要素は、ユーザーが「help」と言ったときに読みあげられる指示を指定します。

入力を収集した後、サービスは、チップ、チップを含む支払合計金額、および 1 人当たりの支払金額を計算し、その結果を読みあげます。計算および出力は、次のマークアップで実行されます。

```
On a bill of <xforms:output style="speak: currency" ref="/tip/amt"/>, a
<xforms:output style="speak-numeral: continuous" ref="/tip/pct"/>
percent tip is
<xforms:output
 style="speak: currency"
 value="concat('USD',round(/tip/pct * substring(/tip/amt,4)) div 100)"
/>, for a total of
<xforms:output
 style="speak: currency"
 value="concat('USD',round((100 * substring(/tip/amt,4)) +
        (/tip/pct * substring(/tip/amt,4))) div 100)"
/>.  If divided evenly by
<xforms:output style="speak-numeral: continuous" ref="/tip/num"/>,
each person would owe
<xforms:output
 style="speak: currency"
 value="concat('USD',round(((100 * substring(/tip/amt,4)) +
        (/tip/pct * substring(/tip/amt,4))) div /tip/num) div 100)"
/>.
```

計算は、<xforms:output> フォーム・コントロールの value 属性で XPath 式を使用して実行されます。amt インスタンス・ノードの値を使用する場合は、substring 関数を使用して最初の「USD」通貨インジケータを削除します。concat 関数を使用して、「USD」接頭辞を計算済金額に追加します。

「USD23.58」のような通貨金額を TTS エンジンで読みあげるとき、その値が通貨金額であることをエンジンに指示しないと、エンジンでは「You ess dee twenty-three point fifty-eight」のように発音します。前述のマークアップ内の style="speak: currency" 属性によって、出力する値は通貨金額として読みあげるように TTS エンジンに指示されます。また、style="speak-numeral: continuous" 属性によって、チップを分割する人数が一連の数字ではなく 1 つの数字として読みあげられます（たとえば、15 は「one five」ではなく「fifteen」と発音されます）。

最後に、チップ計算がレポートされた後、ユーザーはメイン・メニューに戻るか、または別のチップを計算できます。これは、次のマークアップを使用して実行されます。

```
To compute another tip, say another tip.
To return to the main menu, say <a href="main.jsp">main menu</a>.
<xforms:trigger>
 <xforms:label>Another tip</xforms:label>
 <xforms:action ev:event="DOMActivate">
  <xforms:setvalue ref="/tip/amt"/>
  <xforms:setvalue ref="/tip/num"/>
  <xforms:setvalue ref="/tip/pct"/>
  <xforms:setfocus control="amt"/>
 </xforms:action>
</xforms:trigger>
```

TTS エンジンでは、アンカーのコンテンツを含めてテキストが読みあげられ、<xforms:trigger> のラベルは読みあげられません。アンカーのコンテンツ、および <xforms:trigger> の <xforms:label> は、音声ゲートウェイでリスニングされます。アンカーのコンテンツが読みあげられると、アンカーの href 属性がフェッチされます。<xforms:label> のコンテンツが読みあげられると <xforms:trigger> がアクティブ化して、以前に入力されたデータは消去され、最初のフォーム・コントロールにフォーカスが設定されます。インスタンス・ノードを消去した場合は、新しいデータを入力する必要があります。有効なコンテンツを持つフォーム・コントロールにオーラル・モードでアクセスする場合、フォーム・コントロールはスキップされます。

次に、完全で高度な音声サンプルを示します。 要素は、アンカーの有効範囲、および文書の最後の <xforms:trigger> を制御します。最初の では、アンカーとトリガーが有効範囲内にないため、そのコンテンツはリスニングされません。2 番目の では、両方とも有効範囲内です。

```
<?xml version="1.0"?>
<html
 xmlns="http://www.w3.org/1999/xhtml"
 xmlns:xforms="http://www.w3.org/2002/xforms/cr"
 xmlns:ev="http://www.w3.org/2001/xml-events"
 xmlns:mxml="http://xmlns.oracle.com/2002/MobileXML"
 profile="http://xmlns.oracle.com/ias/dtds/xhtml+xforms/0.9.0/1.0"
>
 <head>
  <xforms:model>
   <xforms:instance>
    <tip>
     <amt/>
     <num/>
     <pct/>
    </tip>
   </xforms:instance>
  </xforms:model>
 </head>

 <body>
  <div>
   <xforms:input id="amt" ref="/tip/amt">
    <xforms:label>
     <object data="/audio/howMuch.wav" type="audio/wav">
      How much is the bill?
     </object>
    </xforms:label>

    <xforms:extension>
     <object
      data="builtin:grammar/currency"
```

```
   type="application/vnd.oracle.builtin+grammar"
  />

  <object
   data="builtin:dtmf/currency"
   type="application/vnd.oracle.builtin+grammar"
  />
 </xforms:extension>

 <xforms:help>
  Help.  Say the amount of the bill in dollars and cents.
  For example, twenty-five dollars and ten cents.
 </xforms:help>

 <mxml:handler ev:defaultAction="cancel" ev:event="xforms-hint">
  <mxml:catch count="1">
   <xforms:message>How much is the bill?</xforms:message>
  </mxml:catch>

  <mxml:catch count="5">
   <xforms:load resource="main.jsp"/>
  </mxml:catch>
 </mxml:handler>

 <mxml:handler ev:defaultAction="cancel" ev:event="vxml-nomatch">
  <mxml:catch count="1">
   <xforms:message>
    Please say that again.  How much is the bill?
   </xforms:message>
  </mxml:catch>

  <mxml:catch count="2">
   <xforms:message>
    Say the amount of the bill in dollars and cents.  For
    example, twenty-five dollars and ten cents.
   </xforms:message>
  </mxml:catch>

  <mxml:catch count="5">
   <xforms:load resource="main.jsp"/>
  </mxml:catch>
 </mxml:handler>
</xforms:input>

<xforms:input ref="/tip/num">
 <xforms:label>
  <object data="/audio/howMany.wav" type="audio/wav">
```

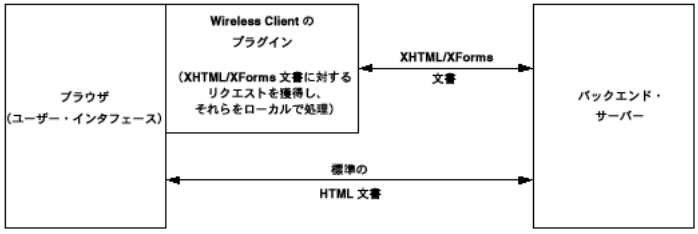
```
   How many are in your party?
  </object>
 </xforms:label>

 <xforms:extension>
  <object
   data="builtin:grammar/number"
   type="application/vnd.oracle.builtin+grammar"
  />

  <object
   data="builtin:dtmf/number"
   type="application/vnd.oracle.builtin+grammar"
  />
 </xforms:extension>
</xforms:input>

<xforms:select ref="/tip/pct">
 <xforms:label>
  <object data="/audio/howBig.wav" type="audio/wav">
   How big a tip would you like to leave?
  </object>
 </xforms:label>

 <xforms:item>
  <xforms:label>small</xforms:label>
  <xforms:value>10</xforms:value>
 </xforms:item>

 <xforms:item>
  <xforms:label>medium</xforms:label>
  <xforms:value>15</xforms:value>
 </xforms:item>

 <xforms:item>
  <xforms:label>large</xforms:label>
  <xforms:value>20</xforms:value>
 </xforms:item>

 <xforms:help>
  Help.  Say small for a ten percent tip, medium for a
  fifteen percent tip, and large for a twenty percent tip.
 </xforms:help>
</xforms:select>

On a bill of <xforms:output style="speak: currency" ref="/tip/amt"/>, a
<xforms:output style="speak-numeral: continuous" ref="/tip/pct"/>
```

```
   percent tip is
   <xforms:output
    style="speak: currency"
    value="concat('USD',round(/tip/pct * substring(/tip/amt,4)) div 100)"
   />, for a total of
   <xforms:output
    style="speak: currency"
    value="concat('USD',round((100 * substring(/tip/amt,4)) +
           (/tip/pct * substring(/tip/amt,4))) div 100)"
   />.  If divided evenly by
   <xforms:output style="speak-numeral: continuous" ref="/tip/num"/>,
   each person would owe
   <xforms:output
    style="speak: currency"
    value="concat('USD',round(((100 * substring(/tip/amt,4)) +
           (/tip/pct * substring(/tip/amt,4))) div /tip/num) div 100)"
   />.
  </div>

  <div>

   To compute another tip, say new tip.
   To return to the main menu, say <a href="main.jsp">main menu</a>.
   <xforms:trigger>
    <xforms:label>New tip</xforms:label>
    <xforms:action ev:event="DOMActivate">
     <xforms:setvalue ref="/tip/amt"/>
     <xforms:setvalue ref="/tip/num"/>
     <xforms:setvalue ref="/tip/pct"/>
     <xforms:setfocus control="amt"/>
    </xforms:action>
   </xforms:trigger>
  </div>
 </body>
</html>
```

OracleAS Wireless Client

OracleAS Wireless Client は Web ブラウザへのクライアント側プラグインで、XHTML/XForms のクライアント側処理およびレンダリングをサポートするように Web ブラウザを拡張します。

図 8-1 Wireless Client のアーキテクチャ



レンダリングおよび処理はすべてブラウザの一部として実行されるため、サーバーへのラウンドトリップは、ユーザーが別の文書を送信またはダウンロードした場合のみ必要です。これによって、必要なすべての XHTML/XForms 処理をサーバーで実行する必要がなく、帯域幅も最小限で済み、低速または信頼性の低いネットワーク上でも迅速に作業を実行できます。

OracleAS Wireless Client のプラグインは、DHTML ブラウザの内部 DOM 構造を操作して必要なユーザー・インタフェースを生成し、すべてのイベントを直接獲得するため、レスポンスは迅速で豊富な機能を利用できます。

Wireless Client の使用

Client は、インストール後、すぐに使用できます。Internet Explorer を起動して、ロケーション・バーに XClient の有効な URI を入力します。XClient の URI はすべて omc プロトコルで始まります。次に、有効な URI の例を示します。

```
omc://chalk.us.oracle.com/testexpense/expenser.xad
omc://chalk.us.oracle.com/testexpense/expense.xforms
```

ユーザー相互作用

XHTML/XForms 文書をロードした後は、プラグインがユーザーからすべてのイベントを引き継いで獲得します。関連するアクションが実行されると、イベントはローカルの XForms プロセッサにフィードバックされて、ローカルに処理されます。

ロギング

XClient のログ・ファイルには、自分の XForms アプリケーションをデバッグするときに役立つ情報が含まれています。このログ・ファイルは HTML ファイルで、デフォルトでは XClient アプリケーション・フォルダの「ログ」ディレクトリに格納されています。記録する情報の数を制御できます。logLevel は、0（ロギングなし）～9（最大ロギング）に設定できます。Offline Manager を使用して、ログ・レベルを設定します。

サーバー側の考慮事項

XClient は、サーバー・マシンから XForms 文書などの文書を取り出します。サーバーでは、各文書に適切なコンテンツ・タイプと有効期限を設定する必要があります。

OracleAS Wireless と XClient の使用

OracleAS Wireless を中間層として使用する場合、XClient は追加機能を使用します。

MIME タイプ

サーバーでは、XForms 文書のコンテンツ・タイプを application/vnd.oracle.xhtml+xforms に設定する必要があります。サーバーでコンテンツ・タイプが適切に設定されないと、XClient で正しく処理されません。

OracleAS Wireless では、インストール時にコンテンツ・タイプを事前定義するため、開梱後すぐに使用できます。

OracleAS Wireless Client のインストール

要件

インストールする前に、デスクトップ・マシンに次のコンポーネントがインストールされていることを確認してください。

- Internet Explorer（6.0 以上）
- Java Development Kit（1.3.x または 1.4.x）

デフォルト・ブラウザが Internet Explorer に設定されていることを確認してください。不明な場合は、次の手順でブラウザをデフォルト・ブラウザに設定できます。

1. Internet Explorer を起動します。
2. メニューから「ツール」→「インターネット オプション」を選択します。
3. 「インターネット オプション」ダイアログで「プログラム」タブを選択します。
4. ダイアログ下部にある「Internet Explorer の起動時に、通常使用するブラウザを確認する」をチェックします。次に Internet Explorer を起動するときに、デフォルト・ブラウザかどうかを確認するメッセージが表示されます。

Wireless Client のインストール

WDK から Wireless Client をインストールするには、次のディレクトリに移動します。

$WDK_HOME/wclient

HTML ファイル install.html を開きます。このファイルには、必要なコンポーネントがインストールされているかどうかをチェックする JavaScript が含まれています。インストールされていない場合は、必要なファイルが追加され、プラグインがインストールされます。

インストールが開始されると、情報が表示されたダイアログに続いて、アプリケーション・フォルダ名を要求するダイアログが表示されます。独自のフォルダ名を入力するか、またはデフォルトのフォルダ名を使用できます。次のダイアログを閉じると、実際のインストールが開始されます。アプリケーション・ファイルがアプリケーション・フォルダにコピーされ、いくつかの環境変数が追加または変更されます。

この時点で、インストール・ページには、インストールが正常終了したと表示されます。

注意：

- OracleAS Wireless Client の使用を開始する前に、再起動が必要になる場合があります。
- ブラウザで、アクティブなプラグインの使用を制限するセキュリティ設定が設定されている場合があります。Wireless Client をインストールする前に、ブラウザを起動し、メニュー項目の「ツール」→「インターネット オプション」を選択します。「セキュリティ」タブを選択して、**「レベルのカスタマイズ」**をクリックします。「ActiveX コントロールとプラグイン」グループにある次のオプションを「有効にする」に設定します。1)「署名済み ActiveX コントロールのダウンロード」、2)「ActiveX コントロールとプラグインの実行」、3)「スクリプトを実行しても安全だとマークされている ActiveX コントロールのスクリプトの実行」。

ユーザーへのデプロイ

エンド・ユーザーへの OracleAS Wireless Client のデプロイは、他のブラウザのプラグインをデプロイするのと同様に簡単です。最初に、OracleAS Wireless Client CAB ファイルを Web サーバー上のアクセス可能な位置にコピーし、ユーザーをその URL にポイントします。Wireless Client CAB ファイルを取得するには、マシンに WDK をインストールしてから CAB ファイルをコピーする必要があります。詳細は、8-73 ページの「OracleAS Wireless Client のインストール」を参照してください。

ユーザーに OracleAS Wireless Client をデプロイするシームレスな方法として、CAB ファイルの URL を XHTML/XForms 文書内の `<OBJECT>` タグに埋め込む方法があります。これによって、コンポーネントがダウンロードされていない場合は、ブラウザで自動的にダウンロードします。次のタグを使用して、アプリケーションのいずれか（またはすべて）の HTML 文書にこのファイルの URL を含めることができます。

```
<OBJECT
classid=CLSID:098f2470-bae0-11cd-b579-08002b30bfeb
id=WClient
codebase="xclient.CAB" >
</OBJECT>
```

このタグを文書の本体に含めると、ブラウザでは、プラグインがインストールされているかどうかを自動的にチェックします。この場合、必要なアクションはありません。ユーザーがOracleAS Wireless Client をインストールしていない場合、ブラウザはユーザーに対してインストールするように要求します。ブラウザは、CAB ファイルをダウンロードし、その中から必要なファイルを抽出して、OracleAS Wireless Client をインストールします。このデプロイ・ファイルの詳細は、8-75 ページの「XClient.CAB ファイル」を参照してください。

classid は、検索対象の COM オブジェクトをブラウザに示します。これは、Wireless Client のメイン・インタフェースの CLSID に設定する必要があります（098f2470-bae0-11cd-b579-08002b30bfeb）。

XClient.CAB ファイル

XCLIENT.CAB は、インストール可能な圧縮ファイルで、OracleAS Wireless Client のプラグイン用のすべてのバイナリ、およびプラグインをブラウザで使用できるように構成するための指示（COM コンポーネントの登録、必要なレジストリ・キーの設定など）が含まれます。

レジストリ・キー

OracleAS Wireless Client は、レジストリを使用して、構成パラメータおよびランタイム・パラメータを格納します。すべてのキーは、次の場所に格納されます。

HKEY_LOCAL_MACHINE/Software/Oracle/XForms

XHTML Mobile Profile

XHMTL Mobile Profile（XHTML MP）は、Open Mobile Alliance（OMA、以前の Wap Forum）によって定義された規格で、準拠するすべてのモバイル・ブラウザでサポートされます。XHTML MP は、W3C によって定義された XHTML 1.1 のサブセットで、W3C で定義された XHTML Basic に基づいています。この項では、XHTML MP をアプリケーション作成言語として使用するときに、OracleAS Wireless によってサポートされる使用方法および機能について説明します。この項の内容は次のとおりです。

概要

W3C で定義された XHTML1.1 仕様（HTML4.1 に基づく）は、モバイル・デバイスや埋込みデバイスでサポートするのが困難です。W3C は、最小限の HTML モジュール（HTML 要素）を使用して、すべてのデバイス（モバイルおよび埋込み）でサポート可能な XHTML Basic を定義しました。XHTML Basic にさらにモジュール（HTML 要素）を追加したのが XHTML MP で、モバイル・デバイスでサポートできます。

XHTML MP は、HTML 仕様に定義されている Forms モジュールをサポートします（拡張コントロールは含みません）。HTML Forms は本質的に UI 指向で、相互作用動作や処理ロジックは定義しません。HTML Forms でこのようなセマンティクスが欠如しているため、アプリケーションを音声インタフェースなどのチャネルで使用できません。OracleAS Wireless は、XHTML MP をビジュアル・モバイル・ブラウザ環境でのアプリケーション作成言語としてのみサポートし、音声、SMS、Instant Messaging インタフェースなどのアクセス・チャネル / モードはサポートしません。

OracleAS Wireless および XHTML MP + CSS Mobile Profile

OracleAS Wireless は、XHTML MP と CSS Mobile Profile を組み合せて、あらゆるビジュアル・メディア表現用のマルチチャネル作成モデルを提供します。

OracleAS Wireless は、XHTML Mobile Profile（付録 A「サポートされる XHTML モジュール」を参照）およびいくつかの追加モジュールをサポートします。さらに、OracleAS Wireless は、特別なモジュール（XHTML2.0 のナビゲーション・リストおよび MXML Media Attribute モジュール）もサポートします。サポートされる XHTML MP モジュールの一覧は、8-78 ページの「サポートされる XHTML Mobile Profile モジュール」を参照してください。

OracleAS Wireless は、W3C で定義された CSS Mobile Profile をサポートします。さらに、OracleAS Wireless は、CSS3 モジュールである CSS Media Queries（メディア機能ベースのスタイル）もサポートします。OracleAS Wireless はクライアント・デバイス上でブラウザを

使用してレンダリングするため、あらゆるデバイスですべてのプロパティがサポートされるわけではありません。サポートできない場合、OracleAS Wireless は適切な表現スタイルを検索します。サポートされる CSS プロパティの一覧は、付録 D「OracleAS Wireless による CSS のサポート」および付録 C「XForms 仕様のサポート」を参照してください。

> **注意：** Open Mobile Alliance は、W3C で定義された CSS Mobile Profile に密接にマップされる、CSS のサブセットも定義しています。OMA の CSS サブセットでは追加の拡張プロパティが定義されていますが、OracleAS Wireless はその追加プロパティをサポートしていません。

OMA（WAP Forum）の XHTML MP 仕様に定義されているとおり、OracleAS Wireless でレンダリングされ、サポートされるすべての XHTML MP 文書は、次の要件を満たす必要があります。

- OMA で定義された XHTML MP DTD に準拠すること。
- XHTML MP 文書は、MIME メディア・タイプ（content-type）を application/vnd.wap.xhtml+xml に設定して、OracleAS Wireless に渡されること。
- XHTML MP 文書に、次の DOCTYPE 宣言があること。

```
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
    "http://www.wapforum.org/DTD/xhtml-mobile10.dtd">
```

サポートされる XHTML Mobile Profile モジュール

OracleAS Wireless は、XHTML MP の次のモジュールをサポートします。

表 8-4 サポートされる XHTML MP モジュール

モジュール	説明
Structure モジュール	要素 : html、head、title、body
Text モジュール	要素 : abbr、acronym、address、blockquote、br、cite、code、dfn、div、em、h1、h2、h3、h4、h5、h6、kbd、p、pre、q、samp、span、strong、var
Hypertext モジュール	要素 : a
List モジュール	要素 : dl、dt、dd、ol、ul、li 拡張要素 : nl、label（詳細は、A-4 ページの「List モジュール」を参照）
Basic Forms および Partial Full Forms モジュール	要素 : form、input、label、select、option、textarea、fieldset、optgroup（注意 : フォーム・コントロールのラベルには、すべて label 要素を使用することをお薦めします。これによって、すべてのデバイスで適切にレンダリングできるためです。）
Basic Tables モジュール	要素 : caption、table、td、th、tr Basic Tables では、ネストした表は使用できません。 OracleAS Wireless は、表の rowspan または colspan をサポートしていません。
Image モジュール	要素 : img
Object モジュール	要素 : object、param Object モジュールをイメージに対して使用する場合、サーバーはイメージ調整をサポートします（詳細は、A-5 ページの「Object モジュール」を参照）。
Meta Information モジュール	要素 : meta
Link モジュール	要素 : link
Base モジュール	要素 : base
Presentation モジュール	要素 : hr、b、big、i、small
Style Sheet モジュール	要素 : style
Style Attribute モジュール	属性 : style
Media Attribute モジュール	属性 : media（詳細は、A-11 ページの「OracleAS Wireless の MXML Media Attribute モジュール」を参照）

XHTML MP HelloWorld の例

1. XHTML MP 文書は、<xml> 宣言から始まる必要があります。文書の最初に次の文字を追加します。

```
<?xml version="1.0" standalone="yes"?>
```

2. <xml> 宣言の直後に、DOCTYPE 宣言を XHTML MP 文書に追加します。

```
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"

  "http://www.wapforum.org/DTD/xhtml-mobile10.dtd">
```

3. <DOCTYLE> 宣言の直後に、<html> 要素をドキュメント・ルートとして追加します。html 要素の開始タグと終了タグを追加します。また、head セクションも追加します。head セクションには、title 要素と style 要素が含まれます。

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
        "http://www.wapforum.org/DTD/xhtml-mobile10.dtd">

<html>
  <head>
    <title>Hello World</title>
    <style type="text/css">
      body {color : #000000}
      h1 {font-family : sans-serif; color : blue}
    </style>
  </head>
</html>
```

4. 次に、body セクションを追加します。この例では、body セクションに、input コントロールがあるフォームが含まれます。

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
        "http://www.wapforum.org/DTD/xhtml-mobile10.dtd">

<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
     <h1>My XHTML MP Page</h1>
    <form action="printForm.jsp" method="get">
      <div>
        <label for="text">
          Hello Visitor, Please Enter your name
        </label>
```

```
        <input id="text" name="text" type="text" size="5"/>
      </div>
      <div>
        <input name="submit" type="submit" value="Submit"/>
      </div>
    </form>
  </body>
</html>
```

これで、XHTML MP 文書をデプロイしてテストする準備ができました。作成者は、自分の Web サーバーに文書をデプロイできます。また、デプロイ時に、文書の MIME メディア・タイプ（content-type）が `application/vnd.wap.xhtml+xml` に設定されていることを確認する必要があります。MIME メディア・タイプは、プログラムによって設定するか、または Web サーバーの構成ファイルを使用して設定できます。また、作成者は、サンプルの送信ページとして使用するページを提供する必要があります。このページで、「Hello World」ページからのフォーム・データを問合せパラメータとして受け取ります。

OracleAS Wireless XML

各項の内容は、次のとおりです。

- OracleAS Wireless XML の概要
- OracleAS Wireless XML と OracleAS Wireless
- コンテンツの表示と書式設定
- 音声アクセスの場合のオーディオを使用した強調
- アプリケーションのナビゲーション
- ドキュメントのリンク
- データ入力とナビゲーション用のフォームの埋込み
- 高度なユーザー相互作用とチャネルの最適化

OracleAS Wireless XML の概要

OracleAS Wireless XML は、以前のバージョンに基づいています。今後は、XHTML+XForms および XHTML MP を使用する必要があります。

次の XML 文書を考えます。

```
<address>
<first-name>Chandra</first-name>
<last-name>Patni</last-name>
<street>400 Oracle Parkway</street>
<zip>94065</zip>
</address>
```

この例では、要素名を見れば、そこにカプセル化されているデータがわかります。この XML 文書は、XSL スタイルシートと呼ばれる別の XML 文書を使用して HTML に変換できます。別の XSL スタイルシートを使用すると、これと同じ XML 文書を WML に変換できます。変換後の文書は、携帯デバイスで表示できます。この機能により、XML は各種デバイスへのポータブル・データの表示と配信に適した言語となっています。XML コンテンツは、将来にわたって引き続き使用可能でもあります。別のスタイルシートを使用して、コンテンツを将来の任意のデバイスに配信できます。したがって、XML 変換は処理中にプログラムで実行できます。Oracle Application Server Wireless は、この変換を正確に実行するフレームワークを提供します。このため、Oracle Application Server Wireless のスキーマで定義された XML フォーマットで表されるコンテンツは、任意のデバイスに随時配信できます。

OracleAS Wireless XML と OracleAS Wireless

Oracle Application Server Wireless のコアでは、アプリケーションからの XML が XSL 変換を使用してデバイス固有のマークアップ言語に変換されます。Oracle Application Server Wireless はアプリケーションと相互作用し、XML をデバイス固有のマークアップ言語に変換するためのフレームワークを提供します。Oracle Application Server Wireless には XML Schema も用意されています。XML Schema は、アプリケーションのコンテンツを任意のデバイスにレンダリングするためのユーザー・インタフェース作成に使用できる要素です。

コンテンツの表示と書式設定

項ごとに様々なトピックを記載しています。各項の内容は、次のとおりです。

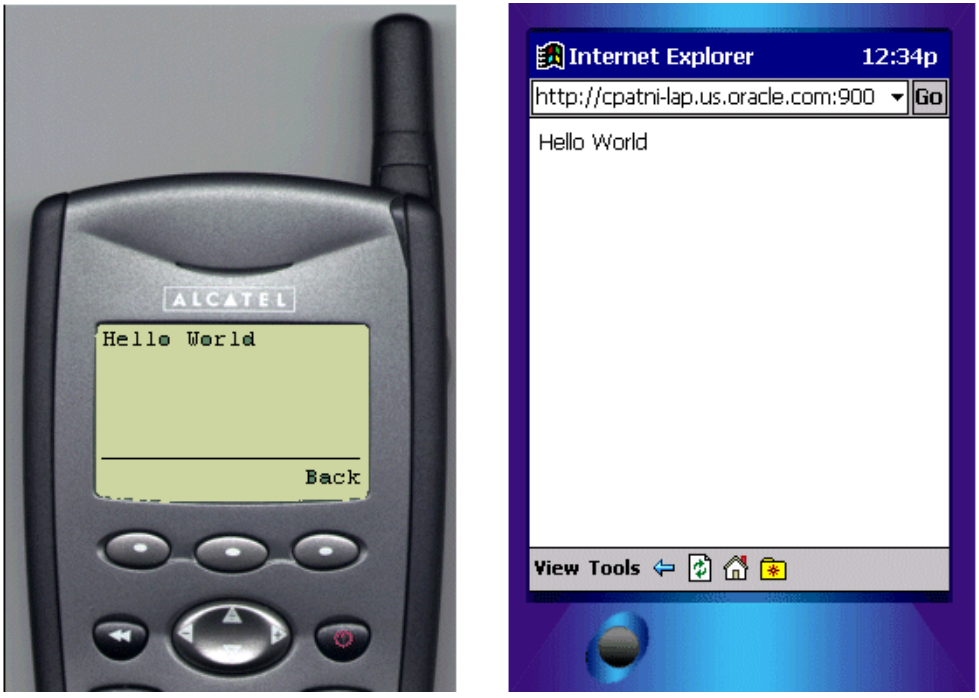
Hello World の例

最初の例に、従来の「Hello World」のコンテンツをモバイル・デバイスに表示する方法を示します。

HelloWorld.xml

```
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>
<!DOCTYPE SimpleResult PUBLIC "-//ORACLE//DTD SimpleResult 1.1//EN"
"http://xmlns.oracle.com/ias/dtds/SimpleResult_1_1_0.dtd">
<SimpleResult>
 <SimpleContainer>
  <SimpleText>
   <SimpleTextItem>Hello World</SimpleTextItem>
  </SimpleText>
 </SimpleContainer>
</SimpleResult>
```

図 8-2 モバイル・デバイスに表示される Hello World のコンテンツ



この例では、XML はデバイス固有のマークアップ言語に変換され、ポケット PC と電話の画面にレンダリングされます。この例は XML の能力を示しています。アプリケーション・プログラマには、ターゲット・デバイスに関する知識は不要です。Oracle Application Server Wireless は、XML を各種デバイス画面にレンダリングします。次の項では、前述の例で使用されている XML の要素、タグおよび属性について説明します。また、デバイスの画面や音声ブラウザでのコンテンツの表示と書式設定に使用できる他のタグについても説明します。

DOCTYPE 宣言

Oracle Application Server Wireless 用に作成する XML 文書では、DOCTYPE 宣言でスキーマのバージョンを指定する必要があります。下位互換性（DOCTYPE 宣言がない場合）を維持するために、Oracle Application Server Wireless Edition 1.0 用のスタイルシートが適用されます。ただし、Oracle Application Server Wireless Runtime で 1.0 のスタイルシートを使用できない場合は、DOCTYPE 宣言の有無に関係なく Oracle Application Server Wireless 1.*x* のスタイルシートが使用されます。1.*x* のスタイルシートが見つからない場合は、エラーになります。

SimpleResult

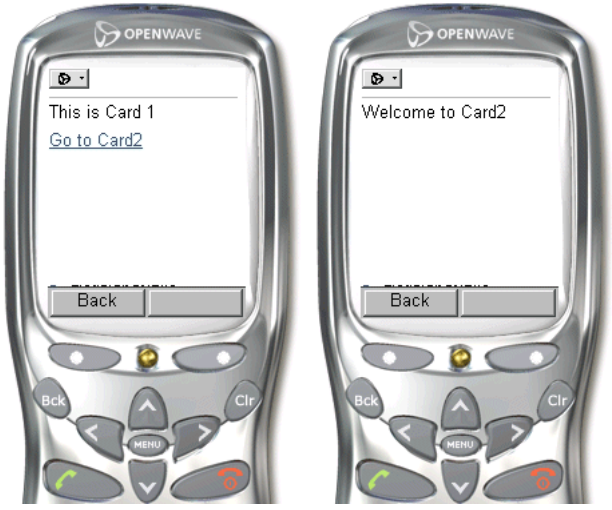
SimpleResult は、Oracle Application Server Wireless の XML Schema のルート要素です。有効な各 Oracle Application Server Wireless XML 文書には、ルート要素として *SimpleResult* が必要です。SimpleResult には、マルチカード・デッキに使用できるように複数の *SimpleContainer* ブロックを含めることができます。

SimpleContainer SimpleContainer は、Form、Menu および Text など、すべての主要ブロック構成メンバーのルートです。メニュー、テキストおよびフォームなどのアイテム要素は、デッキに含まれているカードとして機能できます。`DeckExample.xml` は、カードのプレースホルダとしての SimpleText の使用方法を示しています。ターゲット・デバイスの制限とデバイス上のデッキ・サイズの制限を考慮し、デッキごとのカード数と 1 件のリクエストでの合計コンテンツ・サイズについて、様々な組合せを試して適正なカード数とサイズを判断する必要があります。

DeckExample.xml

```
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>
<!DOCTYPE SimpleResult PUBLIC "-//ORACLE//DTD SimpleResult 1.1//EN"
"http://xmlns.oracle.com/ias/dtds/SimpleResult_1_1_0.dtd">
<SimpleResult>
 <SimpleContainer>
  <SimpleText id="card1">
   <SimpleTextItem>This is Card 1
    <SimpleBreak/>
    <SimpleHref target="#card2">Go to Card2</SimpleHref>
   </SimpleTextItem>
  </SimpleText>
  <SimpleText id="card2">
   <SimpleTextItem>Welcome to Card2</SimpleTextItem>
  </SimpleText>
 </SimpleContainer>
</SimpleResult>
```

図 8-3 携帯電話に表示されているカード



SimpleText、SimpleTextItem 通常、SimpleTextItem の内容はパラグラフに変換されます。SimpleText 要素を使用すると、SimpleTextItem をグループ化できます。SimpleText 要素には、1 つ以上の SimpleTextItem が含まれます。SimpleText タグの id 属性を使用すると、SimpleText 要素をデッキとして参照できます。SimpleText は、WML デバイスと HDML デバイスに別個のカードとしてレンダリングされます。SimpleHref を、HTML のアンカーと同様に SimpleTextItem の子として使用できます。SimpleHref の詳細は、8-97 ページの「SimpleHref、SimpleTimer」を参照してください。SimpleText および SimpleTextItem の *deviceclass* 属性は、「pdabrowser」、「pcbrowser」、「voice」、「microbrowser」、「micromessenger」および「messenger」を取り、小型画面付きのクライアントまたは音声クライアント用の処理を指示できます。deviceclass 属性がない場合、コンテンツは小型画面付きのデバイスと音声対応デバイスの両方にレンダリングされます。デフォルトでは、これらのタグに囲まれたテキストは、テキスト音声合成（TTS）を使用して表されます。SimpleAudio タグと deviceclass 属性を併用すると、このデフォルト動作をオーバーライドするように指定できます。ユーザー操作が簡便になるように、録音されたオーディオを提供できる場合は TTS を使用しないでください。音声インタフェースの場合は、SimpleAudio を使用できます。使用方法は、次のコード抜粋を参照してください。

```
<SimpleText>
 <SimpleTextItem>
  <SimpleAudio src="http://www.domain.com/filename.wav" deviceclass="voice">Alt text
for TTS if the wave file is not found.
 </SimpleAudio>
 </SimpleTextItem>
 <SimpleTextItem deviceclass="microbrowser"> Text for small screen devices
 </SimpleTextItem>
</SimpleText>
```

注意： 指定する .wav ファイルは、CCITT mu-law、8 ビット、8kHz である必要があります。

表示の書式設定

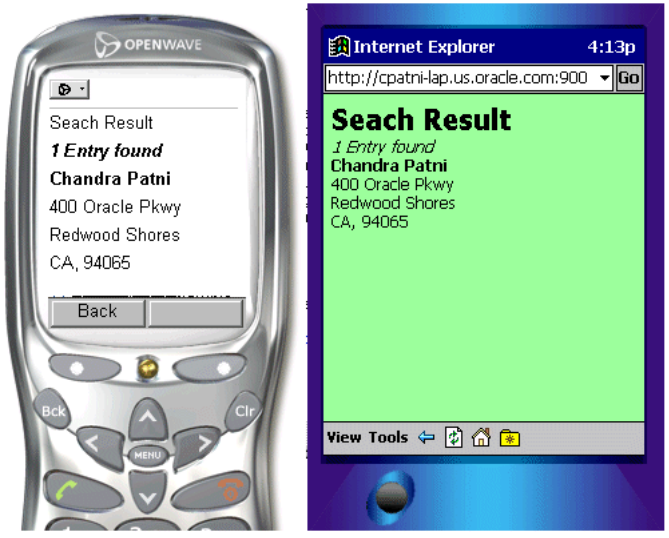
SimpleBreak、SimpleStrong および SimpleEm これらの要素は、画面上のテキスト・コンテンツの表示を微調整するために使用されます。SimpleStrong で囲まれたテキストは（通常は太字で）強調表示されます。SimpleEm で囲まれたテキストも強調表示されますが、通常はイタリックが使用されます。音声対応のアプリケーションの場合は、level 属性を使用して強調レベルを指定できます。level 属性に有効な値は、strong、moderate、none および reduced です。

SimpleBreak を指定すると、ページ上のこのタグの位置で改行されます。rule 属性を使用すると、HTML 出力用に行 <hr> を表示できます。deviceclass を使用すると、小型画面付きデバイスまたは音声対応のデバイス、あるいはその両方用にディレクティブを処理できます。音声対応のアプリケーションの場合は、SimpleBreak を使用すると休止が挿入され、msecs および size 属性を使用すると休止時間を制御できます。詳細は、次の例を参照してください。

FormattingExample.xml

```
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>
<!DOCTYPE SimpleResult PUBLIC "-//ORACLE//DTD SimpleResult 1.1//EN"
"http://xmlns.oracle.com/ias/dtds/SimpleResult_1_1_0.dtd">
<SimpleResult bgcolor="99ff99">
 <SimpleContainer>
  <SimpleText>
   <SimpleTitle>Seach Result</SimpleTitle>
   <SimpleTextItem>
    <SimpleEm level="strong">1 Entry found</SimpleEm>
    <SimpleBreak msecs="500"/>
    <SimpleStrong>Chandra Patni</SimpleStrong>
    <SimpleBreak/>400 Oracle Pkwy
    <SimpleBreak/>Redwood Shores
    <SimpleBreak/>CA, 94065
   </SimpleTextItem>
  </SimpleText>
 </SimpleContainer>
</SimpleResult>
```

図 8-4 書式設定例の結果



表と基本的な書式設定の例

SimpleTable、SimpleTableHeader、SimpleTableBody、SimpleRow および SimpleCol SimpleTable は、表を表示します。表は、それぞれ SimpleTableHeader と SimpleTableBody で抽象化されたヘッダーと本体で構成されています。表の本体は、SimpleRow および SimpleCol 要素で構成されています。表のセルにはイメージを使用できます。TableExample.xml には、表要素の例が用意されています。

TableExample.xml

```
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>
<!DOCTYPE SimpleResult PUBLIC "-//ORACLE//DTD SimpleResult 1.1//EN"
"http://xmlns.oracle.com/ias/dtds/SimpleResult_1_1_0.dtd">
<SimpleResult>
 <SimpleContainer>
  <SimpleTable >
   <SimpleTitle> My Portfolio </SimpleTitle>
   <SimpleTableHeader>
    <SimpleCol>Symbol</SimpleCol>
    <SimpleCol>Price</SimpleCol>
    <SimpleCol>Delta</SimpleCol>
   </SimpleTableHeader>
   <SimpleTableBody>
    <SimpleRow>
     <SimpleCol>ORCL</SimpleCol>
```

```
      <SimpleCol>18.32</SimpleCol>
      <SimpleCol>+0.24</SimpleCol>
     </SimpleRow>
     <SimpleRow>
      <SimpleCol>SUNW</SimpleCol>
      <SimpleCol>17.35</SimpleCol>
      <SimpleCol>+1.06</SimpleCol>
     </SimpleRow>
     <SimpleRow>
      <SimpleCol>CSCO</SimpleCol>
      <SimpleCol>20.30</SimpleCol>
      <SimpleCol>+0.24</SimpleCol>
     </SimpleRow>
     <SimpleRow>
      <SimpleCol>MSFT</SimpleCol>
      <SimpleCol>6647</SimpleCol>
      <SimpleCol>+0.28</SimpleCol>
     </SimpleRow>
    </SimpleTableBody>
   </SimpleTable>
  </SimpleContainer>
</SimpleResult>
```

図 8-5 表と基本的な書式設定の例の結果

OracleAS Wireless XML でのイメージ調整のサポート

アプリケーション開発者は、SimpleImage タグを使用して、OracleAS Wireless XML アプリケーションにイメージを埋め込みます。イメージ調整は、SimpleImage タグの `addImageExtension` 属性を使用してサポートされます。次の表に、addImageExtension に対する値のセマンティクスおよび available 属性を示します。

表 8-5 値のセマンティクス

例	addImageExtension	available	セマンティクス
1	設定しない、true	設定しない	available が設定されないため、デフォルトのデバイス拡張機能を使用します。
2	設定しない、true	拡張機能リスト	拡張機能が一致する場合は、使用可能な拡張機能を使用します。
3	false	設定しない	拡張機能は追加しないでください。`src` はイメージ URL です。
4	false	拡張機能リスト	available リストは無視し、拡張機能は追加しないでください。
5	auto	設定しない	`src` URL の入力を調整用に使用して、すべてのデバイスのイメージを調整します。
6	auto	拡張機能リスト	サポートされている拡張機能を調整用の入力として使用して、すべてのデバイスのイメージを調整します。

例 1: 最も単純な例として、単一イメージがアプリケーションで提供され、デバイス・プロファイルに基づいて、イメージをサポートするすべてのデバイスについてその単一イメージを調整します。

```
<SimpleImage src="http://www.oracle.com/admin/images/oralogo.gif" alt="Oracle logo"
addImageExtension="auto" />
```

例 2: 複数のイメージが提供され、最も適切なイメージが選択されます。サイズまたはコンテンツ・フォーマットを調整する必要がある場合は、最も適切なイメージが調整されます。このとき、デバイスで GIF イメージがサポートされている場合は、必要に応じて GIF イメージを使用して、サイズまたはコンテンツ・フォーマットを調整します。デバイスで WBMP がサポートされている場合は、ネストしたオブジェクトで WBMP イメージを使用してサイズを調整します。デバイスでいずれもサポートしていない場合は、最初にリストされたイメージ・フォーマットが使用されます。

```
<SimpleImage src="http://../images/oralogo" alt="Oracle logo" available="gif wbmp"
addImageExtension="auto"/>
```

例 3: 複数のイメージが提供され、イメージ調整はオフになっています。次の 2 つの行は同じ意味です。

```
<SimpleImage src="http://../images/oralogo" alt="Oracle logo" available="gif wbmp"
addImageExtension="true"/>

<SimpleImage src="http://../images/oralogo" alt="Oracle logo" available="gif wbmp"
/>
```

例 4: イメージ調整をオフにし、提供されたイメージのみ使用します。このフォーマットをサポートしていないデバイスは無視します。

```
<SimpleImage src="http://www.oracle.com/admin/images/oralogo.gif" alt="Oracle logo"
addImageExtension="false" />
```

音声アクセスの場合のオーディオを使用した強調

SimpleAudio と SimpleSpeech

SimpleAudio 要素を使用すると、オーディオを再生できます。src 属性で指定するファイルは、8 ビットの mu-law フォーマットである必要があります。SimpleSpeech 要素を使用すると、抑揚、ピッチおよび他の VoiceXML テキスト音声合成エンジンのパラメータを制御できます。たとえば、class 属性を使用して、SimpleSpeech のコンテンツを say-as タイプの phone、date、digits、literal、currency、number または time として解析することを指定できます。使用方法は、次の例を参照してください。

```
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>
<!DOCTYPE SimpleResult PUBLIC "-//ORACLE//DTD SimpleResult 1.1//EN"
"http://xmlns.oracle.com/ias/dtds/SimpleResult_1_1_0.dtd">
<SimpleResult>
 <SimpleContainer>
  <SimpleText>
   <SimpleTextItem>
    <SimpleAudio src="welcome1.wav">Welcome to Oracle Mobile, India Development
Center</SimpleAudio>
    <SimpleBreak/>
    <SimpleAudio src="welcome2.wav">You can contact us at phone number
</SimpleAudio>
    <SimpleBreak/>
    <SimpleAudio src="phone.wav">
     <SimpleSpeech class="phone">91 080 552 8335</SimpleSpeech>
    </SimpleAudio>
   </SimpleTextItem>
  </SimpleText>
 </SimpleContainer>
</SimpleResult>
```

図 8-6 SimpleAudio と SimpleSpeech の例の結果

音声ナビゲーションの推奨事項

開発者は、Oracle Application Server Wireless 用のアプリケーションの作成中の設計時に、音声ナビゲーションを考慮する必要があります。適切に設計された音声アプリケーションのセマンティクスは、小型画面付きのデバイスやデスクトップ・アプリケーションと異なる傾向があります。Oracle Application Server Wireless ではサービス用の音声インタフェースが自動的に提供されますが、システムは音声制御機能を持つ小型画面付きデバイスのブラウザとしての使用を意図されていません。この場合は、音声インタフェースが開発後に追加されます。アプリケーション開発者は、独自の適切な小型画面と音声インタフェースを持つサービスを開発する必要があり、このような各種デバイスのそれぞれの強みを活用できます。

この種のサービスを初めて開発する場合は、次のモデルに従う必要があります。

1. 小型画面付きのデバイスおよびオーディオ・インタフェースと同じフローおよびマークアップを使用して、サービスの基本バージョンを作成します。
2. 小型画面付きのデバイスと音声電話でテストします。サービスにアクセスできる場合は、作業完了です。

大きいサービス・クラスの場合、特に情報をメニュー方式で提供するデバイスの場合は、この方法で十分です。適切に動作しないと思われるインタフェースがある場合は、それを改善するための作業がいくつかあります。

1. 各デバイス・クラスについてインタフェースを強化するために調整できる属性値が多数あります。
2. 前述の方法で不十分な場合は、*deviceclass* に応じてユーザー・インタフェースの特定の要素を含めたり除外できます。
3. *deviceclass* に応じて、サービスに通じる様々なパスを選択的にたどって、ユーザー・インタフェースを変更できます。

アプリケーションのナビゲーション

この項では、小型デバイスと音声の観点からテキストの書式を処理する Mobile XML を記述する作業の特性について説明する前に、効率的なユーザー・インタフェースを記述する方法について説明します。このユーザー・インタフェースにより、モバイル・ユーザーは最小限の労力で必要なビジネス・ロジックを獲得できます。

概要

この項では、メニュー、ハイパーリンク、電子メール、ヘルプおよび表紙など、各フォームのナビゲーション要素を含む Oracle Application Server Wireless XML ページの作成方法を詳しく説明します。フォームの作成に必要な要素は、メニューとは異なります。これはコア要素であり、Wireless 開発者が様々なデバイス間で豊富な機能を犠牲にせずにユーザー入力を簡素化する、効率的なモバイル・アプリケーションを作成するために必要になるためです。

音声によるナビゲーションは本来が小型画面付きデバイスよりも複雑であるため、この項では小型デバイス用 Oracle Application Server Wireless XML の基礎と、必要な音声の追加に重点を置いて説明します。

メニューを使用すると、サービスのコンシューマは簡単に定義済のオプションにナビゲートして、特定のオプションの様々な URL を起動できます。これに対して、フォームの場合は、通常はターゲットが 1 つで、ユーザー入力に基づいて次ページが指定されるという点でメニューとは異なります。

基本的なナビゲーション

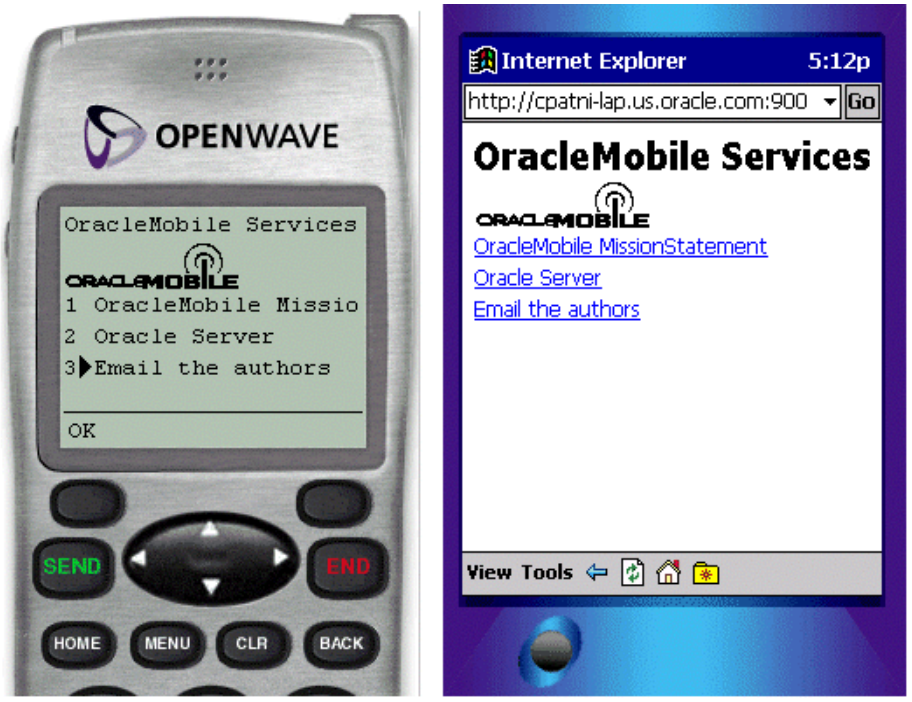
SimpleMenu、SimpleMenuItem

SimpleMenu 要素は、SimpleMenuItem 要素に定義された選択可能なメニュー項目を持つ単一のメニューを表します。各メニューの先頭にイメージを追加できますが、大きいタイトルやイメージの使用は避けてください。例については、SimpleMenuExample.xml を参照してください。

SimpleMenuExample.xml

```
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>
<!DOCTYPE SimpleResult PUBLIC " = //ORACLE//DTD SimpleResult 1.1//EN"
"http://xmlns.oracle.com/ias/dtds/SimpleResult_1_1_0.dtd">
<SimpleResult>
 <SimpleContainer>
  <SimpleMenu>
   <SimpleTitle>OracleMobile Services
    <SimpleImage src="
http://portal.oraclemobile.com/other/oow/oramobile"alt="Oracle Software
Powers the Internet"/>
   </SimpleTitle>
   <SimpleMenuItem target="mission.xml">OracleMobile
MissionStatement</SimpleMenuItem>
   <SimpleMenuItem target="timer.xml">Oracle Server</SimpleMenuItem>
   <SimpleMenuItem target="email.xml">Email the authors</SimpleMenuItem>
   </SimpleMenu>
  </SimpleContainer>
 </SimpleResult>
```

図 8-7 単純なナビゲーション例の結果



音声によるナビゲーション

システムは、メニュー要素の項目を読みあげると同時に、*SimpleMenuItem* 要素の値をリスニングします。これらの値の 1 つが認識されると、ターゲット URL がフェッチされます。ユーザーが音声で何も指定しなければ、システム・デフォルトの「noinput」のメッセージが流れます。ユーザーが音声で何かを指定し、それをシステムで認識できない場合は、システム・デフォルトの「nomatch」のメッセージが再生されます。ただし、この種のメッセージはアプリケーション・プログラマが制御できます。このようなフェイルオーバー・ロジックは、堅牢な音声アプリケーションに不可欠です。アプリケーション開発者は、この種の機能を広範囲に使用する必要があります。多数の項目を含むメニューの場合、音声インタフェースでは、ユーザーに対してメニュー項目全体を読み取らないようにする必要があります。デフォルトを無効にするには、SimpleMenu の autoprompt 属性を false に設定します。かわりに、アプリケーションはユーザー入力を待ってから、ユーザーから要求された場合はヘルプとしてオプション・リストのみを表示します。例については、`EnhancedSimpleMenuExample.xml` を参照してください。アプリケーションで使用されるタグと要素については、この章の後半を参照してください。

EnhancedSimpleMenuExample.xml

```
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>
<!DOCTYPE SimpleResult PUBLIC " = //ORACLE//DTD SimpleResult 1.1//EN"
"http://xmlns.oracle.com/ias/dtds/SimpleResult_1_1_0.dtd">
 <SimpleResult>
  <SimpleContainer>
   <SimpleMenu deviceclass="microbrowser pdabrowser pcbrowser micromessager
messenger">
   <SimpleTitle>Oracle Mobile Services
   <SimpleImage src="http://portal.oraclemobile.com/other/oow/oramobile"
alt="Oracle Software Powers the Internet"/></SimpleTitle>
   <SimpleTitle>Oracle Mobile Services</SimpleTitle>
   <SimpleMenuItem target="mission.xml">Oracle Mobile Mission
Statement</SimpleMenuItem>
   <SimpleMenuItem target="timer.xml">Oracle Server</SimpleMenuItem>
   <SimpleMenuItem target="email.xml">Email the authors</SimpleMenuItem>
  </SimpleMenu>
  <SimpleMenu deviceclass="voice" autoprompt="false">
   <SimpleTitle>
    <SimpleAudio src="title.wav">oracle mobile services
    </SimpleAudio>
   </SimpleTitle>
   <SimpleMenuItem target="mission.xml">Oracle Mobile Mission Statement
    <SimpleGrammar><grammar root="ms">
    <rule id="ms">
    <item repeat="0-1">Oracle</item> mission statement
    </rule>
    </grammar>
    </SimpleGrammar>
   </SimpleMenuItem>
   <SimpleMenuItem target="timer.xml">Oracle Server
    <SimpleGrammar><grammar root="server">
    <rule id="server">
    <item repeat="0-1">Oracle</item> server
    </rule>
    </grammar>
    </SimpleGrammar>
   </SimpleMenuItem>
   <SimpleMenuItem target="email.xml">Email the authors
    <SimpleGrammar><grammar root="email">
    <ruleid="email">
    <one-of>
    <item>email the authors</item>
    <item>email</item>
    <item>email authors</item>
    </one-of>
```

```
    </rule>
    </grammar>
    </SimpleGrammar>
   </SimpleMenuItem>
   <SimpleCatch type="noinput">
    <SimpleAudio src="menuOptions.wav">Please speak up. You may also say help.
    </SimpleAudio>
   </SimpleCatch>
   <SimpleCatch type="nomatch">
    <SimpleAudio src="nomatch.wav">I'm sorry, I did not understand you. Please say
that again or say help.</SimpleAudio>
   </SimpleCatch type="help">
    <SimpleAudio src="menuHelp.wav"> Help. Oracle Mobile. You may say mission
statement, oracle server or email the authors.
    </SimpleAudio>
  </SimpleMenu>
 </SimpleContainer>
</SimpleResult>
```

小型画面付きデバイスの場合、このアプリケーションの出力は前述のコードと同じですが、典型的な音声セッションの場合は次のようになることがあります。

システム：「Oracle Mobile Services」

ユーザー：「Help」

システム：「Help. Oracle Mobile. You may say mission statement, oracle server or email the authors.」

ユーザー：「I am going to trick you.」

システム：「I'm sorry, I did not understand you. Please say that again or say help.」

ユーザー：「Email authors.」

音声ゲートウェイには、SimpleTitle、SimpleTextItem などを読み取るテキスト音声合成（TTS）エンジンが用意されています。TTS でわかりやすいサウンドを出すには、適切な間隔と記号が必要です。

deviceclass が「voice」以外の値に設定されていないかぎり、SimpleFormOption および SimpleMenuItem にはテキストの記号を使用しないでください。これは、この 2 つのタグに囲まれたテキストは音声認識グラマーの生成に使用され、多くの音声ゲートウェイでは音声構文でテキストの記号を処理できないためです。開発者が合成メッセージを使用しない場合は、事前に録音されたオーディオ・ファイルを再生するように指定できます。オーディオ・ファイルのロケーションは、<SimpleAudio> タグで指定できます。通常、エンド・ユーザーが TTS を使用するのは望ましくないとみなされるため、できるかぎり TTS ではなく事前に録音された人の音声を使用する必要があります。

ドキュメントのリンク

SimpleHref、SimpleTimer

ドキュメントをリンクする場合は、ハイパーリンクとして SimpleHref を使用できます。また、次の 2 つの例のように、SimpleHref と mailto: ハンドラを使用して電子メールの送信に使用することもできます。同様に、通話機能を持つデバイスには、callto: ハンドラを使用できます。アプリケーション開発者は、通話機能またはメール機能をサポートする deviceclass 属性を指定する必要があります。SimpleHref を音声デバイスで使用することはお薦めしません。これは、テキスト音声合成エンジンで読み取られるインライン・テキストが、リンクを横断するために発声される語句でもあることを音声で示すことは問題があるためです。SimpleHref のかわりに SimpleMenuItem を使用してください。

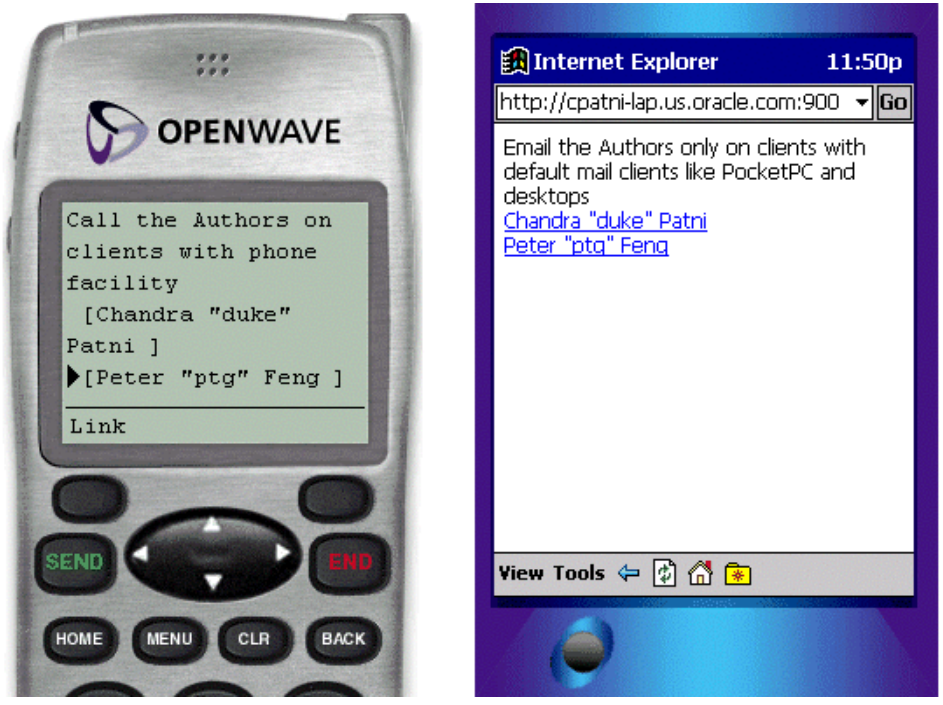
SimpleTimer を使用すると、指定した遅延の後に、文書をフェッチできます。また、この要素を使用して、商品展示による販売促進情報、スポンサ情報またはシステム単位の重要なメッセージを表示することもできます。

ContactAuthors.xml

```
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>
<!DOCTYPE SimpleResult PUBLIC "-//ORACLE//DTD SimpleResult 1.1//EN"
"http://xmlns.oracle.com/ias/dtds/SimpleResult_1_1_0.dtd">
<SimpleResult>
 <SimpleContainer>
  <SimpleText deviceclass="pdabrowser pcbrowser micromessenger messenger
microbrowser">
   <SimpleTextItem deviceclass="pdabrowser pcbrowser  micromessenger
messenger">Email the Authors only on clients with default mail clients like PocketPC
and desktops
    <SimpleBreak/>
    <SimpleHref target="mailto:chandra.patni@oracle.com">Chandra "duke" Patni
    </SimpleHref>
    <SimpleBreak/>
    <SimpleHref target="mailto:peter.feng@oracle.com">Peter "ptg" Feng
    </SimpleHref>
   </SimpleTextItem>
   <SimpleTextItem deviceclass="microbrowser">Call the Authors on clients with phone
facility
    <SimpleBreak/>
    <SimpleHref target="callto:1234567890">Chandra "duke" Patni
    </SimpleHref>
    <SimpleBreak/>
    <SimpleHref target="callto:1234567890">Peter "ptg" Feng
    </SimpleHref>
   </SimpleTextItem>
  </SimpleText>
  <SimpleMenu deviceclass="voice">
```

```
  <SimpleTitle>Call the Authors on voice clients</SimpleTitle>
  <SimpleMenuItem target="callto:1234567890">Chandra Patni</SimpleMenuItem>
  <SimpleMenuItem target="callto:1234567890">Peter Feng</SimpleMenuItem>
  </SimpleMenu>
 </SimpleContainer>
</SimpleResult>
```

図 8-8 電子メールによるデモ例の結果

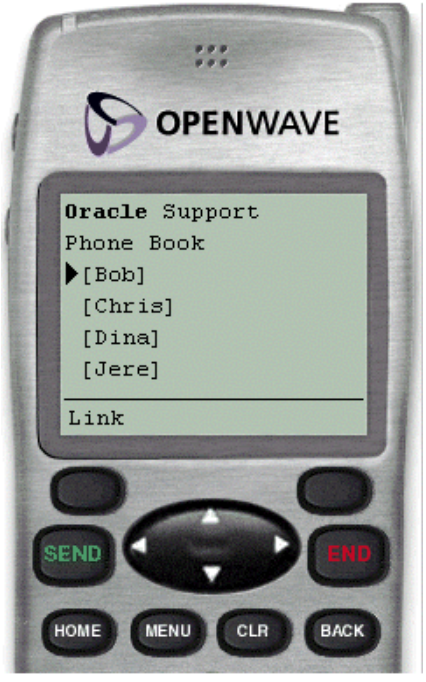


PhoneCallDemo.xml

```
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>
<!DOCTYPE SimpleResult PUBLIC "-//ORACLE//DTD SimpleResult 1.1//EN"
"http://xmlns.oracle.com/ias/dtds/SimpleResult_1_1_0.dtd">
<SimpleResult>
 <SimpleContainer>
  <SimpleText deviceclass="microbrowser">
    <SimpleTextItem><SimpleEm>Oracle</SimpleEm> Support </SimpleTextItem>
    <SimpleTextItem>Phone Book<SimpleBreak/>
    <SimpleHref target="callto:14155551212">Bob</SimpleHref>
    <SimpleHref target="callto:16505551212">Chris</SimpleHref>
```

```
    <SimpleHref target="callto:14085551212">Dina</SimpleHref>
    <SimpleHref target="callto:17075551212">Jere</SimpleHref>
    </SimpleTextItem>
  </SimpleText>
<SimpleMenu deviceclass="voice">
  <SimpleTitle><SimpleEm>Oracle</SimpleEm> Support <SimpleBreak/> Phone
Book</SimpleTitle>
  <SimpleMenuItem target="callto:14155551212">Bob</SimpleMenuItem>
  <SimpleMenuItem target="callto:16505551212">Chris</SimpleMenuItem>
  <SimpleMenuItem target="callto:14085551212">Dina</SimpleMenuItem>
  <SimpleMenuItem target="callto:17075551212">Jere</SimpleMenuItem>
  </SimpleMenu>
 </SimpleContainer>
</SimpleResult>
```

図 8-9 通話によるデモ例の結果



SimpleAction SimpleAction を使用すると、ユーザーを新規のコンテキストにナビゲートさせる送信操作を定義できます。モバイル・デバイスでは、送信操作を、携帯デバイスの場合はキーを押す操作、音声対応デバイスの場合はコマンドを音声で指定する操作など、デバイスの多数の入力方法に関連付けることができます。また、SimpleAction を異なるページやデッキ内の異なるカードへのナビゲーションに使用し、音声ブラウザのデフォルト動作をオーバーライドすることもできます。携帯電話の場合、主な使用方法は携帯電話と PDA のボタン（左右）をオーバーライドし、SimpleHref と同じナビゲーション機能を提供することです。

多くのプログラミング言語と同様に、特定のタイプの SimpleAction には有効範囲決定規則が適用されます。たとえば、SimpleAction を SimpleMenu の子として定義し、さらにそれを囲んでいる特定タイプの SimpleContainer の子としても定義すると、SimpleMenu 内の SimpleAction タグにより SimpleContainer の SimpleAction がオーバーライドされます。*type* 属性の値が異なる場合は、コンテキスト内で 2 つの SimpleAction がアクティブになります。2 つの要素が同じコンテキスト内で同じ *type* および同じ *deviceclass* の値を使用して定義されている場合、SimpleAction の動作は未指定になります。使用方法は、次の例を参照してください。

SimpleCache SimpleCache を使用すると、WAP ゲートウェイまたはクライアントのブラウザ、あるいはその両方によるコンテンツのキャッシュ・ポリシーを指定できます。

- WAP ゲートウェイが URL のコンテンツのキャッシュを許可されている場合、キャッシュ・ポリシーはパブリックであるとみなされます。
- デバイスによるコンテンツのキャッシュのみが許可されている場合、キャッシュ・ポリシーはプライベートであるとみなされます。

SimpleCache は SimpleHref、SimpleGo、SimpleMenuItem、SimpleAction などの子として指定できます。また、SimpleCache はユーザーに対してプリフェッチ・ポリシーの指定を許可することもできます（ブラウザでサポートされている場合）。その場合は、現行のコンテンツが表示されている間に、URL がプリフェッチされる必要があります。キャッシュされたデータの TTL（Time to live）は、引数としてミリ秒数をとる `ttl` 属性で指定します。

データが機密性を持つ場合や、指定した期間が経過すると失効する場合は、SimpleCache を使用する必要があります。

SimpleMeta SimpleMeta を使用すると、アプリケーションでデバイス・ブラウザを介してメタ情報を指定し、その情報をトランスフォーマに渡すことができます。

DocumentLinkingDemo.xml 次にドキュメントのリンク例を示します。

```
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>
<!DOCTYPE SimpleResult PUBLIC "-//ORACLE//DTD SimpleResult 1.1//EN"
"http://xmlns.oracle.com/ias/dtds/SimpleResult_1_1_0.dtd">
<SimpleResult>
 <SimpleContainer id=”message”>
 <SimpleTimer target="#employeePortal" timer="30"/>
  <SimpleText>
```

```
   <SimpleTextItem> There will be ice cream bars in every lobby at Headquarters to
promote the use of the new employee wireless portal.
   </SimpleTextItem>
  </SimpleText>
 </SimpleContainer>
 <SimpleContainer>
  <SimpleText id="employeePortal">
  <SimpleTitle>
   <SimpleImage valign="top" src=
http://portal.oraclemobile.com/other/oow/oraclemobile alt="oraclemobile icon"/>
  </SimpleTitle>
   <SimpleTextItem>Welcome to <SimpleEm>OracleMobile</SimpleEm> Employee Portal
<SimpleBreak/>
   </SimpleTextItem>
   <SimpleAction type="secondary" label="Support" target="phone.xml"/>
   <SimpleHref label="PORTAL" id="portal" target="form.xml"> enterPortal
   </SimpleHref>
  </SimpleText>
 </SimpleContainer>
</SimpleResult>
```

図 8-10 ドキュメントのリンクによるデモ例の結果

音声による拡張

SimpleDTMF SimpleDTMF では VoiceXML DTMF の構文（つまり、キー入力の順序）を指定します。音声アプリケーションの例では、ユーザーは音声で「withdraw」と指定するか、デバイス上で「2」を選択して、メニュー項目「withdraw」を選択できます。

SimpleDTMF.xml

```
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>
<!DOCTYPE SimpleResult PUBLIC "-//ORACLE//DTD SimpleResult 1.1//EN"
"http://xmlns.oracle.com/ias/dtds/SimpleResult_1_1_0.dtd">
<SimpleResult>
 <SimpleCache ttl="0"/>
 <SimpleContainer>
  <SimpleMenu wrapmode="nowrap" autoprompt="false">
   <SimpleTitle>Voice demo</SimpleTitle>
   <SimpleMenuItem target="deposit.jsp">Deposit
    <SimpleDTMF> <grammar root="one">
    <rule id="one">1</rule>
    </grammar>
    </SimpleDTMF>
   </SimpleMenuItem>
   <SimpleMenuItem target="HelloWorld.jsp">Withdraw
    <SimpleDTMF> <grammar root="two">
    <rule id="two">2</rule>
    </grammar>
    </SimpleDTMF>
   </SimpleMenuItem>
   <SimpleCatch type="cancel">
   <SimpleGo target="cancel.jsp/>
   </SimpleCatch>
   <SimpleCatch type="help">
    <SimpleAudio src="help2.wav">Help. For deposit, you may say deposit or press 1.
For withdraw, you
    may say withdraw or press 2.</SimpleAudio>
   </SimpleCatch>
   <SimpleCatch type="help" count="2">
    <SimpleAudio src="help.wav">Help. For deposit, you may say deposit or press 1.
For withdraw, you
    may say withdraw or press 2. You may also say cancel to return to account
menu.</SimpleAudio>
   </SimpleCatch>
  </SimpleMenu>
 </SimpleContainer>
</SimpleResult>
```

SimpleCatch SimpleCatch はイベントを捕捉します。これは音声専用のタグです。このタグを使用すると、noinput、nomatch、exit、cancel、error、help、telephone.disconnect などの定義済の音声イベントやエラー条件を獲得し、それに対する操作を実行できます。たとえば、noinput イベントの場合は、ユーザーに対してなんらかのヘルプ指示を表示し、入力を求めるプロンプトを再表示できます。イベント型は、SimpleCatch に必須の type 属性で指定します。また、count 属性を使用してイベントの発生件数を得ることができます。デフォルト値は 1 です。この属性によって、イベントの複数の発生を複数の方法で処理できます。たとえば、イベントの n 番目の発生を、それまでの発生とは異なる方法で処理できます。頻繁に発生する使用例では、件数の増加につれてヘルプの詳細を増やすために使用できます。使用方法は、SimpleDTMF.xml を参照してください。

SimpleGrammar SimpleGrammar は、カスタマイズされた音声認識グラマーを提供します。この構文を使用すると、開発者はリスニングするボキャブラリのみでなく、発声とデータ値の間のマッピングを提供できます。この種のマッピングのルールがリモート・ロケーションにある場合は、src 属性を使用してファイル名を指定できます。次の例に、SimpleGrammar の使用方法を示します。

```
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>
<!DOCTYPE SimpleResult PUBLIC "-//ORACLE//DTD SimpleResult 1.1//EN"
"http://xmlns.oracle.com/ias/dtds/SimpleResult_1_1_0.dtd">
<SimpleResult>
 <SimpleContainer>
  <SimpleMenu deviceclass="voice">
   <SimpleTitle src="title.wav">Please select a freeway</SimpleTitle>
   <SimpleMenuItem target="./traffic.jsp?index=5">I 5
    <SimpleGrammar> <grammar route="i5">
    <rule id="i5">
    <one-of>
    <item>i five</item>
    <item>interstate five</item>
    <item>five</item>
    <item>route five</item>
    <item>san diego</item>
    </one-of>
    </rule>
    </grammar>
    </SimpleGrammar>
   </SimpleMenuItem>
   <SimpleMenuItem target="./traffic.jsp?index=8 ">I 8
    <SimpleGrammar><grammar root="i8">
    <rule id="i8">
    <one-of>
    <item>i eight</item>
    <item>interstate eight</item>
    <item>eight</item>
    <item>route eight</item>
```

```
    <item>alvarado freeway</item>
    <item>mission valley freeway</item>
    <item>ocean beach freeway</item>
    </one-of>
    </rule>
    </grammar>
    </SimpleGrammar>
   </SimpleMenuItem>
   <SimpleMenuItem target="./traffic.jsp?index=15 ">I 15
    <SimpleGrammar> <grammar root="i15">
    <rule id="i15">
    <one-of>
    <item>i fifteen</item>
    <item>fifteen</item>
    <item>nterstate fifteen</item>
    <item>escondido freeway</item>
    <item>escondido</item>
    </one-of>
    </rule>
    </grammar>
    </SimpleGrammar>
   </SimpleMenuItem>
   <SimpleMenuItem target="./traffic.jsp?index=805 ">I 805
    <SimpleGrammar> <grammar root="i805">
   <rule id="i805">
   <one-of>
   <item>i eight zero five</item>
   <item>i eight hundred five</item>
   <item>eight zero five</item>
   <item>eight hundred five</item>
   <item>interstate eight zero five</item>
   <item>interstate eight hundred five</item>
   <item>route eight zero five</item>
   <item>route eight hundred five</item>
   </one-of>
   </rule>
   </grammar>
    </SimpleGrammar>
   </SimpleMenuItem>
  </SimpleMenu>
 </SimpleContainer>
</SimpleResult>
```

この例では、最後のメニュー・オプションが「i eight hundred five」であっても、ユーザーは <one-of> 要素の <item> に指定されたコマンドの 1 つを音声で指定できます。SimpleGrammar は、ユーザーが親しみやすく使用しやすい音声アプリケーションの作成に役立つ構成メンバーです。また、アプリケーション開発者は、そのローカライゼーションの

問題をある程度取り込むこともできます。たとえば、「sure」、「ok」、「yes」、「please」および「yes please」は、いずれも世界各地で（英語圏での）「yes」を指すために使用されます。SimpleGrammar を使用すると、このような音声表現の多様性をアプリケーションに取り込むことができます。

返信メールの例 ここでは、返信メールを実装するために使用するコードを示します。

たとえば、電話機で電子メールを聞いたユーザーが「reply」と言うと、ユーザーに対して、返信を読みあげてシャープ（#）キーを押すようにメッセージが表示されます。録音された返信は、元の電子メール送信者に添付ファイルとして送信されます。

これを実装するために、発声された内容は、VoiceXML ゲートウェイで録音された後、マルチチャネル・サーバーとアプリケーションに渡され、返信メールに添付されます。

MXML:

```
<SimpleResult>
  <SimpleContainer>
    <SimpleForm method="post"
  enctype="application/x-www-form-urlencoded" target="recordaudio.jsp">
      <SimpleFormItem type="audio" enctype="audio/wav"
> name="recorded_audio_msg" beep="true" dtmfterm="true">
        Say something to record after the beep.
      </SimpleFormItem>
    </SimpleForm>
  </SimpleContainer>
</SimpleResult>
```

次に、recordaudio.jsp（一部）を示します。

```
// WRITE THE AUDIO CONTENT TO A FILE
// AND PLAY BACK THE CONTENTS
String myrecording = request.getParameter("record_audio_msg");
String wavdirectory = "path/testharness/audio/"; // ABSOLUTE PATH OF WHERE TO STORE
THE AUDIO
String myWaveFile  = wavdirectory + "test.wav";
File myWFile = new File(myWaveFile);
String result = "";
try {
  byte[] myWaveBytes = myrecording.getBytes();
  FileOutputStream myFileOutput = new FileOutputStream(myWaveFile);
  myFileOutput.write(myWaveBytes);
  myFileOutput.close();
  result = "<SimpleAudio
src=¥"http://iaswvoice.oracle.com/testharness/audio/" + fileName +
"test.wav¥"/>"; // THIS IS TO CHECK TO MAKE SURE THE FILE WAS RECORDED
```

```
} catch (IOException e) {
  result = "Error No Audio File Created";
}
  %>
Value: <%=result%> <!-- PLAY THE AUDIO FILE FOR THE USER, SEE ABOVE RESULT STRING
-->
```

Mobile XML の音声によるナビゲーション要素 次の基本的な音声コマンドは、ユーザーが随時使用できます。通常は、*help* および *cancel* に対するシステムの応答を個々のサービスにあわせて調整する必要があります。

- **Main menu**: 任意の時点で発声できます。デフォルトでは OracleAS Wireless のメイン・メニューに移動します。
- **Goodbye**: OracleAS Wireless の 1 つのインスタンスまたはユーザーとのセッションを終了するには電話を切ります。
- **Exit**: Goodbye と同じ。
- **Help**: 状況依存ヘルプ。
- **Cancel**: システムがコマンドや入力を正しく認識していないときなどに、ダイアログを中断または再開する場合に使用します。

Help Help は、音声アプリケーションでユーザーが「help」コマンドを起動したときに、状況依存ヘルプを提供するために使用されます。音声インタフェースでは、できるかぎり Help を使用する必要があります。小型画面によるアプリケーションのヘルプとは異なり、音声によるヘルプは音声インタフェースのナビゲーションに不可欠であるため、開発時に取り込んでください。使用方法は、`EnhancedSimpleMenuExample.xml` を参照してください。

データ入力とナビゲーション用のフォームの埋込み

項ごとに様々なトピックを記載しています。各項の内容は、次のとおりです。

概要

フォームは、ユーザーとの対話用の基本的なビルディング・ブロックを提供します。電話および PDA 用のフォームは、フォーム要素を除いてまったく同じです。HTML のフォームと同様に、モバイル・デバイスのフォームもサーバーに名前 / 値パラメータを渡すために使用されます。サポートされている場合は、デバイスの画面上に複数のフォーム項目をレイアウトできます。したがって、ユーザーはフォーム項目を任意の順序で移入できます。フォーム項目について一定の書式制限を指定し、フォーム・フィールドの型、保証および妥当性を確保できます。たとえば、米国の郵便番号には 5 桁という制限を指定できます。ただし、妥当性チェックのほとんどはサーバー側で実行する必要があります。この制約は、デバイス上のリソースが限られていることによるものです。音声ブラウザの場合は、すべてを音声ゲートウェイで処理する必要があります。このため、マークアップ言語レベルで様々な妥当性チェックと例外処理が使用可能になります。

ユーザーとの基本的な対話

SimpleForm SimpleForm は HTML フォームに似ており、SimpleFormItem と SimpleFormSelect の任意のコレクションを単一のエンティティとして提供します。SimpleFormSelect を使用すると、リスト、ラジオ・ボタンまたはチェックボックス・コントロールを表示できます。フォームには子として SimpleTitle があり、指定した場合はフォームのタイトルとして表示されます。SimpleForm と SimpleBind を併用すると、複数の方法でフォーム処理をトリガーでき、フォームの送信時に複数のタスクを実行できます。

SimpleFormItem SimpleFormItem は、デスクトップ・ブラウザのテキスト・フィールド、テキスト領域、パスワード・フィールドおよび非表示フィールドと等価です。項目のタイプは、displaymode 属性を使用して指定できます。属性値は、text、textarea、noecho または hidden です。SimpleFormItem を使用すると、ユーザーからの入力を取得できます。この要素はプロンプトを表示し、ユーザーからの入力を待ちます。この要素のコンテンツは解析可能な文字形式で表され、フォーム項目のデフォルト値を指定します。たとえば、ログイン画面とゲスト・ブック画面を次の例のように表示できます。

FormExample.xml

```
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>
<!DOCTYPE SimpleResult PUBLIC "-//ORACLE//DTD SimpleResult 1.1//EN"
"http://xmlns.oracle.com/ias/dtds/SimpleResult_1_1_0.dtd">
<SimpleResult>
 <SimpleContainer>
  <SimpleForm target="login.jsp" method="post">
   <SimpleFormItem name="userName">User Name:</SimpleFormItem>
   <SimpleFormItem name="password" displaymode="noecho">Password:</SimpleFormItem>
  </SimpleForm>
 </SimpleContainer>
</SimpleResult>
```

図 8-11 FormExample.xml の例の結果

GuestBook.xml

```
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>
<!DOCTYPE SimpleResult PUBLIC "-//ORACLE//DTD SimpleResult 1.1//EN"
"http://xmlns.oracle.com/ias/dtds/SimpleResult_1_1_0.dtd">
<SimpleResult>
 <SimpleContainer>
  <SimpleForm target="sendMail.jsp" method="post">
   <SimpleTitle>Thanks for signing my guestbook.</SimpleTitle>
   <SimpleFormItem name="Name">Name:</SimpleFormItem>
   <SimpleFormItem name="message" displaymode="textarea">Message:</SimpleFormItem>
```

```
  </SimpleForm>
 </SimpleContainer>
</SimpleResult>
```

図 8-12 GuestBook.xml の例の結果

ユーザー・フォームの完成

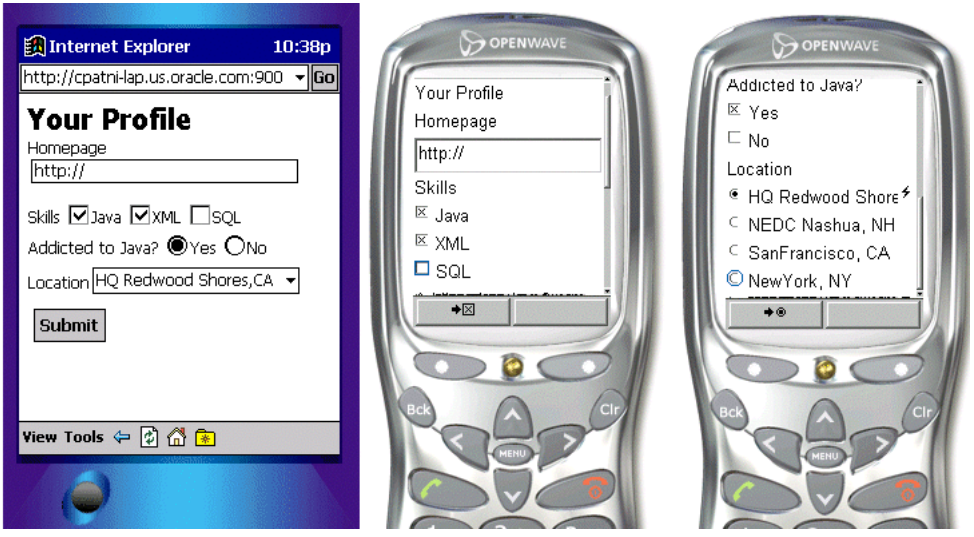
SimpleFormSelect、SimpleFormOption および SimpleOptGroup これらの要素では選択したオプションのリストを表示します。`displaymode` 属性を使用して、ドロップダウン・リスト、チェックボックスおよびラジオ・ボタンを表示できます。チェックボックスまたはオプション・リストでは、`multiple` 属性を使用して単一選択または複数選択を許可できます。表示される項目は、*SimpleFormOption* 要素で抽象化されます。SimpleOptGroup では、SimpleFormOption 要素が階層形式でグループ化されます。この要素は、長いオプション・リストを見やすく表示できない小型画面付きデバイスに役立ちます。SimpleFormOption 要素のコンテンツは解析可能な文字データで表され、フォーム項目のデフォルト値を指定します。使用方法は、次の例を参照してください。

Profile.xml

```
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>
<!DOCTYPE SimpleResult PUBLIC "-//ORACLE//DTD SimpleResult 1.1//EN"
"http://xmlns.oracle.com/ias/dtds/SimpleResult_1_1_0.dtd">
<SimpleResult>
 <SimpleContainer>
```

```
  <SimpleForm name="employeeinfo" target="process.jsp">
   <SimpleTitle>Your Profile</SimpleTitle>
   <SimpleFormItem name="homepage" default="http://">Homepage</SimpleFormItem>
   <SimpleFormSelect name="skills" displaymode="checkbox" multiple="true">
    <SimpleTitle>Skills</SimpleTitle>
    <SimpleFormOption value="Java">Java</SimpleFormOption>
    <SimpleFormOption value="xml">XML</SimpleFormOption>
    <SimpleFormOption value="sql">SQL</SimpleFormOption>
   </SimpleFormSelect>
   <SimpleFormSelect name="nerd" displaymode="checkbox">
    <SimpleTitle>Addicted to Java?</SimpleTitle>
    <SimpleFormOption value="yes">Yes</SimpleFormOption>
    <SimpleFormOption value="no">No</SimpleFormOption>
   </SimpleFormSelect>
   <SimpleFormSelect name="location" displaymode="list">
    <SimpleTitle>Location</SimpleTitle>
    <SimpleFormOption value="Redwood Shores_CA">HQ Redwood
Shores,CA</SimpleFormOption>
    <SimpleFormOption value="Nashua_NH">NEDC Nashua, NH</SimpleFormOption>
    <SimpleFormOption value="SanFrancisco_CA">SanFrancisco, CA</SimpleFormOption>
    <SimpleFormOption value="NewYork,NY">NewYork, NY</SimpleFormOption>
   </SimpleFormSelect>
   </SimpleForm>
  </SimpleContainer>
</SimpleResult>
```

図 8-13 Profile.xml の例の結果



音声の拡張

SimpleGrammar、SimpleValue および SimpleDTMF

SimpleGrammar: SimpleGrammar は、カスタマイズされた音声認識グラマーを提供します。SimpleGrammar の使用方法の詳細は、8-103 ページの「SimpleGrammar」を参照してください。

SimpleValue: SimpleValue は、実行時まで認識されない動的情報のプレースホルダです。この要素は、1 つのデッキ上の複数のカードを処理し、クライアント側データの妥当性チェックを獲得する場合に役立ちます。

SimpleDTMF: これは、入力の処理に使用されるキーボード・バインドです。次の例では、フォーム項目 ZipInput でターゲットに 232 のみが渡され、それ以外は何もしません。

```
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>
<!DOCTYPE SimpleResult PUBLIC "-//ORACLE//DTD SimpleResult 1.1//EN"
"http://xmlns.oracle.com/ias/dtds/SimpleResult_1_1_0.dtd">
<SimpleResult>
 <SimpleContainer>
  <SimpleForm id="Starting" target="test2a.jsp">
   <SimpleFormItem name="addrInput" slot="value">
    simple grammar test, please say oracle or san mateo
   <SimpleGrammar> <grammar root="addr">
   <rule id="addr">
   <one-of>
   <item>Oracle <tag>value = "bridge"</tag></item>
   <item>San Mateo <tag>value = "foster city"</tag></item>
   </one-of>
   </rule>
   </grammar>
   </SimpleGrammar>
   </SimpleFormItem>
   <SimpleFormItem name="zipInput" slot="zip">
    <SimpleDTMF> <grammar root="n5">
    <rule id="n5"> 95 <tag>zip = "232"</tag> </rule>
    </grammar>
    </SimpleDTMF>
     Simple DTMF test, please press 95
   </SimpleFormItem>
  </SimpleForm>
 </SimpleContainer>
</SimpleResult>
```

音声フォームの推奨事項 これまでに、次のような小型画面付きデバイス用のフォームを記述しました。

```
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>
<!DOCTYPE SimpleResult PUBLIC "-//ORACLE//DTD SimpleResult 1.1//EN"
"http://xmlns.oracle.com/ias/dtds/SimpleResult_1_1_0.dtd">
<SimpleResult>
 <SimpleContainer>
  <SimpleForm target="guess.jsp">
   <SimpleFormItem name="guess">
    <SimpleTitle>
    I am thinking of a number between 1 and 100.
    What is your first guess?
    </SimpleTitle>
   </SimpleFormItem>
  </SimpleForm>
 </SimpleContainer>
</SimpleResult>
```

この例は、小型画面付きデバイスで正しく機能します。ただし、音声による入力には不十分です。音声認識が機能するのは、リスニング対象としてきわめて狭い範囲のボキャブラリが規定されている場合のみです。この種のボキャブラリの記述は、音声認識グラマーと呼ばれます。<SimpleMenu> および <SimpleFormSelect> は、<SimpleMenuItem> および <SimpleFormOption> のリストでこの種の構文を提供します。ただし、前述の例では、システムは任意の番号をリスニングする必要があります。これは、<SimpleFormItem> の type 属性で次のように示されます。

```
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>
<!DOCTYPE SimpleResult PUBLIC "-//ORACLE//DTD SimpleResult 1.1//EN"
"http://xmlns.oracle.com/ias/dtds/SimpleResult_1_1_0.dtd">
<SimpleResult>
 <SimpleContainer>
  <SimpleForm target="guess.jsp">
   <SimpleFormItem name="guess" type="number">
    <SimpleTitle>
    I am thinking of a number between 1 and 100.
    What is your first guess?
    </SimpleTitle>
   </SimpleFormItem>
  </SimpleForm>
 </SimpleContainer>
</SimpleResult>
```

type="number" に設定すると、システムは音声で指定された番号に対応する発声をリスニングし、その発声を認識できる場合は対応する番号を識別子 guess に割り当てます。number の他に、値 boolean、digits、date、time、currency および phone でも、リスニング対象の

ボキャブラリを指定できます。開発者は、type 属性を指定するのみでなく、次のガイドラインに従って音声機能を拡張できます。

- <SimpleAudio> 要素を使用して、音声認識を事前に録音されたオーディオで強化できます。
- <SimpleValue> を使用して、認識された発声を確認のためにエコーし、入力が正しく認識されていない場合はユーザーに取り消させることができます。
- 状況依存ヘルプを随時提供します。
- 必要に応じて、deviceclass 属性を使用し、オーディオおよびテキスト・メッセージを音声にあわせて調整します（ただし、この属性を使用するとマークアップが紛らわしくなるため、使用を控えてください）。
- ユーザーに、「取消」を使用して取り消させるのではなく、必要な場所にアクセスする適切なコマンドを提供して、先に進むことでサービスを継続するオプションを随時提供します。
- 場合によっては、認識障害（noinput、nomatch）とインターネットのフェッチ障害（error.badfetch）用に、特殊なイベント・ハンドラを提供します。

次の例では、これらのガイドラインを実装することでユーザー操作を改善しています。

```
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>
<!DOCTYPE SimpleResult PUBLIC "-//ORACLE//DTD SimpleResult 1.1//EN"
"http://xmlns.oracle.com/ias/dtds/SimpleResult_1_1_0.dtd">
<SimpleResult>
 <SimpleContainer>
  <SimpleForm target="tipcalc.jsp">
   <SimpleFormItem name="howmuch" type="currency">How much is the bill?
   </SimpleFormItem>
   <SimpleFormItem name="howmany" format="N*" type="number">
   How many are in your party?
    <SimpleCatch type="cancel">Canceling.
    <SimpleClear>
        <SimpleName name="howmuch"/>
    </SimpleClear>
    </SimpleCatch>
   </SimpleFormItem>
   <SimpleFormSelect name="howbig" deviceclass="microbrowser pdabrowser pcbrowser
micromessenger messenger">
    <SimpleTitle>How big do you want your tip to be?</SimpleTitle>
    <SimpleFormOption value="10">small (10%)</SimpleFormOption>
    <SimpleFormOption value="15">medium (15%)</SimpleFormOption>
    <SimpleFormOption value="20">large (20%)</SimpleFormOption>
   </SimpleFormSelect>
   <SimpleFormSelect name="howbig" deviceclass="voice" autoprompt="false">
    <SimpleTitle>
     How big do you want your tip to be?
```

```
     For 'ten percent' say 'small',
     for 'fifteen percent' say 'medium',
     for 'twenty percent' say 'large'.
    </SimpleTitle>
    <SimpleFormOption value="10">small</SimpleFormOption>
    <SimpleFormOption value="15">medium</SimpleFormOption>
    <SimpleFormOption value="20">large</SimpleFormOption>
    <SimpleCatch type="nomatch">Please say that again</SimpleCatch>
    <SimpleCatch type="cancel">Canceling.
    <SimpleClear>
        <SimpleName name="howmuch"/>
        <SimpleName name="howmany"/>
    </SimpleClear>
    </SimpleCatch>
   </SimpleFormSelect>
  </SimpleForm>
 </SimpleContainer>
</SimpleResult>
```

シグネチャ獲得フォーム・コントロールの使用

Spectrum24 WebClient for Palm Computing Platform など一部のブラウザは、シグネチャを獲得する機能をサポートします。OracleAS Wireless XML を使用して開発されたアプリケーションは、シグネチャ獲得をサポートするために必要なターゲット・マークアップを生成できます。このリリースでは、次のブラウザでシグネチャ獲得がサポートされます。

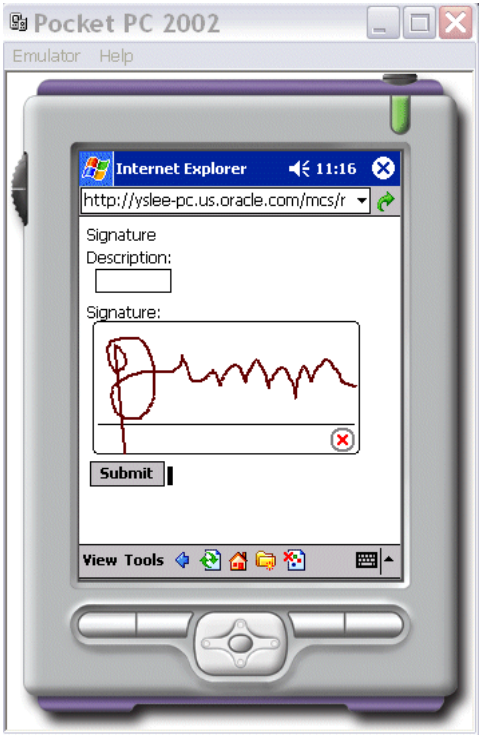
- Palm OS 4.1 用 Symbol Spectrum24 WebClient for Palm Computing Platform バージョン 2.8-10
- Microsoft Pocket PC の Microsoft Pocket Internet Explorer 4.1
- Microsoft CE3 以上の Microsoft Pocket Internet Explorer

サポートされている Microsoft Pocket PC および Windows Mobile プラットフォームでは、Oracle Signature Capture Plug-in for Pocket Internet Explorer がインストールされている必要があります。

SimpleFormItem type=signature シグネチャ獲得機能を持つターゲット・ブラウザでこの機能をサポートするために、OracleAS Wireless XML の <SimpleFormItem> タグに signature タイプが追加されています。次のサンプル・コードは、OracleAS Wireless XML を使用して開発されたアプリケーションでシグネチャ獲得フォーム・コントロールを使用する方法を示します。

```
<SimpleResult>
<SimpleContainer>

    <SimpleForm method="post" target="processForm.jsp" layout="linear">
      <SimpleTitle>Signature</SimpleTitle>

      <SimpleFormItem name="Text" size="6" type="text">
          <SimpleTitle>Description:<SimpleBreak/></SimpleTitle>
      </SimpleFormItem>

      <SimpleFormItem name="capture" type="signature">
        <SimpleTitle>Signature: :<SimpleBreak/></SimpleTitle>
      </SimpleFormItem>
      <SimpleAction name="submit" type="submit" value="submit"/>
    </SimpleForm>

  </SimpleContainer>
</SimpleResult>
```

図 8-14 Pocket PC 2000 Emulator でレンダリングされたシグネチャ獲得コード



シグネチャ獲得コンポーネントのサポート ターゲット・ブラウザがシグネチャ獲得機能をサポートしていない場合でも、シグネチャ獲得コントロール・タグ `<SimpleFormItem type=signature>` を使用する OracleAS Wireless XML ページは機能します。この場合、シグネチャ獲得コントロール・タグは無視されてレンダリングされません。

高度なユーザー相互作用とチャネルの最適化

概要

この項では、Oracle Application Server Wireless に用意されている高度なユーザー相互作用のテクニックについて説明します。これまでに、ユーザーが操作（電話の場合はソフト・キーを押す、音声対応デバイスの場合はコマンドを音声で指定するなど）を実行する場合に、Oracle Application Server Wireless でユーザーにタスクを指定させる方法について説明しました。高度なユーザー相互作用により、デバイスでサポートされている場合は、ユーザーがトリガーしたアクションに応答して多数のタスクを実行できます。また、ユーザーが入力した値に基づいてタスクを実行する機能も、きわめて役立ちます。

Oracle Application Server Wireless には、洗練されたタスクとアクションのバインドを簡単に作成できるように精巧なスキームが用意されています。このスキームは、SimpleText、SimpleForm、SimpleFormItem、SimpleFormSelect、SimpleMenu、SimpleResult または SimpleContainer のコンテキストに使用できる SimpleBind 要素により実行されます。

SimpleBind を使用したイベントとタスク

SimpleBind を使用すると、SimpleMatch 要素の子によって指定された条件に応答して実行されるタスクを指定できます。SimpleMatch では、primary、secondary または continue の各キー、noinput などのイベント、音声または DTMF 構文、フォームまたはフォーム項目（SimpleFinish）に入力する条件、メニュー項目などを指定できます。SimpleTask には 1 つのタスクのみ指定できます。SimpleMatch に指定されたいずれかの条件が発生すると、SimpleTask に指定されたタスクが実行されます。SimpleTask では、SimpleSwitch および SimpleCase 要素を使用してタスクを選択的に実行することもできます。これらの要素は、多くのプログラミング言語のスイッチおよびケース構成メンバーに似ています。

SimpleSwitch の場合は、特定のユーザー入力の値が SimpleCase 要素に列挙された値と比較されます。SimpleTask では、次のタスクを指定できます。

- SimpleGo を使用してリモート・ロケーションにアクセスするタスク
- SimpleTextItem を使用してテキスト項目を表示（または発声）するタスク
- SimpleRefresh を使用してデバイスの画面をリフレッシュするタスク（サポートされている場合）
- SimpleClear および SimpleName を使用して、指定したデバイスのフォーム・フィールドを消去するタスク
- SimpleReprompt を使用して、音声ユーザーに入力プロンプトの再表示を許可するタスク
- SimpleExit を使用してアプリケーションを終了するタスク
- SimpleDisconnect を使用してデバイスを接続状態から切断するタスク（たとえば、音声ゲートウェイを介してサービスにアクセスしているときに電話を切る場合）

- SimplePrev を使用して戻るタスク
- SimpleSubmit および SimpleName を使用してフォーム項目を送信するタスク

SimpleBind 要素のレンダリング特性は、SimpleDisplay 要素で指定します。SimpleDisplay では、レンダリングされて表示された実際の内容を含む子要素として SimpleTextItem がサポートされます。これにより、MenuItem 用のオーディオを再生したりテキストをレンダリングできます。SimpleBindExample.xml の例を参照してください。

SimpleBind.xml

```
<SimpleBind deviceclass="voice microbrowser">
          <SimpleMatch>
            <SimpleFinish/>
            <SimpleGrammar>
              <grammar root="affirmative">
              <rule id="affirmative">
              <one-of>
              <item>yes</item>
              <item>correct</item>
              <item>true</item>
              <item>one</item>
              </one-of>
              </rule>
              </grammar>

            </SimpleGrammar>
            <SimpleDTMF> <grammar root="one">
            <rule id="one">1</rule>
            </grammar>
            </SimpleDTMF>
          <SimpleKey type="primary"/>
          </SimpleMatch>

          <SimpleTask>
            <SimpleSubmit
              target="changepin.jsp"
              name="Submit"
              method="post">
              <SimpleName name="p_old_pin" />
              <SimpleName name="p_new_pin" />
            </SimpleSubmit>
          </SimpleTask>

          <SimpleDisplay>
            <SimpleTextItem deviceclass="voice">
              <SimpleAudio src="sayYesOrPressOne.wav">
                say yes, or press one, to submit
```

```
      </SimpleAudio>
    </SimpleTextItem>

    <SimpleTextItem deviceclass="microbrowser">
      Submit
    </SimpleTextItem>
  </SimpleDisplay>
</SimpleBind>
```

図 8-15 SimpleBind、SimpleMatch および SimpleDisplay の結果

デバイス固有の SimpleBind SimpleBind は、主として音声アプリケーションの作成中に役立ちます。ただし、アプリケーションでは、`deviceclass` 属性を使用し、特定のデバイスに基づいて SimpleBind を使用できます。この属性は、値 pdabrowser、pcbrowser、voice、microbrowser、micromessenger および messenger を取ります。

デバイス・ヘッダーとデバイス・クラス

各項の内容は、次のとおりです。

- Article.jsp
- PageNavigation.Java
- 非同期対応の OracleAS Wireless XML アプリケーション

Oracle Application Server Wireless では、デバイスが次の 2 つの基準に基づいて分類されます。

- デバイスのフォーム要素
- デバイスの通信チャネル（同期リクエスト / レスポンス・モードまたは非同期モード）

開発者は、デバイス固有のプロパティを使用可能にする付加価値サービスを開発できます。たとえば、Oracle Application Server Wireless では、大きいレスポンスのサーバー側管理はサポートされていません。サービスでは、デバイスのレスポンスの最大サイズを使用して、ナビゲーションを動的に提供できます。サポートされているヘッダーは、次のとおりです。

- `X-Oracle-Device.Class`: デバイスのチャネル・モードとフォーム要素を示します。`Device.Class` のそれぞれの値は、一意の通信チャネル・モードと一意のフォーム要素を示します。属性 `deviceclass` の値セットは、ヘッダー `X-Oracle-Device.class` と同じです。`device.class` は、ターゲット・デバイスのマークアップ言語を表すものではないため注意してください。
- `X-Oracle-Device.Orientation`: デバイスのオリエンテーションを示します。アプリケーションでは、このヘッダーを使用して特定デバイスのレンダリング・スタイルを変更できます。可能な値は landscape と portrait です。デフォルト値は portrait です。
- `X-Oracle-Device.MaxDocSize`: デバイスで処理できるコンテンツの最大バイト数の概算値。この近似になるのは、Oracle Application Server Wireless XML のサイズが変換後のデバイス固有のマークアップ言語とは異なる場合があるためです。サービスが MaxDocSize より大きい Oracle Application Server Wireless XML 文書を戻す場合、そのリクエストに対するレスポンスは未指定になります。MaxDocSize によりバインドされるドキュメント・サイズが、デバイスにプッシュできるコンテンツ・サイズになるという保証はありません。deviceclass 用のパラメータの値は、Oracle Application Server Wireless の管理ツールで設定します。デフォルト値は 0 です。
- `X-Oracle.Device.Secure`: Oracle Application Server Wireless Server とデバイス間の接続が、現行のリソース・リクエストの発行時に保護されていたかどうかを示します。可能な値は true または false です。

Article.jsp

次の JSP は、PageNavigation Bean を使用して新規コンテンツを複数のトリップで配信します。

```
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>
<!DOCTYPE SimpleResult PUBLIC "-//ORACLE//DTD SimpleResult 1.1//EN"
"http://xmlns.oracle.com/ias/dtds/SimpleResult_1_1_0.dtd">
<%@ page import="oracle.wireless.xmldevguide.PageNavigation"%>
<%
boolean loopback = Boolean.valueOf(request.getParameter("loopback")).booleanValue();
int pageIndex = 0;
try {
    pageIndex = Integer.parseInt(request.getParameter("pageIndex"));
}
catch(Exception ex){}
%>
<SimpleResult>
<SimpleContainer>
    <jsp:useBean id="contentHandler" class="oracle.wireless.xmldevguide.PageNavigation"
scope="session"/>
    <%
    if(!loopback) {
        String size = request.getHeader("X-Oracle-Device.MaxDocSize");
        if(size != null && !("0".equals(size))) {
           contentHandler.setDeckSize(Integer.parseInt(size));
        }
        pageIndex = 0;
        // get the article content from a source.
        String articleContent = "OracleMobile Online Studio is an online "+
        "developer portal for quickly building, testing and deploying "+
        "wireless applications. It lets any developer, systems integrator "+
        "or independent software vendor quickly develop a mobile application "+
        "that is immediately accessible from all devices. This unique, next "+
        "generation environment allows companies to benefit from faster time "+
        "to market, increased productivity, and a dramatically simplified "+
        "testing cycle, while providing access to the latest mobile applications "+
        "and tools. It enables you to focus on your business logic which is your "+
        "core competency, while we focus on the device complexity, our core "+
        "competency. <SimpleBreak/><SimpleBreak/>"+
        "OracleMobile Online Studio's build, test, and deploy model is new and "+
        "unique to software development. It presents a hosted approach to developing "+
        "dynamic content. You do not need to download any software or tools to start "+
        "using it. All you need to do is access the OracleMobile Online Studio, "+
        "register, and login. Once authenticated, you will have access to "+
        "reusable modules, examples, documentation, runtime information, and other "+
        "useful resources.  <SimpleBreak/><SimpleBreak/>"+
        "Now you can even use OracleMobile Online Studio to write a single application "+
        "that can be accessed via both wireless and voice interfaces. Listen to your "+
        "OracleMobile Online Studio applications by calling: "+
```

```
        "888-226-4854. <SimpleBreak/><SimpleBreak/>"+
        "Simplify the development of your OracleMobile Online Studio application "+
        "with Where2Net's daVinci Studio.";

         contentHandler.setContent(articleContent);
    }
    String nextURL = null;
    String previousURL = null;
    int numPages = contentHandler.getAvailablePages();
    if(numPages > 1) {
        nextURL = (pageIndex < numPages - 1) ?
"article.jsp?loopback=true&amp;pageIndex="+(pageIndex + 1) : null;
        previousURL = (pageIndex > 0) ? "article.jsp?loopback=true&amp;pageIndex="+
(pageIndex - 1) : null;
    }
    String articleTitle = (pageIndex == 0) ? "OracleMobile online studio" : "contd...";
    %>
    <SimpleText>
     <SimpleTitle><%=articleTitle%></SimpleTitle>
    <%
        String s = (nextURL == null) ? "articleIndex.jsp" : nextURL;
        if(pageIndex != numPages - 1) {
            %>
            <SimpleAction type="primary2" label="Close" target="articleIndex.jsp"/>
            <SimpleAction type="primary1" label="Next" target="<%=s%>"/>
            <%
        }
        else {
            %>
            <SimpleAction type="primary1" label="Close" target="<%=s%>"/>
            <%
        }
        %>
        <SimpleTextItem><%=contentHandler.getPage(pageIndex)%></SimpleTextItem>
        <%
        if(previousURL != null) {
            %>
            <SimpleTextItem><SimpleHref
target="<%=previousURL%>">Previous</SimpleHref></SimpleTextItem>
            <%
            }
            if(nextURL != null){
                %>
                <SimpleTextItem><SimpleHref
target="<%=nextURL%>">Next</SimpleHref></SimpleTextItem>
                <%
            }
            %>
    </SimpleText>
    </SimpleContainer>
 </SimpleResult>
```

PageNavigation.Java

```
package oracle.wireless.xmldevguide;

import Java.io.StringReader;
import Java.io.StringWriter;
import Java.io.Serializable;
import Java.io.IOException;

import Java.util.ArrayList;

/**
 * The bean breaks a text content into mutiple deck of a defined size. Content
 * deck do not include any formatting information of the content which should
 * be provided by the content view.
 *
 * @author Chandra Patni
 * @version 1.0
 */
public class PageNavigation implements Serializable {

    /**
     * To keep the location of a page
     */
     private class Page {
        /**
         * starting index of the page, inclusive of start
         */
        public int start;
        /**
         * end index of the page, exclusive
         */
        public int end;
        /**
         * returns the length of the page
         */
        public int length() {
            return end - start;
        }

        /**
         * retruns the content of the page
         */
        public String toString() {
            return content.substring(start, end);
        }
     }
```

```
/**
 * Default size of a deck in characters. The actual deck size will be adjusted
 * so that a word is not split. However, an orphan, end of paragraph etc
 * conditions are not checked for.
 */
public static final int DECK_SIZE = 900;

/**
 * size of a deck. default value is 900 chars
 */
private int deckSize = DECK_SIZE;

/**
 * Sets the size of one deck. Should be called before setContent()
 */
public void setDeckSize(int value) {
    deckSize = value;
}

/**
 * Returns the size of one deck.
 */
public int getDeckSize() {
    return deckSize;
}

/**
 * Conent to be decked
 */
private String content;

/**
 * Pages in the content
 */
private Page pages[];

/**
 * The total number of pages by the content
 */
private int totalPages;

/**
 * Default constructor
 */
public PageNavigation() {
}
```

```
/**
 * Default constructor
 */
public PageNavigation(String content) {
    setContent(content);
}

/**
 * get the page content at the given index
 */
public String getPage(int index) {
    return pages[index].toString();
}

/**
 *  Returns the total number of pages
 */
public int getAvailablePages() {
    if(pages == null) return 0;
    return pages.length;
}

/**
 * initializes the bean
 */
private void init() {
    // get the rough estimate of pages
    totalPages = content.length() / deckSize + 1;
    // initialize the array
    int lastIndex = 0;
    ArrayList list = new ArrayList(totalPages);
    Page p = null;
    while((p = getNextPage(lastIndex)) != null) {
        list.add(p);
        lastIndex = p.end;
    }
    pages = (Page []) list.toArray(new Page[list.size()]);
}

private Page getNextPage(int lastIndex) {
    if(lastIndex >= content.length()) return null;
    char c = content.charAt(lastIndex);
    while(Character. isWhitespace(c)) {
        if(++lastIndex >= content.length()) return null;
        c = content.charAt(lastIndex);
    }
```

```
            Page p = new Page();
            p.start = lastIndex;
            // again look for whitespaces while trimming the content.
            p.end = p.start + deckSize;
            if(p.end >= content.length()) {
                p.end = content.length();
                return p;
            }
            // if not then we need to figure out the previous white space
            do {
                c = content.charAt(p.end);
                if(Character. isWhitespace(c)) {
                    return p;
                }
                p.end--;
                if(p.end == 0) {
                    p.end = p.start + deckSize;
                    return p;
                }
            }while(true);
        }

        /**
         * sets the content to the specified value. default MIME type is text/plain
         */
        public void setContent(String s) {
            content = s;
            init();
        }
    }
```

非同期対応の OracleAS Wireless XML アプリケーション

概要

開発者は非同期デバイス用に異なる論理フロー（異なる方法による結果のレンダリングなど）を選択できます。この場合は、リクエストが非同期デバイス・クラスからのものかどうかを認識できる必要があります。そのためには、ユーザー・リクエストのデバイス・クラス属性をチェックします。詳細は、第 10 章「メッセージ・アプリケーションの作成」を参照してください。

非同期デバイスからのリクエストの場合、`deviceclass` 属性値は messenger または micromessenger です。アダプタ形式で作成されたサービスの場合は入力引数から、HTTP または OC4J アダプタに準拠するサービスの場合は HTTP ヘッダーから、情報を取得できます。ServiceContext に定義されている入力引数 `_DeviceCategory` では、アダプタ形式のサービスのデバイス・クラス値を指定します。HTTP または OC4J ベースのサービスの場合は、HTTP ヘッダー `x-oracle-device.class` を介して値を取り出すことができます。

同様に、アプリケーションによって作成される非同期固有の OracleAS Wireless XML の結果の任意のセクションで、値 messenger または micromessenger が要素の属性 `deviceclass` にバインドされます。非同期デバイスでは、すべてのデバイスに共通する要素（`deviceclass` 属性が指定されていない）や、属性に値 messenger または micromessenger を含んでいる要素が処理されます。

すべての OracleAS Wireless XML サービスは、技術的な観点から非同期対応にすることができます。アプリケーションの作成方法を決定する場合や、既存のアプリケーションを非同期対応にする場合には、非同期デバイスを使用中のユーザー操作を考慮する必要があります。これは、タイプ（画面サイズやフォーム要素など）の異なるデバイスにアプリケーションをレンダリングする方法を決定する際に適用したのと同じ作業です。非同期デバイスではプレーン・テキストの入力を使用するクライアントが想定されるため、ユーザー操作の考慮がさらに必要になります。多数のユーザー相互作用を想定するサービスや、複雑な UI を伴うサービスは、適切に機能しない場合があります。

また、一部の OracleAS Wireless XML タグは非同期デバイスで適切ではなく、非同期デバイスにおける XML タグ・セットの特定のセマンティクスを把握する必要があります。非同期デバイスではクライアント側の参照機能は想定されないため、通常、デバイス上で特定のキーまたは操作を想定するタグは非同期デバイスには不適切です。次の表に、特定の非同期

セマンティクスを持つタグを示します。この表には、他のデバイス・チャネルと解析機能を共有するタグは含まれていません。

表 8-6 OracleAS Wireless XML タグのセマンティクスの概要

OracleAS Wireless XML タグ	セマンティクス
SimpleAction	SimpleMenuItem および SimpleHref と同じに扱われます。各 SimpleMenuItem、SimpleHref または SimpleAction には、非同期ユーザーが選択できるように、デバイス結果で接頭辞として番号が使用されます。
SimpleAudio	無視 : 非同期デバイスには適用されません。
SimpleBind	無視 : 非同期デバイスには適用されません。
SimpleBreak	改行を出力します。
SimpleCache	無視 : 非同期デバイスには適用されません。
SimpleCase	無視 : 非同期デバイスには適用されません。
SimpleCatch	無視 : 非同期デバイスには適用されません。
SimpleCol	テキストを出力します。
SimpleDisconnect	無視 : 非同期デバイスには適用されません。
SimpleDisplay	無視 : 非同期デバイスには適用されません。
SimpleDTMF	無視 : 非同期デバイスには適用されません。
SimpleEM	テキストを出力します。
SimpleEvent	無視 : 非同期デバイスには適用されません。
SimpleExit	無視 : 非同期デバイスには適用されません。
SimpleFinish	無視 : 非同期デバイスには適用されません。
SimpleFooter	無視 : 非同期デバイスには適用されません。
SimpleForm	ユーザーから発行されるパラメータを対応するキーとペアにできるように、サーバー側でフォーム状態がメンテナンスされます。
SimpleFormItem	戻されるページに項目テキストが出力されます。ユーザーは、ページに表示される項目と同じ順序で対応する項目値を入力します。
SimpleFormOption	戻されるメッセージにフォーム・オプションのリストが出力され、各フォーム・オプションには接頭辞として番号が追加されます。ユーザーは、接頭辞番号またはオプションのテキストを使用して選択項目を入力できます。たとえば、選択項目「州」にオプション「1 AL、2 CA、3 UT...」が含まれているとします。ユーザーは値「2」または「CA」を指定してオプション「CA」を選択できます。

表 8-6 OracleAS Wireless XML タグのセマンティクスの概要（続き）

OracleAS Wireless XML タグ	セマンティクス
SimpleFormSelect	テキストを出力します。
SimpleGo	無視 : 非同期デバイスには適用されません。
SimpleGrammar	無視 : 非同期デバイスには適用されません。
SimpleHeader	テキストを出力します。
SimpleHelp	テキストを出力します。
SimpleHref	SimpleMenuItem と同じに扱われます。ユーザーが対応する番号をレスポンスに使用して項目を選択できるように、すべての SimpleMenuItem には接頭辞として番号が追加されます。
SimpleImage	無視 : 非同期デバイスには適用されません。
SimpleKey	無視 : 非同期デバイスには適用されません。
SimpleMatch	無視 : 非同期デバイスには適用されません。
SimpleMenu	ページに改行が挿入されます。メニュー状態はサーバー側でメンテナンスされます。
SimpleMenuItem	戻されるページにメニュー項目の値が出力され、メニュー項目に識別用の番号接頭辞が追加されます。ターゲット URL と番号接頭辞はサーバーに格納されるため、ユーザーは選択後に URL を取り出すことができます。
SimpleMenuItemField	テキストを出力します。
SimpleMItem	無視 : 非同期デバイスには適用されません。
SimpleName	無視 : 非同期デバイスには適用されません。
SimpleOptGroup	無視 : 非同期デバイスには適用されません。
SimplePrev	無視 : 非同期デバイスには適用されません。
SimpleProperty	無視 : 非同期デバイスには適用されません。
SimpleRefresh	無視 : 非同期デバイスには適用されません。
SimpleReprompt	無視 : 非同期デバイスには適用されません。
SimpleRow	改行を出力します。
SimpleSpeech	無視 : 非同期デバイスには適用されません。
SimpleStrong	テキストを出力します。
SimpleTableBody	改行を出力します。

表 8-6 OracleAS Wireless XML タグのセマンティクスの概要（続き）

OracleAS Wireless XML タグ	セマンティクス
SimpleTableHeader	改行を出力します。
SimpleTask	無視 : 非同期デバイスには適用されません。
SimpleText	改行を出力します。
SimpleTextItem	テキストを出力します。
SimpleTimer	無視 : 非同期デバイスには適用されません。
SimpleTitle	テキストを出力します。
SimpleValue	無視 : 非同期デバイスには適用されません。

9

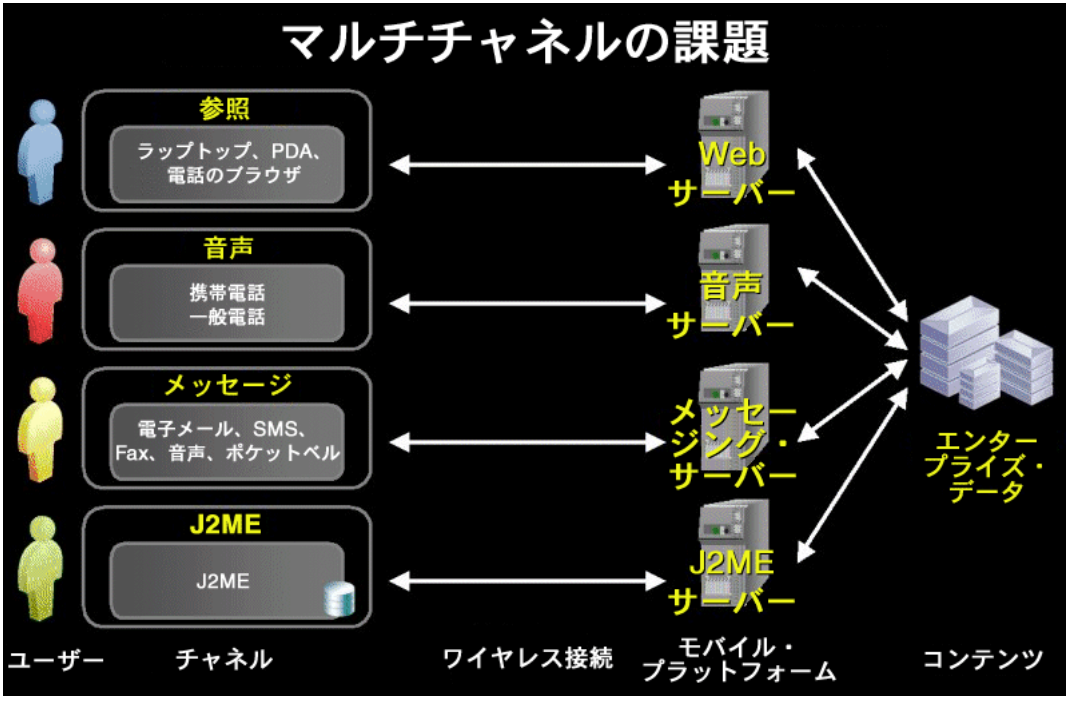
マルチチャネル・サーバーの使用

項ごとに様々なトピックを記載しています。各項の内容は、次のとおりです。

概要

進化し続けてきたインターネットは、ここ数年でさらに大きく進歩しました。従来のユーザーは、ポピュラーな 2 つのブラウザのいずれかを使用して、デスクトップ・パーソナル・コンピュータからインターネットにアクセスするのが一般的でした。しかし、これは大きく様変わりしました。今日、インターネットには、スマートフォン、携帯情報端末（PDA）、ポケットベル、一般電話、自動車、さらには冷蔵庫などの家電製品からもアクセスできるようになりました。このことは、企業がサービスを提供する機会が大きく広がっていることを示します。しかし、これはまた、開発者にとっては大きな課題となります。従来は、唯一のマークアップ言語である HTML を使用してアプリケーションを開発し、そのアプリケーションにアクセスする 2 つのブラウザを提供するのみでした。今日では、ほとんどすべてのチャネルで独自のマークアップ言語が使用され、ブラウザは機能が多様化して同じマークアップ言語でも異なるバージョンがサポートされています。たとえば、ほとんどのスマートフォンは WML をサポートしています。しかし、機種によってサポートしている WML のバージョンが異なります。同じバージョンの WML をサポートしている場合でも、標準とはいくらか違いがある場合があります。電話機では、一般的に VoiceXML が使用されています。この場合も、異なるバージョンの VoiceXML が様々な音声ゲートウェイでサポートされています。開発者は、このような多様なチャネルに対して、どのように Web アプリケーションを作成すればよいでしょうか。チャネル固有のマークアップ言語を使用しているチャネルごとにアプリケーションを作成するのも 1 つの方法です。しかし、この方法には、膨大な時間とコストが必要です。

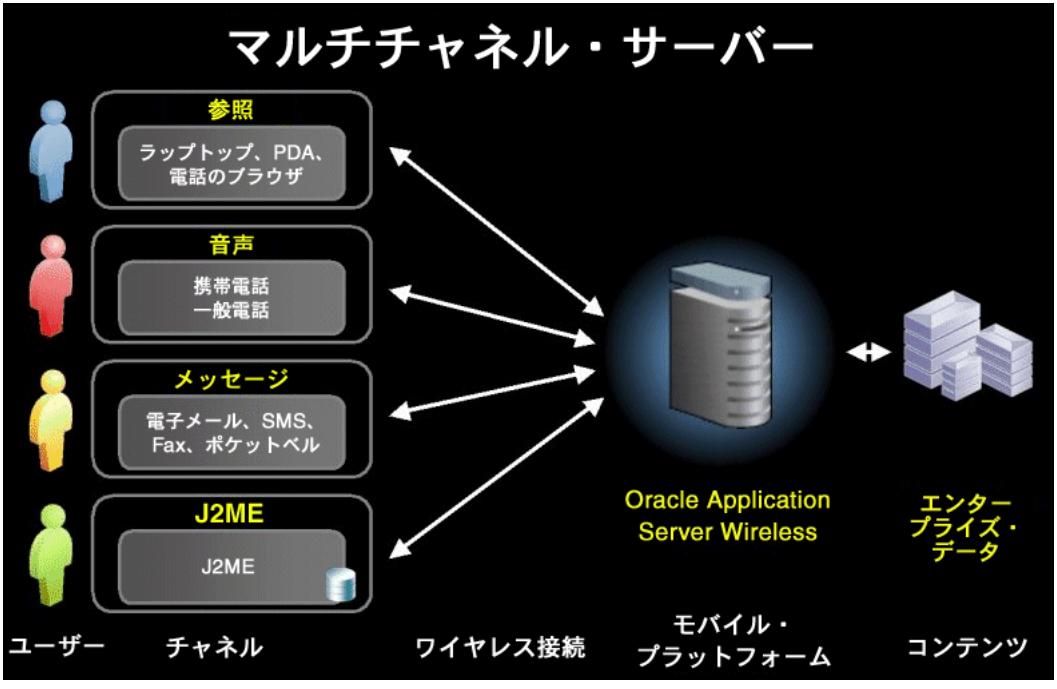
図 9-1 マルチチャネルの課題



マルチチャネルの利点

オラクル社は、マルチチャネルの課題に真剣に取り組み、開発者や企業がすべてのチャネルに対応できる 1 つのアプリケーションを簡単に開発できるように、新しいソリューションの開発を目指しました。OracleAS Wireless マルチチャネル・サーバー（MCS）は、開発者をすべてのチャネルにおける複雑さから解放します。MCS は、Web アプリケーションにアクセスする単一のブラウザとして機能します。

図 9-2 マルチチャネルによるソリューション



マルチチャネル・サーバーでは、アプリケーション開発用に、デバイスに依存しない次の 3 つのマークアップ言語がサポートされます。

- XHTML+XForms: XForms（http://www.w3.org/TR/xforms/）をサポートする XHTML 1.0（http://www.w3.org/TR/xhtml1/）標準マークアップ言語
- XHTML Mobile Profile（XHTML MP）: Open Mobile Alliance 社（http://www.openmobilealliance.org/）によって定義された標準マークアップ言語（http://www1.wapforum.org/tech/documents/WAP-277-XHTMLMP-20011029-a.pdf）
- OracleAS Wireless XML: OracleAS Wireless によって定義されたマークアップ言語

開発者によっては、マルチチャネルは念頭になく、たとえば音声チャネルでのみアプリケーションを公開する場合があります。このような場合でも、MCS を使用する理由は何でしょうか。MCS を使用する理由には、少なくとも次の 2 点があげられます。

- VoiceXML ではなく、XHTML を使用してアプリケーションを開発できます。ほとんどの Web 開発者は XHTML の知識があり、新たに VoiceXML の知識は必要ありません。これによって、時間とコストが節約できるのは明白です。
- 様々な音声ゲートウェイで、異なるバージョンの VoiceXML がサポートされています。このため、システム固有の VoiceXML を使用してアプリケーションを開発した場合は、そのアプリケーションが、他のすべての音声ゲートウェイで動作することを保証する必要があります。MCS を使用すると、アプリケーションへの変更なしに、多くの音声ゲートウェイでアプリケーションが動作することが保証されます。

現在は 1 つのチャネルでのみアプリケーションを公開する予定でも、将来、別のチャネルからもそのアプリケーションにアクセス可能にすることが有益になる場合があります。その場合、MCS は新しいチャネルもサポートするため、アプリケーションを変更する必要はありません。

マルチチャネル・サーバーの機能

次に、マルチチャネル・サーバーの主な機能について説明します。

- マルチチャネル・コンテンツ調整 : デバイスに依存しない同じコンテンツを、異なるデバイス上の異なるチャネルに配信します。マルチチャネル・サーバーは、現行のユーザー・デバイス機能に基づいてアプリケーションのコンテンツを調整します。また、デバイス固有のマークアップ言語、画面サイズ、ネットワーク速度などにも対応します。
- 標準マークアップ言語のサポート : 標準マークアップ言語を使用してアプリケーションを開発します。これによって、開発者は新しいマークアップ言語を学習する必要がないため、開発の時間とコストを節約できます。アプリケーションが 1 つのチャネル（たとえば、音声チャネル）に配信される場合でも、開発者は新しいチャネル固有のマークアップ言語を学習する必要はありません。開発者は、非常にポピュラーな XHTML を使用しながら、作成したコンテンツを VoiceXML、WML などのデバイスに配信できます。
- デバイスとゲートウェイでの動作保証 : 市場には多数のデバイスやゲートウェイが出回っているため、それらすべてのデバイスやゲートウェイ上でアプリケーションが動作することをアプリケーション開発者が保証するのは困難です。PC の主要なブラウザは 2 つのみですが、ほとんどの Web アプリケーションは、そのいずれかのブラウザ上で最適に動作するためにも調整が必要になります。各種の新しいモバイル・デバイスの場合は、開発者がそのすべてのデバイスでアプリケーションが機能することを保証するのは不可能です。マルチチャネル・サーバーは、すべてのデバイスとの互換性を保証します。
- デバイス検出 : マルチチャネル・サーバーは、高度なアルゴリズムを使用して、リクエストが出されたデバイスを検出します。デバイス検出では、User-Agent、Accept などの HTTP ヘッダーが照合されます。一部のデバイス機能（比較的新しいデバイスの場合）は、デバイスによってリクエストとともに送信され、コンテンツの最適な調整用に使用されます。

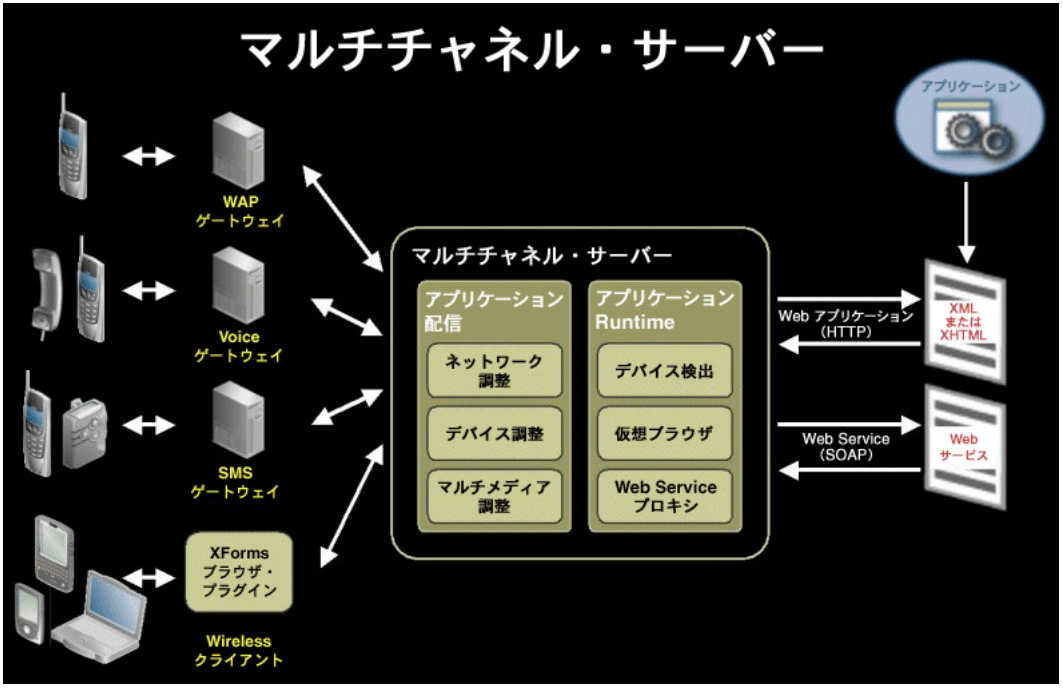
- マルチメディア調整 : マルチチャネル・サーバーは、テキスト調整とともに、イメージ、着信音、音声構文およびオーディオ / ビデオ・ストリームのデバイス固有の調整も行います。
- 単一のブラウザ : 複数のアプリケーションが相互に対話するブラウザは、マルチチャネル・サーバーのみです。これによって、アプリケーション開発者は、エンド・ユーザーのデバイスで発生する不具合から解放されます。たとえば、ほとんどのモバイル・デバイスは HTTP Cookie をサポートしていません。しかし、ユーザー・セッションの維持には、Cookie を使用するのが最も簡単な方法です。マルチチャネル・サーバーは、エンド・ユーザーのデバイスにかわって、セッションと他の Cookie を処理します。
- XForms エンジン : マルチチャネル・サーバーによって、ユーザー・デバイスに対する XForms（http://www.w3.org/TR/xforms/）のサポートが強化されます。
- 既存のポータルとの併用 : マルチチャネル・サーバーをポータル・サーバーの手前にデプロイすることによって、既存のポータルを介してデバイスにアクセスできます。
- URL のキャッシュ : デバイスによっては、メモリー量が非常に制限されている場合があり、デバイスで受信できるコンテンツが制限されます。マルチチャネル・サーバーでは、デバイスに送信する長い URL をキャッシュすることで、コンテンツのサイズを減らすことができます。

Oracle Application Server には、マルチチャネル・サーバー以外に、MCS に基づいて作成された Wireless and Voice Portal が用意されています。Wireless and Voice Portal によって、次の機能が追加されます。

- ポータル : Wireless and Voice Portal には、ユーザー管理やサービス管理、ACL など、完全なポータル機能が用意されています。ユーザーはこのポータルを使用して、各サービスに対する起動パッドとなるカスタマイズ可能な独自のホーム・ページを作成できます。使用可能なサービスには、データベース情報、パーソナライズ、アラート、ロケーション・サービスなど、様々な形態があります。コンテンツ・ソースが膨大な数であるために、管理しやすい方法を使用して、各種デバイスに各種アプリケーションを最適の方法で配信することは、さらに複雑になります。
- ネットワーク調整 : Wireless and Voice Portal は、HTTP プロトコル以外に各種プロトコルもサポートしているため、HTTP 以外のデバイスからもアクセス可能です。ネットワーク調整は、拡張可能なフレームワークに基づいて行われます。このフレームワークによって、顧客は自分のドライバにプラグインしてネットワーク・プロトコルを調整できます。
- J2ME WebServices: Wireless and Voice Portal では WebServices プロキシが提供されるため、J2ME 対応デバイスは WebServices にアクセスできます。これによって、WebServices として公開された外部アプリケーションにアクセスできるため、MIDlet 開発者はアプリケーションを拡張できます。

次の図に、マルチチャネル・サーバーの主なコンポーネントを示します。

図 9-3 マルチチャネル・サーバーのコンポーネント



マルチチャネル・サーバーは、拡張可能なモバイル・アプリケーションのプラットフォームです。これは、交換可能な複数のモジュールで構成されています。モジュール実装を交換することによって、デフォルト動作を変更できます。すべてのモジュールとそのデフォルト実装について、次に説明します。

マルチメディア調整

概要

OracleAS Wireless のマルチメディア調整サービスでは、イメージ、着信音、音声構文およびオーディオ / ビデオの各配信に関するデバイス固有の調整が提供されます。このサービスは、コアのマルチチャネル・サーバーの中枢部分です。

デバイスやそのデバイス上のブラウザによって、サポートされるイメージ・フォーマット、および画面サイズや色の深度が異なります。リクエストに応答して、OracleAS Wireless が実行するコンテンツ調整の一部には、クライアント・デバイスにあわせたイメージの動的な調整があります。さらに、新しい Intelligent Messaging コンポーネントは、イメージ調整サービスを使用して、EMS および MMS 用のイメージを変換します。

Java API として提供される着信音調整では、着信音データを、最も一般的な電話でサポートされているフォーマットに変換できます。サポートされているフォーマットには、RTTL、iMelody および MIDI が含まれます。着信音調整のフレームワークによって、開発者は、着信音の新規フォーマット用のサポートを簡単に追加できます。Intelligent Messaging コンポーネントは、着信音調整サービスを使用して、着信音を変換し、SMS、EMS および MMS を介して配信します。

さらに、マルチメディア調整サービスによって、音声ゲートウェイ・ベンダーは OracleAS Wireless のプラットフォームを拡張して新規またはベンダー固有の構文フォーマットをサポートできるため、OracleAS Wireless の音声サポートが拡張されます。このような構文フォーマットは、（`SimpleGrammar` タグを介して）音声ブラウザでサポートされるフォーマットに変換されます。OracleAS Wireless XML に定義されている構文はインラインとみなされ、URL 参照によって提供される構文は外部とみなされます。新しい構文フォーマットをサポートする必要がある音声ゲートウェイ・ベンダーは、OracleAS Wireless XML 構文を自社の構文フォーマットに変換するための XSL スタイルシートを簡単に提供できます。インライン構文変換は音声トランスフォーマで直接サポートされます。また、オラクル社は、関連する XSL スタイルシートを使用して、外部の音声構文変換用のフレームワークを提供します。

イメージ調整機能

イメージ調整の重要な機能は、次のとおりです。

- 複数の入力イメージ・フォーマットをサポートします。
  - ファイル形式 : BMP、GIF、JFIF、PNG、TIFF、WBMP
  - コンテンツ・フォーマット : MONOCHROME、1BIT、2BIT、4BIT、8BIT、12BIT、16BIT、24BIT、32BIT、48BIT、LUT、DRCT、RGB、GRAY
  - 圧縮形式 : JPEG、BMPRLE、LZW、LZWHDIFF、FAX3、FAX4、HUFFMAN3、PACKBITS、GIFLZW、DEFLATE
- 複数の出力イメージ・コンテンツ・フォーマットをサポートします。これには、異なる色の深度、圧縮形式、色スキームおよびファイル形式が含まれます。
  - ファイル形式 : JFIF（JPEG）、GIF、BMP、WBMP、PNG
  - 圧縮形式 : JPEG、GIFLZW、BMPRLE、DEFLATE
  - コンテンツ・フォーマット : MONOCHROME、2BITLUTGRAY、4BITLUTGRAY、8BITLUTGRAY、8BITLUTRGB、24BITRGB、8BITGRAY
- イメージのスケール変更とサイズ変更をサポートします。
  - 固定ディメンションにスケール変更します。
  - アスペクト比を維持し、枠ボックスに収まるようにスケール変更します （元のイメージのディメンションが対象のイメージのディメンションより小さい場合は、元のイメージのディメンションを使用して枠ボックスが定義されます)。
  - デバイス / ネットワークで設定されたサイズ制限を順守するために、イメージ・データのサイズをバイト単位で縮小します。
  - 前述のイメージ処理機能は、OracleAS Wireless の J2EE 準拠のコンポーネントとして提供されます。
- URL の長さ制限をサポートします。
- インバウンドとアウトバウンドのイメージのキャッシュをサポートします。インバウンド・キャッシュとは、元のイメージを（Web Cache を使用して）中間層にキャッシュすることを意味します。これによって、異なるタイプの複数のデバイスから同じイメージがリクエストされた場合でも、元のイメージのフェッチは 1 回で済みます。アウトバウンド・キャッシュとは、調整済イメージを（Web Cache を使用して）キャッシュすることを意味します。これによって、類似のデバイス上で複数のユーザーが同じ調整済イメージを共有できます。

注意： イメージのキャッシュ・ポリシーは、元のイメージの所有者が決定します。MCS はキャッシュ・ヘッダーを Web Cache に渡すため、特定のイメージがキャッシュ不能の場合でも、Web Cache はそのイメージをキャッシュしません。

イメージを使用したマルチチャネル・アプリケーションの作成

イメージ調整を使用し、XHTML と Wireless XML のそれぞれでアプリケーションを開発する方法の詳細は、第 3 章「OracleAS Wireless Developer Kit」を参照してください。

コマンドライン・ツール

イメージを Web または Wireless アプリケーションにデプロイする前にバッチ・モードで変換するために、コマンドライン・ツールが用意されています。

- 名前 : `ImageGenerate.{bat|sh}`
- 説明 : Java アプリケーションを起動し、各種デバイスでサポートされるあらゆるフォーマットで、指定の入力イメージからイメージを生成するシェルまたは UNIX シェル・スクリプト。バッチ・ファイルやスクリプト・ファイルは、{IAS_HOME}¥wireless¥bin¥ImageGenerate.bat（Windows の場合）または {IAS_HOME}/wireless/bin/ImageGenerate.sh（UNIX の場合）に格納されています。
- 使用方法 :

```
ImageGenerate.{bat|sh} -inFile filename -outFile filename [-outW width] [-outH
height] -outFormat format [-outContent contentType] [-dataSizeLimit limit]
[-maintainRatio {true|false}]
```

- パラメータ
  - inFile filename: 処理する入力ファイルのファイル名。必須の引数です。
  - outFile filename: 処理後の結果ファイル名。必須の引数です。
  - outW width: 結果イメージの幅（ピクセル単位）。オプションの引数です。
  - outFormat format: 結果イメージの出力ファイル形式。

    例については、『Oracle *inter*Media ユーザーズ・ガイド』（http://www.otn.oracle.com/docs/products/oracle9i/doc_library/901_doc/nav/docindex.htm）で完全な仕様を参照してください。

    次の例が含まれています。GIFF: Gif フォーマット、JFIF: jpg フォーマット、WBMP: wbmp フォーマット、PNGF: png フォーマット。必須の引数です。

- outContent contentType: 結果イメージのコンテンツ・フォーマット。

  たとえば、MONOCHROME を指定すると、イメージが白黒表現に変更されます。GIF イメージの場合は、4BITLUT を指定すると、4 ビット（16 色）表現に変更されます。完全な仕様は、『Oracle *inter*Media ユーザーズ・ガイド』を参照してください。オプションの引数です。

- dataSizeLimit limit: GIF イメージ用。指定のサイズにイメージを収めるために、ピクセルの色数を減らし（最終的には白黒表現まで）、必要に応じてイメージ・サイズを縮小します。オプションの引数です。

- maintainRatio {true | false}: イメージのアスペクト比を維持します。outW と outH のサイズで囲まれたボックスにイメージを収めます。デフォルトは true です。オプションの引数です。

  例 :

```
ImageGenerate -inFile stock_600_450.jpg -outFile stock_240_180.gif -outW 240
-outH 180 -outContent monochrome -outFormat giff
```

注意： Java_HOME または ORACLE_HOME のいずれかが環境変数として定義されていることを確認してください。

ImageProcessor API を使用した拡張

説明

ImageProcessor インタフェースは、process という名前のメソッドで構成されています。このメソッドは、指定のイメージ URL を、コール側デバイスでの表示に適したイメージにリンクしている別の URL にリライトします。デバイス情報は、コール側デバイスに対応する oracle.panama.model.Device インスタンスにハンドルを渡すことによって提供されます。通常、リライトされた URL は、入力イメージ URL とデバイス特性に基づいて調整済イメージを動的に生成できるサーバーを指し示します。

インタフェース oracle.panama.multimedia.ImageProcessor

```
package oracle.panama.multimedia;

import oracle.panama.model.Device;

/** Use this interface to replace the existing Image processing
 *  implementation for all formats with your own implementation
 */
public interface ImageProcessor {
    ImageResponse process(ImageRequest request, Device) throws MultimediaException;
}
```

oracle.panama.multimedia.ImageRequest クラスと ImageResponse クラスは、入力イメージと対象の出力イメージに関する情報を獲得します。詳細は、oracle.panama.multimedia の Javadoc を参照してください。

実装

この oracle.panama.multimedia.impl.ImageProcessorImpl インタフェースのデフォルト実装によって、マルチメディア調整サービスのイメージ調整サーブレットを指し示すように URL がリライトされます。このサーブレットは、元のイメージをフェッチして動的に処理し、調整済イメージをリクエスト側デバイスに戻します。

構成

ImageProcessor インタフェースのデフォルト実装を変更する場合は、OracleAS Wireless Tools を使用して、実装クラスの名前を指定する必要があります。

1. OracleAS Wireless Tools に管理者でログインします。
2. 「システム」フォルダにある**「サイト管理」**をクリックします。
3. 「コンポーネント構成」の下にある**「マルチメディア適応サービス」**をクリックします。

4. 「イメージ・プロバイダ・クラス名」フィールドの値を実装クラス名に変更し、「**OK**」をクリックします。
5. 最後に、実装クラスが OracleAS Wireless のクラスパスに含まれていることを確認します。

着信音調整

機能

着信音調整を使用して、特定の入力フォーマットで指定された着信音を、デバイスでサポートされているフォーマットに変換します。この変換は、XMS メッセージ・アプリケーションで自動的に実行されます。ただし、開発者が着信音調整を制御できるように、Java API が用意されています。

サポートされている変換は、次のとおりです。

- RTTTL から Nokia バイナリ、IMelody、MIDI
- IMelody から Nokia バイナリ、RTTTL、MIDI

RingtoneProcessor Java API

RingtoneProcessor インタフェースは、指定の着信音を出力デバイスで要求されるフォーマットにリライトする、process という名前のメソッドで構成されています。process メソッドは、RingtoneRequest インスタンスを受け取り、変換された着信音を出力として RingtoneResponse インスタンスに戻します。このインタフェースは、メッセージ・フレームワークによって起動し、直接使用することもできます。

インタフェース oracle.panama.multimedia.RingtoneProcessor

```
package oracle.panama.multimedia;

/** Use this interface to replace the existing Ringtone processing
 *  implementation for all formats with your own implementation
 */
public interface RingtoneProcessor {
    RingtoneResponse process(RingtoneRequest request) throws MultimediaException;
}
```

クラス oracle.panama.multimedia.RingtoneRequest

```
package oracle.panama.multimedia;
import Java.io.InputStream;
public class RingtoneRequest {
        /** Ringtone input data types */
        public static int RINGTONE_DATA_STRING = 0;
        public static int RINGTONE_DATA_STREAM = 1;
```

```
        public static int RINGTONE_DATA_BYTES = 2;
        public static int RINGTONE_DATA_NULL = -1;
        public String inputFormat = null;
        public String outputFormat = null;
    public String inputMimeType = null;
    public String outputMimeType = null;
    private String dataString = null;
        private InputStream dataStream = null;
        private byte[] dataBytes = null;
        private int dataType = RINGTONE_DATA_NULL;
        public RingtoneRequest()  {
        }

        public void setData (String ringtone) {
        this.dataString = ringtone;
        this.dataType = RINGTONE_DATA_STRING;
        }

        public void setData (InputStream ringtone) {
        this.dataStream = ringtone;
        this.dataType = RINGTONE_DATA_STREAM;
        }

        public void setData (byte[] ringtone) {
        this.dataBytes = ringtone;
        this.dataType = RINGTONE_DATA_BYTES;
        }

        public String getDataAsString () {
        return this.dataString;
        }

        public InputStream getDataAsStream() {
        return this.dataStream;
        }

        public byte[] getDataAsBytes() {
        return this.dataBytes;
        }

        /** note, no setDataType method is provided to prevent
         *  inconsistency. The dataType attribute is set when
         *  setting data.
         */
        public int getDataType() {
        return this.dataType;
        }
}
```

クラス oracle.panama.multimedia.RingtoneResponse

```
package oracle.panama.multimedia;

    import Java.io.OutputStream;

    public class RingtoneResponse {
        public String inputFormat = null;
        public String outputFormat = null;
        public String inputMimeType = null;
        public String outputMimeType = null;
        private String dataString = null;
        private OutputStream dataStream = null;
        private byte[] dataBytes = null;
        private int dataType = RingtoneRequest.RINGTONE_DATA_NULL;
        public RingtoneResponse()  {
        }

        public void setData (String ringtone) {
        this.dataString = ringtone;
        this.dataType = RingtoneRequest.RINGTONE_DATA_STRING;
        }

        public void setData (OutputStream ringtone) {
        this.dataStream = ringtone;
        this.dataType = RingtoneRequest.RINGTONE_DATA_STREAM;
        }

        public void setData (byte[] ringtone) {
        this.dataBytes = ringtone;
        this.dataType = RingtoneRequest.RINGTONE_DATA_BYTES;
        }

        public String getDataAsString () {
          return this.dataString;
        }

        public OutputStream getDataAsStream() {
        return this.dataStream;
        }

        public byte[] getDataAsBytes() {
        return this.dataBytes;
        }

        /** note, no setDataType method is provided to prevent
         *  inconsistency. The dataType attribute is set when
```

```
         *  setting data.
         */
        public int getDataType() {
        return this.dataType;
        }
}
```

実装

クラス oracle.panama.multimedia.RingtoneProcessorImpl は、前述のインタフェースを実装します。この実装では、構成パラメータを参照して、着信音変換の実装のマトリックスをロードします。着信音コンバータは、フォーマットを別のフォーマットに変換し、着信音コンバータ Java API セクションに定義されているインタフェースを実装します。新しい着信音フォーマット（既存のフォーマット RTTTL に類似した RTTTL2 など）のサポートに必要なのは、RTTTL から RTTTL2（またはその逆）に変換するための Java コードを追加することのみです。新しいフォーマットを、他のすべてのサポート済フォーマットに変換する必要はありません。あるフォーマットを別のフォーマットに変換する方法が複数ある場合は、変換手順が最も少ないコンバータが選択されます。たとえば、IMelody から RTTTL2 への変換で、RTTTL を RTTTL2 に変換する新しいコンバータのみを指定した場合、MCS では、最初に IMelody を RTTTL に変換してから RTTTL2 に変換します。IMelody を RTTTL2 に変換する独自のコンバータを指定した場合は、前述のコンバータより変換手順が少ないため、このコンバータが選択されます。

構成

RingtoneProcessor インタフェースのデフォルト実装を変更する場合は、OracleAS Wireless Tools を使用して、実装クラスの名前を指定する必要があります。

1. OracleAS Wireless Tools に管理者でログインします。
2. 「システム」フォルダにある**「サイト管理」**をクリックします。
3. 「コンポーネント構成」の下にある**「マルチメディア適応サービス」**をクリックします。
4. 「着信音プロバイダ・クラス名」フィールドの値を実装クラス名に変更し、**「OK」**をクリックします。
5. 最後に、実装クラスが OracleAS Wireless のクラスパスに含まれていることを確認します。

サンプル使用方法

次のサンプル・プログラムは、RingtoneProcessor の使用方法を示します。

```
import Java.io.FileOutputStream;
import Java.io.ByteArrayOutputStream;
import oracle.panama.multimedia.RingtoneProcessor;
import oracle.panama.multimedia.RingtoneRequest;
import oracle.panama.multimedia.RingtoneResponse;
import oracle.panama.multimedia.MultimediaException;

public class RingtoneUserTest {

    public static void main(String[] args) {
        try {
            RingtoneProcessor processor =
            RingtoneProcessorProvider.getProvider();
            if (processor != null) {
               RingtoneRequest request = new RingtoneRequest();
               request.inputFormat = "IMELODY";
            request.setData("BEGIN:IMELODY¥nVERSION:1.2¥nFORMAT:CLASS1.0¥nMELODY:r1a
            1a2a3e3lmnop2a2a2a2g2a2r3a3a3e3g2a3a3a2a2g2¥nEND:IMELODY¥r¥n");
            request.outputFormat = "RTTTL";
               RingtoneResponse response = processor.process(request);
               int dType = response.getDataType();
               if (dType == RingtoneRequest.RINGTONE_DATA_STRING) {
               System.out.println(response.getDataAsString());
               } else if (dType == RingtoneRequest.RINGTONE_DATA_STREAM) {
               FileOutputStream outFile =
               new FileOutputStream("ringtone.txt");
               ((ByteArrayOutputStream)
               response.getDataAsStream()).writeTo(outFile);
               } else if (dType == RingtoneRequest.RINGTONE_DATA_BYTES) {
                 // process the byte array response
               } else {
               // process failed to set the response data
               }
        } else {
            // Processor is null!! No ringtone converter available
            System.out.println("No ringtone converter available");
        }
        } catch (Exception ex) {
            ex.printStackTrace();
        } catch (Error e) {
            e.printStackTrace();
            System.out.println("Error parsing ringtone");
        }
        }
}
```

着信音コンバータ Java API

説明

着信音変換インタフェースは、指定の着信音のフォーマットを別のフォーマットにリライトする、オーバーロードされたメソッド（convert）で構成されています。この convert メソッドは、RingtoneRequest のインスタンスを受け取り、変換された着信音を RingtoneResponse インスタンスに戻します。RingtoneProcessorImpl ではこのインタフェースを使用して、サポートされているフォーマットに応じて各種コンバータをコールします。このインタフェースを実装し、いくつかの構成情報を追加することによって、新しい着信音フォーマットのサポートが追加されます。

インタフェース oracle.panama.multimedia.RingtoneConverter

```
public RingtoneResponse convert (RingtoneRequest request);
```

実装

クラス oracle.panama.multimedia.RingtoneConverterImpl は、前述のインタフェースを実装します。この実装によって、指定されたすべての着信音変換がサポートされます。

構成

新しいフォーマットに RingtoneConverter 実装を追加することによってデフォルトの RingtoneProcessor 実装を拡張する場合は、構成ファイル config.properties を使用して RingtoneConverter 実装クラスを指定する必要があります。構成ファイルは、[ORACLE_HOME]/wireless/server/classes/oracle/panama/multimedia にあります。
新しいフォーマット名をプロパティ ringtone.formats に追加し、実装クラスをプロパティ ringtone.converters に追加する必要があります。

```
------------------
#Formats

ringtone.formats=RTTTL NOKIA IMELODY MIDI

#Converters in one string:triplets of "from format","to format","impln class"

ringtone.converters=RTTTL NOKIA oracle.panama.multimedia.RingtoneConverterImpl¥
                RTTTL IMELODY oracle.panama.multimedia.RingtoneConverterImpl ¥
                IMELODY RTTTL oracle.panama.multimedia.RingtoneConverterImpl ¥
                IMELODY MIDI oracle.panama.multimedia.RingtoneConverterImpl

------------------
```

たとえば、RTTTL2 を追加する場合のプロパティ・ファイルは次のようになります。

```
-------------------------
#Formats

ringtone.formats=RTTTL NOKIA IMELODY MIDI RTTTL2

#Converters in one string:triplets of "from format","to format","impln class"

ringtone.converters=RTTTL NOKIA oracle.panama.multimedia.RingtoneConverterImpl¥
                 RTTTL IMELODY oracle.panama.multimedia.RingtoneConverterImpl ¥
                 IMELODY RTTTL oracle.panama.multimedia.RingtoneConverterImpl ¥
                 IMELODY MIDI oracle.panama.multimedia.RingtoneConverterImpl ¥
                 RTTTL RTTTL2 my.package.RTTTLToRTTTL2Converter ¥
-----------------------
```

RTTTL2 から RTTTL へのコンバータも追加した場合は、別のトリプレット（RTTTL2 RTTTL another.package.RTTTL2Converter など）を ringtone.converters に追加する必要があります。

デバイス調整

デバイス調整は、ソース・コンテンツをターゲット・デバイスに変換するプロセスで、次のような様々な要素に対応して最適化します。

- 環境（電話会社、接続速度など）
- デバイスのフォーム要素（幅、高さ、色など）
- ユーザー作業環境

OracleAS Wireless は、XHTML/XForms、XHTML MP および OracleAS Wireless マークアップ言語で記述された入力ソース文書を各種のモバイル・デバイスに適応させます。

- HTML
- XHTML
- cHTML
- WAP/WML
- HDML
- MML
- VoiceXML
- SMS
- MMS
- Instant Messaging クライアント

OracleAS Wireless のデバイス調整によって、開発者には次のような利点があります。

- デバイス・フォーム要素が自動的に認識されます。
- すべてのデバイス間でレンダリングが最適化されます。
- フォーム要素に基づいてレンダリングが最適化されます。
- 広範なデバイス・ナレッジ・ベースを使用できます。

デバイス・リポジトリ

OracleAS Wireless のデバイス・リポジトリには、システムの中核となる豊富なデバイス情報が格納されます。管理者と開発者は、OracleAS Wireless Tools を使用して新しいデバイス情報を追加できます。

デバイス・リポジトリ内のすべての情報は、ターゲット・デバイスに対するソース・コンテンツ調整で使用されます。リポジトリ内の情報を最新の状態に維持することが重要です。

デバイス・リポジトリへのアクセス

開発者は、次の方法でデバイス・リポジトリ内の情報にアクセスできます。

- Oracle Application Server のデバイス・リポジトリ API
- W3C の CSS Media Queries 規格

Oracle Application Server のデバイス・リポジトリ API は一連の Java API で、Java と JSP の開発者がデバイス・リポジトリへのアクセスをプログラムで制御するために使用する API です。OracleAS Wireless のマークアップ作成者は、CSS Media Queries を使用して、ソース文書からデバイス・リポジトリの情報に直接アクセスできます。CSS Media Queries は W3C 規格（勧告候補ステータス）です。CSS Media Queries の詳細は、付録 D「OracleAS Wireless による CSS のサポート」を参照してください。

デバイスの検出

OracleAS Wireless は、サービス・リクエストを発行したデバイスのタイプを自動的に検出します。デバイス検出コンポーネントは、UserAgent（使用可能な場合）を使用して、デバイス・リポジトリ内のデバイスの中からサービス・リクエストに関連付けるデバイスを判別します。

デバイスの検出ルールは、次のとおりです。

1. UserAgent が HTTP ヘッダーで使用不可の場合は、ステップ 4 に進みます。
2. ユーザー・エージェント一致文字列が HTTP ヘッダーの UserAgent と一致するデバイスをデバイス・リポジトリから選択します。
3. 複数のデバイスが戻された場合は、ユーザー・エージェント一致文字列が最も長いデバイスを選択します。この結果、完全に一致するデバイスが 1 つの場合は、そのデバイスが戻されます。これで、デバイスの検出が完了します。
4. Accept HTTP ヘッダーを使用し、IETF RFC-2616 仕様に従って優先コンテンツ・タイプを判別します。
5. 優先 MIME タイプと一致する最初のデバイスが戻されます。
6. リクエストに x-up-devcap-screenpixels および x-up-devcap-screenchars HTTP ヘッダーが含まれている場合は、デバイスの ScreenWidth、ScreenHeight、ScreenRows、ScreenColumns 属性を使用して最も近似のロジカル・デバイスが検出されます。

7. デバイスが検出されない場合は、ログ・ファイルにエラーが記録されます。

OracleAS Wireless でのデバイス検出は、次のインタフェースを実装するフック・クラスを指定することで、カスタマイズできます。

```
oracle.panama.rt.hook.DeviceIdentificationHook
```

フックのデフォルト実装は、次のとおりです。

```
oracle.panma.rt.hook.DeviceIdentificationPolicy class
```

動的な HTTP ヘッダー作成と UAProf

デバイス・リポジトリ API は、HTTP リクエストにフォーム要素情報がある場合、HTTP ヘッダーの動的なフォーム要素作成を実行します。動的な HTTP ヘッダー作成は、次の手順で実行されます。

1. リポジトリからデバイス情報を取り出し、`oracle.panama.model.DeviceV2 object` のインスタンスを作成します。
2. HTTP ヘッダー内で既知のデバイス・フォーム要素情報を検索し、`oracle.panama.model.DeviceV2` の適切な属性を更新します。

デバイス・トランスフォーマ

Oracle Application Server のデバイス調整の最終フェーズでは、デバイス・トランスフォーマを選択して起動し、サポートされているソース入力文書からターゲット・デバイスへのレンダリングに適したマークアップ言語を生成します。デバイス・リポジトリ内のすべてのデバイスには、そのデバイスに適したトランスフォーマのリストがあります。

デバイス・トランスフォーマは、入力ソース文書として受け取る OracleAS Wireless のマークアップ言語ごとに、次に示すように、グループ化されます。

表 9-1 デバイス・トランスフォーマの入力マークアップ

トランスフォーマの接頭辞	OracleAS Wireless のマークアップ	OracleAS Wireless のマークアップの MIME タイプ
mxml-	mobile-xml	text/vnd.oracle.mobilexml
xforms-	xhtml+xforms	application/vnd.oracle.xhtml+xforms
xhtml-	xhtml+mp	application/vnd.wap.xhtml+xml

次の表に、OracleAS Wireless のマークアップ言語で受け取られるトランスフォーマ、およびトランスフォーマで生成されるマークアップを示します。

表 9-2 mobile-xml 用のデバイス・トランスフォーマ

トランスフォーマ	ターゲット・マークアップ	説明
mxml-ASYNC_Java	text/plain	SMS デバイス、テキスト
mxml-Adomo	text/vxml	Adomo 音声ゲートウェイ
mxml-Verascape	text/vxml	Verascape 音声ゲートウェイ
mxml-VoiceGenie	text/vxml	VoiceGenie 音声ゲートウェイ
mxml-avantgo	text/html	AvantGo ブラウザ
mxml-blazer	text/html	Handspring Blazer ブラウザ
mxml-chtml	text/html	cHTML ブラウザ
mxml-ciscoip	text/xml	Cisco IP 電話
mxml-goweb	text/html	GoWeb ブラウザ
mxml-hdml	text/x-hdml	HDML ブラウザ
mxml-hdml-kddi	text/x-hdml	EZweb HDML ブラウザ
mxml-html32	text/html	W3C HTML 3.2 準拠のブラウザ
mxml-html40	text/html	W3C HTML 4.0 準拠のブラウザ
mxml-mml	text/html	J-PHONE Type C3 以上
mxml-mml-t04	text/html	J-PHONE Type C2
mxml-palm-family	text/html	Palm ブラウザ
mxml-pocketpc	text/html	PocketPC PDA ブラウザ
mxml-smil	application/smil	MMS SMIL
mxml-wml11	text/vnd.wap.wml	WML11 準拠のブラウザ
mxml-wml11-ericsson	text/vnd.wap.wml	Ericsson WML11 ブラウザ
mxml-wml11-openwave	text/vnd.wap.wml	Openwave WML11 ブラウザ
mxml-wml11-wig	text/vnd.wap.wml	WIG ブラウザ
mxml-xmp	text/html	XHTML MP ブラウザ

次の表に、XHTML+XForms 用のデバイス・トランスフォーマを示します。

表 9-3 XHTML+XForms 用のデバイス・トランスフォーマ

トランスフォーマ	ターゲット・マークアップ	説明
xforms-Verascape	text/vxml	Verascape 音声ゲートウェイ
xforms-VoiceGenie	text/vxml	VoiceGenie 音声ゲートウェイ
xforms-async_xhtml	text/plain	SMS、テキスト
xforms-chtml	text/html	cHTML ブラウザ
xforms-hdml	text/x-hdml	HDML ブラウザ
xforms-html32	text/html	W3C HTML 3.2 準拠のブラウザ
xforms-html32-handheld	text/html	HTML 3.2 HandHeld Friendly ブラウザ
xforms-html40	text/html	W3C HTML 4.0 準拠のブラウザ
xforms-mml	text/html	J-PHONE Type C3 以上
xforms-mms-smil	application/smil	MMS SMIL
xforms-palm-family	text/html	Palm ブラウザ
xforms-wml11	text/vnd.wap.wml	WML11 準拠のブラウザ
xforms-wml11-ericsson	text/vnd.wap.wml	Ericsson WML11 ブラウザ
xforms-wml11-openwave	text/vnd.wap.wml	Openwave WML11 ブラウザ
xforms-xmp	text/html	XHTML MP ブラウザ

次の表に、XHTML + MP 用のデバイス・トランスフォーマを示します。

表 9-4 XHTML+MP 用のデバイス・トランスフォーマ

トランスフォーマ	ターゲット・マークアップ	説明
xhtml-chtml	text/html	cHTML ブラウザ
xhtml-hdml	text/x-hdml	HDML ブラウザ
xhtml-html32	text/html	W3C HTML 3.2 準拠のブラウザ
xhtml-html32-handheld	text/html	HTML 3.2 HandHeld Friendly ブラウザ
xhtml-html40	text/html	W3C HTML 4.0 準拠のブラウザ
xhtml-mml	text/html	J-PHONE Type 3
xhtml-mms-smil	application/smil	MMS SMIL
xhtml-palm-family	text/html	Palm ブラウザ
xhtml-wml11	text/vnd.wap.wml	WML11 準拠のブラウザ
xhtml-wml11-ericsson	text/vnd.wap.wml	Ericsson WML11 ブラウザ
xhtml-wml11-openwave	text/vnd.wap.wml	Openwave WML11 ブラウザ
xhtml-xmp	text/html	XHTML MP ブラウザ

デバイス・リポジトリ API

oracle.panama.model.Device API は使用されなくなりました。このリリースでは、新しい API の oracle.panama.model.DeviceV2 を使用します。Java アプリケーションと JSP アプリケーションからデバイス・リポジトリにアクセスするには、DeviceV2 インタフェースを使用する必要があります。

DeviceV2 インタフェースには、次に示すように、古い Device API からもアクセスできます。

```
Device device = RequestFactory.lookupRequest();
DeviceV2 devicev2 = device.getDeviceV2();
```

このコード部分は、現行の HTTP リクエストのコンテキストに応じて、デバイス・リポジトリからターゲット・デバイスを取り出します。つまり、devicev2 は実際のデバイス情報へのハンドルです。

DeviceV2 インタフェースへのハンドルが取得されると、デバイス属性や機能の取出しは直接行われます。機能の値を取り出すには、次の例に示すように、Java の boolean、String または int の 3 つのメソッドがあります。

```
boolean bool = devicev2.getDeliveryContextAttributeBoolean(DeviceAttr.COLORCAPABLE);
String model = devicev2.getDeliveryContextAttributeString(DeviceAttr.MODEL);
int width = devicev2.getDeliveryContextAttributeString(DeviceAttr.DEVICEWIDTH);
```

デバイスの属性または機能は、すべて oracle.panama.model.DeviceAttr インタフェースにリストされています。次の表に、すべてのデバイスの属性または機能を示します。

表 9-5 一般的なデバイス機能

デバイス属性	説明
VENDOR	デバイス・メーカー。
MODEL	モデル番号。
DEVICECLASS	デバイス・クラス（使用中止）。
MEDIA	CSS Media Queries で使用する CSS メディア・タイプ。
DEVICETAG	リポジトリ内の関連するデバイスを識別してグループ化するタグ。
DEVICEWIDTH	表示領域の幅（ピクセル単位）。
DEVICEHEIGHT	表示領域の高さ（ピクセル単位）。
PIXELPITCH	ピクセルのサイズ（mm 単位）（ビットマップ・デバイスのみ）。
DEFAULTFONTSIZE	デバイスで使用するデフォルトのフォント・サイズ。
GRID	グリッド・デバイス（ビットマップ・デバイス以外）。
COLORCAPABLE	true の場合、デバイスでは色をレンダリングできます。

表 9-5 一般的なデバイス機能（続き）

デバイス属性	説明
PAGEDMEDIA	true の場合、これは WAP/WML などページ化されたデバイスです。
BITSPERPIXEL	モノクローム・デバイス用のピクセル当たりのビット数、またはカラー・デバイス用のカラー・コンポーネント当たりのビット数。
MAXDOCSIZE	最大ドキュメント・サイズ。
TEXTINPUTCAPABLE	true の場合、デバイスではテキスト入力をサポートします。
NUMBEROFSOFTKEYS	デバイスでサポートされるソフトキーの数。
KEYBOARD	キーボード・タイプ。qwerty、phone keypad、disambiguating のいずれかです。

表 9-6 ブラウザ機能

ブラウザ機能	説明
MARKUP LANGUAGE	デバイスでサポートするマークアップのリスト。
PROLOG	文書の XML プロローグ。
SUPPORTSAMPERSANDENTITY	true の場合、デバイスでは XML アンパサンド・エンティティをサポートします （使用中止）。
SUPPORTSRELATIVEURL	true の場合、関連する URL がサポートされます。
SUPPORTSCOOKIE	true の場合は、Cookie がサポートされます。
MESSAGINGBASED	true の場合は、非同期メッセージがサポートされます。
TABLESCAPABLE	true の場合は、表がサポートされます。
AUDIOCONTENT	サポートされるオーディオ MIME タイプのリスト。
EMAILCAPABLE	true の場合は、電子メールの送受信が可能になります。
TEXTTOSPEECH	true の場合は、TTS エンジンがサポートされます。
SPEECHGRAMMAR	true の場合は、構文がサポートされます。
RECORDSPEECH	true の場合は、音声を録音できます。
VOICECALLCAPABLE	true の場合は、音声でコールできます。
CALLCONTROLCAPABLE	true の場合は、コールを制御できます。
DEFAULTMARKUPLANGUAGE	デバイスに送信するデフォルトの MIME タイプ。
ACCEPT	受け入れた MIME タイプのリスト。

表 9-6 ブラウザ機能（続き）

ブラウザ機能	説明
ACCEPT_CHARSET	受け入れた文字コードのリスト。
IMAGECAPABLE	true の場合は、イメージを表示できます（使用中止）。
IMAGECONTENTTYPES	サポートされるイメージ MIME タイプのリスト。
VICEOCAPABLE	true の場合は、ビデオがサポートされます（使用中止）。
VIDEOCONTENTTYPES	サポートされるビデオ MIME タイプのリスト（使用中止）。
VIDEOMODE	サポートされるビデオ・モード（使用中止）。
AUDIOCONTENTTYPES	サポートされるオーディオ MIME タイプのリスト。
REVERSEENTITYMAP	XML エンティティの変換ルールのリスト。

表 9-7 メッセージ機能

メッセージ機能	説明
DELIVERYTYPES	サポートされるメッセージ配信タイプまたはチャネルのリスト。
BANDWIDTH	ネットワーク速度。
URLCAPABLE	URL のハイパーリンクに直接従うことができます。
NOKIASMARTMESSAGING CAPABLE	true の場合は、Nokia スマート・メッセージがサポートされます。
RINGTONECAPABLE	true の場合、ダウンロード可能な着信音がサポートされます。
OPERATORLOGOCAPABLE	true の場合は、オペレータ・ロゴをダウンロードできます。
VCARDCAPABLE	true の場合は、VCARD をサポートできます。
VCALENDARCAPABLE	true の場合は、VCALENDAR がサポートされます。
SYNCMLCAPABLE	true の場合は、SYNCML がサポートされます。
MESSAGESIZELIMIT	1 メッセージ当たりの最大文字数。
MULTIMEDIAAUDIOFORMATS	マルチメディア・オーディオ・タイプ。
MULTIMEDIAIMAGEFORMATS	マルチメディア・イメージ・タイプ。
MULTIMEDIAVIDEOFORMATS	マルチメディア・ビデオ・フォーマット。
SMILLAYOUTCAPABLE	true の場合、MMS ブラウザでは SMIL レイアウト・タグをサポートします。

表 9-8 VoiceXML ゲートウェイ機能

VoiceXML ゲートウェイ機能	説明
GRAMMARCONTENTTYPES	サポートされる音声構文の MIME タイプのリスト。
MIMETYPE_TEXTTOVOICEGRAMMAR	テキストから生成された音声構文のゲートウェイ固有の MIME タイプ。
MIMETYPE_OGSTOVOICEGRAMMAR	OGS 構文から生成された音声構文のゲートウェイ固有の MIME タイプ。
MIMETYPE_TEXTTODTMFGRAMMAR	テキストから生成された DTMF 構文のゲートウェイ固有の MIME タイプ。
MIMETYPE_OGSTODTMFGRAMMAR	OGS 構文から生成された DTMF 構文のゲートウェイ固有の MIME タイプ。
MIMEMAP_APPLICATION_SRGS_XML	application/srgs+xml に対するゲートウェイ固有の MIME タイプを入力します。
MIMEMAP_APPLICATION_X_ABNF	application/x-abnf に対するゲートウェイ固有の MIME タイプを入力します。
MIMEMAP_APPLICATION_X_GSL	application/x-gsl に対するゲートウェイ固有の MIME タイプを入力します。
MIMEMAP_APPLICATION_X_JSGF	application/x-jsgf に対するゲートウェイ固有の MIME タイプを入力します。
MIMEMAP_APPLICATION_X_DTMF	application/x-dtmf に対するゲートウェイ固有の MIME タイプを入力します。
MIMEMAP_APPLICATION_X_WATSON	application/x-watson に対するゲートウェイ固有の MIME タイプを入力します。
MIMEMAP_APPLICATION_X_SWI	application/x-swi に対するゲートウェイ固有の MIME タイプを入力します。

表 9-9 Java（J2ME）機能

Java（J2ME）機能	説明
JavaCAPABLE	true の場合は、J2ME がサポートされます。
JavaPLATFORM	CDC などのデバイスにインストールされた Java 構成。
JVMVERSION	JavaVM バージョン。
JavaPROFILE	MIDP などのデバイスにインストールされた Java プロファイル。
JavaPROVISIONPROTOCOL	SUN-OTA などのプロビジョニング・プロトコル。
JavaMAXDOWNLOADSIZE	インストールされた JVM でダウンロードできる最大サイズ。
JVMHEAPSIZE	JVM ヒープ・サイズ。

デバイスの情報と分類

OracleAS Wireless サーバーは、HTTP ヘッダーを使用して、ユーザー・デバイスに関する情報をバックエンド・アプリケーションに送信します。アプリケーションでは、この情報を使用して、生成するコンテンツを最適化できます。次の表に、アプリケーションが受け取るヘッダーを示します。

表 9-10 デバイスの情報と分類

HTTP ヘッダー名	説明
X-Oracle-Device.Class	デバイスのチャネル・モードとフォーム要素を示します。Device.class のそれぞれの値は、一意の通信チャネル・モードと一意のフォーム要素を示します（可能な値については、後述する説明と代表的なデバイスを参照）。
X-Oracle-Device.Orientation	デバイスのオリエンテーションは、フォーム要素とともに、アプリケーションで特定デバイスのレンダリング・スタイルを変更するのに役立ちます。可能な値は、landscape と portrait で、デフォルト値は portrait です（システムによる指定がない場合、または幅と高さが同じ場合）。
X-Oracle-Device.MaxDocSize	現行リクエストを発行したデバイスで処理される XML 文書（サービス・レスポンス）の最大サイズ（バイト単位）。文書のバイト・サイズとターゲット・デバイスのダイジェスト・バイト・サイズはマップできないため、この値は近似値になります。オーディオやイメージなどの埋込みコンテンツでは、このサイズを考慮する必要があります。サービスが MaxDocSize より大きい XML 文書を戻す場合、そのリクエストに対するレスポンスは未定義になります。
X-Oracle.Device.Secure	可能な値は true または false です。OracleAS Wireless サーバーとデバイス間の接続が、現行のリソース・リクエストの発行時に保護されていたかどうかを示します。
X-Oracle-Orig-User-Agent	OracleAS Wireless サーバーへのリクエストが HTTP プロトコルを介して発行され、デバイスが User-Agent HTTP ヘッダーを送信した場合は、そのヘッダー名を使用してヘッダーがアプリケーションに再送信されます。
X-Oracle-Orig-Accept	OracleAS Wireless サーバーへのリクエストが HTTP プロトコルを介して発行され、デバイスが Accept HTTP ヘッダーを送信した場合は、そのヘッダー名を使用してヘッダーがアプリケーションに再送信されます。

マルチチャネル・サーバー Runtime の変更

OracleAS Wireless マルチチャネル・サーバー（MCS）Runtime は、OC4J サーブレット、非同期サーバー、音声サーバーからは直接起動され、OC4J Servlet Filter からは間接的に起動されます。MCS Runtime は、様々な通信チャネル（音声、Hypertext Transaction Protocol（HTTP）、Instance Messaging、SMS、電子メール、双方向ポケットベルなど）を使用するデバイス、ユーザー・エージェントおよび自律型モバイル・エージェントからのリクエストを処理します。MCS Runtime は、これらのチャネルからのサービス・リクエストを調整し、デバイス固有の機能を利用するためにサービス・レスポンスにトランスコードします。これによって、各デバイスが持つ様々な特異性から開発者を解放します。リクエストは、OracleAS Wireless MCS によって、異なる通信チャネルから標準 J2EE サーブレット 2.3 サービス・リクエストに調整されるため、開発者は、業界標準のサーブレット API、JSP、XHTML、XForms、CSS、さらには Oracle 独自の OracleAS Wireless XML を使用して、汎用的なモバイル・アプリケーションを開発できます。MCS では、中央管理されたデバイス情報リポジトリ内にあるデバイス・モデルの拡張可能なレパートリを効率的に利用できるため、デバイス固有の機能を活用できます。

この項では、MCS Runtime の機能について説明します。ここでは、MCS Runtime セッション管理、セッションの永続性、Runtime API と拡張性、コンテンツ調整および URL リライト・メカニズムについて説明します。また、MCS Runtime は、自動セッション・トラッキングを実施し、タイムアウト時間を経過してセッションが期限切れになった場合、またはデバイスが切断された場合に、セッションを終了します。

MCS Runtime セッション管理

OracleAS Wireless MCS Runtime では、MCS セッション ID を指定するパラメータ *PAsid* を追加して各 URL をリライトすることで、OC4J サーブレット・セッションから独立して Runtime セッションが追跡されます。セッション・トラッキングによって、一連のリクエストが同じユーザーから発行されていることが識別されます。MCS Runtime セッションには、デバイス資格証明、ユーザー作業環境、Runtime コンテキスト、Cookie、URL キャッシュおよび状況依存サービスに必要な他の状態が含まれています。さらに、これらの MCS セッションの状態は永続的な場合もあります。MCS セッション ID がリクエストで参照されると、その MCS セッション ID を使用して、MCS セッションの永続的な状態がリストアされます。たとえば、*PAsid* パラメータが URL に格納されます。MCS Runtime では Runtime セッションをメンテナンスします。これによって、代替チャネルの一時セッションを介して接続しているユーザーは、長期間有効な MCS Runtime セッションを共有できます。永続的な MCS セッションによって、セッションの継続期間が長くなり、複数モーダルの相互作用がさらに継続されます。

MCS Runtime セッションは、OC4J サーブレット・セッションにバインドできます。WAP HTTP State Management Specification（http://www.wapforum.org/）を実装している WAP 2.0 デバイスでは、セッション管理のために Cookie をサポートできます。市販の WAP ゲートウェイのほとんどでは、WAP デバイスにかわって Cookie が管理されます。デバイスまたはゲートウェイで Cookie がサポートされない場合、OC4J サーブレット・コンテナは URL をリライトしてセッションを追跡します。MCS Runtime でもセッションが追跡されるため、同じ OC4J サーブレット・セッションに複数の MCS Runtime セッションをバインド

できます。たとえば、Cookie リポジトリは共有のため、複数のブラウザが同じサーブレット・セッションを共有する場合がありますが、同じデバイス上の 2 つのブラウザ・ウィンドウで 2 つの独立した MCS Runtime セッションをオープンできます。

MCS Runtime セッションの状態は同じアイランドにある他の OC4J インスタンスにレプリケートできるため（アイランドは、複数の OC4J インスタンス間でセッションの状態をレプリケートします)、最初のインスタンスが失敗した場合は、デバイス・リクエストを同じアイランドにある他の OC4J インスタンスにリダイレクトできます。デフォルトでは、OC4J サーブレット・セッションへのバインドは使用可能になっており、OC4J セッションのレプリケーションおよびフェイルオーバーを構成するにはこのバインドが必要です。MCS Runtime セッションが他のチャネルのアクティブなセッション（音声、Instance Messaging、SMS など）にバインドされていない場合は、サーブレット・セッションが失効すると、そのサーブレット・セッションにバインドされている MCS Runtime セッションは失効します。MCS Runtime セッションの失効によって解放されるのはメモリー内のリソースのみで、永続的な MCS セッションの状態は維持されます。このセッションの状態は、Runtime セッションが再度アクティブ化されるとリストアできます。

MCS Runtime セッションは、アイドル時間がサイト全体の構成パラメータ「ランタイム・セッション・ライフ時間（秒)」で指定した値を超えると失効します。このパラメータは、「システム」→「`Wireless` サーバー : 管理」→「ランタイム構成」コンソールにあります。デフォルトのセッション・ライフ時間は 10 分です。MCS Runtime セッションが OC4J サーブレット・セッションにバインドされている場合、このパラメータは、OC4J サーブレット・セッションの有効期限（デフォルトは 30 分）でオーバーライドされます。Runtime セッションからサーブレット・セッションにバインドしているセッションは、`System.properties` ファイル内のパラメータ設定 `enable.http.session.binding=false` で使用禁止にできます。MCS セッションの永続性を有効にするには、「システム」→「`Wireless` サーバー : 管理」→「ランタイム構成」コンソールにある「ランタイム・セッションの永続性の有効化」オプションを使用します。デフォルトでは、Runtime セッションの永続性は無効です。永続的な MCS セッションは、アイドル時間が、同じコンソールにある「永続的なセッション・ライフ時間（日)」で指定された日数を超えると、永続記憶域から消去されます。MCS の永続的なセッション・ライフ時間には、OC4J サーブレット・セッション・ライフ時間や MCS Runtime セッション・ライフ時間より長い日数を指定できます。デフォルト設定は 2 日です。

MCS Runtime API

OracleAS Wireless MCS Runtime API は、Java インタフェースを提供してランタイム実行ステータスを検証し、ランタイム実行フローを追跡し、デフォルトの実行セマンティクスを補強します。Runtime API は、次の Java パッケージで構成されています。

- `oracle.panama.rt` は、ステータスの管理に必要な基本 Runtime オブジェクトへのインタフェースを提供します。
- `oracle.panama.rt.event` は、Java イベント・モデルに基づいてランタイム実行順序を監視するためのインタフェースを提供します。
- `oracle.panama.rt.hook` は、Runtime のカスタマイズ可能な基本コンポーネントへのインタフェースとこれらのインタフェースに対するデフォルトの実装ポリシーを提供します。

これらのパッケージは、`wireless.jar` ファイルに含まれています。Java アプリケーションまたは MCS Runtime API に依存するプラグイン・コンポーネントをコンパイルする場合は、Java クラスパスに `wireless.jar` を挿入したことを確認してください。

Runtime オブジェクト

`oracle.panama.rt` パッケージは、Runtime API のコアを定義します。イベント・リスナーなど、Runtime のカスタム・プラグイン・コンポーネントは、`oracle.panama.rt` パッケージの Request、Response および Session の各インタフェースを使用できます。

次の項では、このパッケージのインタフェースについて説明します。次のインタフェースがあります。

- Request
- Response
- Session

Request リクエスト・オブジェクトは、サービス・リクエストの URL パラメータおよび HTTP ヘッダー属性を定義するために使用されます。また、ユーザー・エージェント・タイプ、デバイス・モデルおよびその他の Runtime コンテキストも定義します。リスナーは、リクエストからイベントをサブスクライブできます。

Request インタフェースの次のメソッドによって、リクエスト・オブジェクトに関連付けられているパラメータをアクセス、置換、追加または削除できます。

- `Object getAttribute(AttributeCategory category, String name)`
- `Object setAttribute(AttributeCategory category, String name, Object attribute)`

このメソッドを使用して、属性の名前と値にアクセスします。属性には、ユーザー・パラメータ、システム・パラメータ、または、アプリケーション・コンテキストがあります。属性のカテゴリは 3 種類あります。

- PARAMETERS
- RUNTIME
- CONTEXTS

Request に関して最も重要な属性カテゴリは、ユーザーが発行した問合せパラメータとフォーム・パラメータを含む PARAMETERS です。MCS Runtime は、パラメータを取り出すために URL 問合せ文字列を解析します。Runtime のその他のコンポーネントは、これらのパラメータをプログラム上で設定できます。MCS Runtime は URL をリライトおよびキャッシュするため、一部のパラメータは Runtime セッション内で URL キャッシュから取り出されます。MCS Runtime では、セッション内で HTTP リクエストからの問合せ文字列と URL キャッシュの両方を解析して、問合せパラメータの詳細リストを作成する場合があります。

Response Response インタフェースは、MCS Runtime のレスポンス・オブジェクトを表します。リスナーは、Response からイベントをサブスクライブできます。Response オブジェクトは、Request オブジェクトの実行結果です。

Session Session インタフェースは、MCS Runtime のセッション・オブジェクトを表します。リクエストは、有効なセッション・コンテキストでのみ実行できます。セッションは、最大非活動間隔を超過した場合に期限切れになります。開発者は、セッション期間中の情報を対応するセッション・オブジェクトに格納できます。リスナーは、セッションからイベントをサブスクライブできます。MCS Runtime セッションの確立方法については、9-31 ページの「MCS Runtime セッション管理」を参照してください。

セッションは、他の Runtime コンテキスト間で、デバイス資格証明、パーソナライズ作業環境、圧縮していない URL、Cookie および XForms 文書をキャッシュします。SMS および Instant Messaging サーバーは、Runtime セッションを広範囲にわたって使用して、動的に生成された短縮名（URL を識別する 1 桁のメニュー番号など）を管理します。セッションの状態には、コンテンツ・プロバイダのアプリケーションに対する認証の様々な状態を表す Cookie が含まれます。MCS Runtime は、Runtime セッションで XForms 文書をキャッシュします。セッション所有者は、インスタンス・データをアプリケーションに送信する前に、リクエスト / レスポンス・メッセージをいくつか使用して XForms 文書と相互作用できます。

Session インタフェースの次のメソッドによって、セッション・オブジェクトに関連付けられているパラメータをアクセス、置換、追加または削除できます。

- `Object getAttribute(AttributeCategory category, String name)`
- `Object setAttribute(AttributeCategory category, String name, Object attribute)`

このメソッドを使用して、属性の名前と値にアクセスします。属性のカテゴリは 3 種類あります。

- PARAMETERS
- RUNTIME
- CONTEXTS

イベント・リスナー

MCS Runtime は、OC4J サーブレット、Servlet Filter、音声サーバーまたは非同期サーバーから起動されます。サーブレット 3.2 フィルタと同様に、MCS Runtime controller は、MCS Runtime の交換可能な Request、Response および Session の各リスナーを介して拡張できます。

これらのオブジェクトに関係するリスナーが登録されている場合、MCS セッションの確立、MCS セッションの失効またはリクエストとレスポンスの処理中に、MCS Runtime は実行の進捗状況を通知するイベントの順序を生成します。リスナーは、MCS Runtime controller の実行フローを変更せず、その進捗を監視し、リクエストおよびレスポンスのデータを変更できます。イベント・リスナーに対して有効なアプリケーションには、データ・ロギング、パフォーマンス監視、およびコンテキストを認識した高度なカスタマイズが含まれます。`oracle.panama.rt.event` パッケージは、JDK イベント・モデルに基づいて API を定義します。

リスナーとイベントは、リスナーがオブザーバを表す重要なオブザーバ設計パターンを形成します。3 つのタイプのリスナーが定義されています。

- RequestListener インタフェース
- ResponseListener インタフェース
- SessionListener インタフェース

ListenerRegistrationHook は、Request、Response、Session などの対象からイベントを受信するためにリスナーをサブスクライブします。

RequestListener インタフェース `oracle.panama.rt.event.RequestListener` の実装側は、次のイベントを受信できます。

- before request
- request begin
- service begin
- service end
- transform begin
- transform end
- request end

- after request
- request error

どのリクエスト関連イベントが生成されるかは、OracleAS Wireless Tools の「システム・マネージャ」→「サイト」→「Wireless Web サーバー」→「イベントおよびリスナー」でイベント・マスクを指定して制御できます。たとえば、RequestListener に request begin イベントを受信させる場合は、OracleAS Wireless Tools の「システム・マネージャ」→「サイト」→「Wireless Web サーバー」→「イベントおよびリスナー」コントロール・パネルで「'request begin' イベントを使用可能にする」オプションを true に設定する必要があります。 サイト構成プロパティ名を次に示します。

- wireless.http.event.beforeRequest
- wireless.http.event.requestBegin
- wireless.http.event.requestEnd
- wireless.http.event.serviceBegin
- wireless.http.event.serviceEnd
- wireless.http.event.transformBegin
- wireless.http.event.transformEnd
- wireless.http.event.requestError
- wireless.http.event.afterRequest

RequestListener は、requestBegin(RequestEvent) の処理中に入力パラメータを傍受し、サービスの起動前に、リクエスト・パラメータに追加ビジネス・ルールを適用できます。

ResponseListener インタフェース oracle.panama.rt.event.ResponseListener の実装側は、レスポンス関連イベントを受信できます。有効なレスポンス関連イベントは、response error のみです。MCS Runtime で ResponseListener に response error イベントを受信させる場合は、OracleAS Wireless Tools の「システム・マネージャ」→「サイト」→「Wireless Web サーバー」→「イベントおよびリスナー」コントロール・パネルで「'response error' イベントを使用可能にする」オプションを true に設定する必要があります。サイト構成プロパティ名を次に示します。wireless.http.event.responseError

SessionListener インタフェース oracle.panama.rt.event.SessionListener の実装側は、セッション・ライフ・サイクル・イベントを受信できます。有効なセッション・イベントは、次のとおりです。

- before session
- session begin
- session authenticated
- session end
- after session

どのセッション・イベントが生成されるかは、OracleAS Wireless Tools の「システム・マネージャ」→「サイト」→「Wireless Web サーバー」→「イベントおよびリスナー」コントロール・パネルでイベント・マスクを指定して制御できます。たとえば、SessionListener に session begin イベントを受信させる場合は、OracleAS Wireless Tools の「システム・マネージャ」→「サイト」→「Wireless Web サーバー」→「イベントおよびリスナー」コントロール・パネルで「'session begin' イベントを使用可能にする」オプションを true に設定します。サイト構成プロパティ名を次に示します。

- wireless.http.event.beforeSession
- wireless.http.event.sessionBegin
- wireless.http.event.sessionAuthenticated
- wireless.http.event.sessionEnd
- wireless.http.event.afterSession

MCS Runtime リスナー実装のガイドライン

次に、カスタマイズしたイベント・リスナーを設定する手順を説明します。

注意： クラスパスを正しく設定していることを確認し、関連するすべてのファイルを追加する必要があります。追加する必要があるファイルは、log.xml ファイルで確認してください。

---

1. RequestListener、ResponseListener または SessionListener インタフェースを実装します。

   注意： 各リスナーで次のメソッドを提供する必要があります。

   ```
   public static <ClassName> getInstance();
   ```

   ---

2. クラスパスの wireless.jar ファイルを使用して、手順 1 からの新規 Java ソース・ファイルをコンパイルします。
3. 「システム・マネージャ」→「サイト」→「Wireless Web サーバー」→「イベントおよびリスナー」コントロール・パネルでイベント・マスク・エントリを変更して、特定のイベントの生成を使用可能にします。
4. OracleAS Wireless Tools の「システム・マネージャ」→「サイト」→「Wireless Web サーバー」→「イベントおよびリスナー」コントロール・パネルで、RequestListener、ResponseListener および SessionListener のクラス名を指定します。サイト構成プロパティ名を次に示します。
   - wireless.http.locator.combined.listener.classes

- wireless.http.locator.session.listener.classes
- wireless.http.locator.response.listener.classes
- wireless.http.locator.request.listener.classes

5. OracleAS Wireless インスタンスを再起動します。

いずれかのイベント・リスナーで AbortServiceException がスローされ、リクエストの拒否を MCS Runtime controller に通知する場合がありますが、この拒否通知は、サービスが起動されていない時点で、次のいずれかのイベントが実行されている間に発生した場合にのみ有効です。

- beforeRequest(RequestEvent)
- beforeSession(SessionEvent)
- sessionAuthenticated(SessionEvent)
- requestBegin(RequestEvent)
- sessionBegin(SessionEvent)
- serviceBegin(RequestEvent)

serviceEnd()、transformBegin() および transformEnd() イベントの実行時に、複数のリスナーで AbortServiceException がスローされ、ユーザーに対するサービスのコンテンツが拒否される場合がありますが、サービス起動の継続的な効果はロールバックできません。sessionEnd()、afterSession()、requestEnd() および afterRequest() メソッドでは、AbortServiceException はスローされません。

次のサンプル・コードで、Request、Response および Session のリスナー・インタフェースを実装するリスナーについて説明します。このサンプルのリスナーでは、Request、Response および Session のすべてのイベントをリスニングしています。また、このサンプルのリスナーでは、リクエストの応答時間を記録します。

イベント・リスナーの実装

```
package oracle.panama.rt.event;

import oracle.panama.rt.Request;
import oracle.panama.rt.Response;
import oracle.panama.rt.Session;
import oracle.panama.rt.AttributeCategory;
import oracle.panama.rt.event.RequestEvent;
import oracle.panama.rt.event.ResponseEvent;
import oracle.panama.rt.event.SessionEvent;
import oracle.panama.rt.event.RequestListener;
import oracle.panama.rt.event.ResponseListener;
import oracle.panama.rt.event.SessionListener;
import oracle.panama.rt.event.AbortServiceException;
```

```
public class Listener implements RequestListener, ResponseListener, SessionListener {

      private final static String BEFORE_REQUEST    = "L__L1";
      private final static String REQUEST_BEGIN     = "L__L2";
      private final static String SERVICE_BEGIN     = "L__L3";
      private final static String SERVICE_END       = "L__L4";
      private final static String TRANSFORM_BEGIN   = "L__L5";
      private final static String TRANSFORM_END     = "L__L6";
      private final static String REQUEST_END       = "L__L7";
      private final static String AFTER_REQUEST     = "L__L8";
      private final static String BEFORE_SESSION    = "L__L9";
      private final static String SESSION_BEGIN     = "L__LA";
      private final static String SESSION_END       = "L__LB";
      private final static String AFTER_SESSION     = "L__LC";

      public void beforeSession(SessionEvent event) throws AbortServiceException {
            System.out.println(event.toString());
      }

      public void sessionBegin(SessionEvent event) throws AbortServiceException {       //[29]
            StringBuffer buf = new StringBuffer(1000000);
            event.getSession().setAttribute(AttributeCategory.RUNTIME, this.SESSION_BEGIN, buf);
      }

      public void beforeRequest(RequestEvent event) throws AbortServiceException {
            event.put(BEFORE_REQUEST, new Long(event.getTimeStamp()));
      }

      public void requestBegin(RequestEvent event) throws AbortServiceException {
            event.put(REQUEST_BEGIN, new Long(event.getTimeStamp()));
      }

      public void serviceBegin(RequestEvent event) throws AbortServiceException {
            event.put(SERVICE_BEGIN, new Long(event.getTimeStamp()));
      }

      public void serviceEnd(RequestEvent event) throws AbortServiceException {
            event.put(SERVICE_END, new Long(event.getTimeStamp()));
      }

      public void transformBegin(RequestEvent event) throws AbortServiceException {
            event.put(TRANSFORM_BEGIN, new Long(event.getTimeStamp()));
      }

      public void transformEnd(RequestEvent event) throws AbortServiceException {
            event.put(TRANSFORM_END, new Long(event.getTimeStamp()));
      }

      public void requestEnd(RequestEvent event) throws AbortServiceException {
```

```
            event.put(REQUEST_END, new Long(event.getTimeStamp()));
      }

       public void afterRequest(RequestEvent event) throws AbortServiceException {                    //[54]
             Long val;
             long t1, t2, t3, t4, t5, t6;

              StringBuffer buf = (StringBuffer) event.getRequest().getSession().getAttribute(
AttributeCategory.RUNTIME, this.SESSION_BEGIN);

              /* compute total response time */
              t6 = event.getTimeStamp();
              val = (Long) event.get(this.BEFORE_REQUEST);
              t1 = val.longValue();
              buf.append("Request time = ");
              buf.append(t6 - t1);
              buf.append("¥r¥n");

              /* compute service time */
              val = (Long) event.get(this.SERVICE_END);
              t3 = val.longValue();
              val = (Long) event.get(this.SERVICE_BEGIN);
              t2 = val.longValue();
              buf.append("Service time = ");
              buf.append(t3 - t2);
              buf.append("¥r¥n");

              /* compute transform time */
              val = (Long) event.get(this.TRANSFORM_END);
              t5 = val.longValue();
              val = (Long) event.get(this.TRANSFORM_BEGIN);
              t4 = val.longValue();
              buf.append("Transform time = ");
              buf.append(t5 - t4);
              buf.append("¥r¥n");
      }

       public void sessionEnd(SessionEvent event) throws AbortServiceException {             // [84]
             StringBuffer buf = (StringBuffer) event.getSession().getAttribute( AttributeCategory.RUNTIME,
this.SESSION_BEGIN);
             System.out.println(buf.toString());
             System.out.println(event.toString());
      }

      public void afterSession(SessionEvent event) throws AbortServiceException {
             System.out.println (event.toString());
      }

      public void requestError(RequestEvent event) throws AbortServiceException {
             System.out.println(event.toString());
```

```
    }

    public void responseError(ResponseEvent event) throws AbortServiceException {
          System.out.println(event.toString());
    }

}
```

このサンプルでは、Session、Request および Response のすべてのイベントをリスニングするサンプルのリスナーについて説明しています。ここでは、セッションを使用して、同じセッション内にあるすべてのリクエストをグループ化する方法を示しています。29 行目の `sessionBegin()` メソッドでは、セッション内のすべてのイベントを記録するための大きな文字列バッファを作成します。84 行目の `sessionEnd()` メソッドでは、セッションの最後に、セッションのログが格納された文字列バッファを出力します。イベント・オブジェクトに配置される値は、イベント・ソースのライフ・サイクル全体にわたって存続し、後続イベントの実行時に取り出すことができます。また、リスナーによって値を Request オブジェクトまたは Session オブジェクトの RUNTIME 属性カテゴリに配置することもできます。この両方の手法によって、リスナーは個々のイベント・ソースのイベントを相互に関連付けてトレースできます。このサンプルでは、リスナーは各イベントのタイムスタンプをイベント・オブジェクトに格納しています。54 行目の `afterRequest()` メソッドに示すように、このタイムスタンプはリクエストの最後に取り出され、リクエストの合計応答時間、サービス時間および変換時間を計算するために使用されます。

リスナーのデプロイ リスナーをデプロイするには、すべての `.class` ファイルを classes ディレクトリにコピーします。デフォルトのディレクトリは次のとおりです。$ORACLE_HOME/wireless/server/classes.

MCS のリバース・プロキシ、URL リライト、キャッシュおよび圧縮

MCS Runtime は、文書を解析してレスポンス文書内の URL をリライトした後に、文書を変換してデバイスに配信します。MCS は、チャネル・プロトコル、ターゲット・ホスト名およびポート番号も含めてレスポンス文書内のすべての URL を置換するため、文書内のリンクに続く後続のリクエストをすべて傍受できます。このため、MCS はデバイスに対するリバース・プロキシ・サーバーとして機能します。

MCS Runtime は、MCS セッションで当初の URL をキャッシュし、それぞれの URL を、*PAsid* パラメータと *PAckey* パラメータのみで構成される短い URL に置換します。*PAsid* パラメータは、MCS Runtime セッション ID を指定します。また、*PAckey* パラメータは、MCS セッションの URL を参照するキーを指定します。MCS セッション ID、URL キャッシュ、キャッシュ・キーおよび Cookie は、永続的な状態の MCS セッションに含まれます。MCS Runtime では、MCS セッションでキャッシュされた当初の URL のパラメータを使用して傍受するリクエストを修正します。

MCS は、リバース・プロキシで仮想ブラウザとして機能し、MCS セッションでキャッシュされた Cookie に添付する、ターゲット・ホストへの HTTP リクエストとポート番号を生成します。MCS での URL のキャッシュおよびリライト方式によって、URL に多数の非表示フィールドが含まれる場合でも高い圧縮率を実現できます。

MCS 仮想ブラウザ・モデル

リバース・プロキシ・サーバーとして機能する MCS は、チャネル・プロトコル、ターゲット・ホスト、ポート番号、フォームまたは問合せパラメータを含むすべての URL をリライトするため、デバイスからのすべてのリクエストのプロキシとして機能できます。このため、MCS では、マルチチャネル・プロキシのリクエストを HTTP コンテンツ・プロバイダのリクエストに変換できます。コンテンツ・プロバイダへの HTTP リクエストを生成するとき、MCS はそのコンテンツ・プロバイダへの仮想ブラウザとして機能し、一般的なユーザー・エージェント・タイプを提示します。コンテンツ・プロバイダによるアプリケーションの記述が必要なのは、MCS 仮想ブラウザの一般的なユーザー・エージェント・タイプに対してのみです。これによって、マルチチャネル・アプリケーションの配信モデルが簡略化され、同時に、XHTML、XForms、CSS など業界標準のマークアップ言語に基づいた強力な開発モデルが提供されます。

仮想ブラウザとして機能する MCS は、コンテンツ・プロバイダ・アプリケーションからの URL リダイレクトに従うことができます。また、MCS は HTTP ヘッダー Referer もサポートしているため、外部アプリケーションはリクエストのコンテキストをトレースできます。MCS は、HTTP プロトコルと HTTPS プロトコルの両方を使用でき、他の永続 Cookie とともにセッション Cookie をサポートします。この Cookie は永続的な MCS セッションに含めることができます。MCS は、デバイスが MCS Runtime へのアクセスに使用するメソッドに応じて、GET メソッドまたは POST メソッドのいずれかを使用します。MCS Runtime がデフォルトで GET メソッドを使用するのは、HTTP チャネルがデバイスで使用されていない場合のみです。

MCS Runtime は、HTTP レスポンス・コード 301 ～ 305 などのリダイレクト・レスポンス・コードを検出し、HTTP Location ヘッダーに指定されたリダイレクト URL に従います。

MCS は、POST ベースのリダイレクトもサポートできます。POST ベースのリダイレクトを送信するには、コンテンツ・プロバイダは、次の jsp ファイルに示すように、値を true に設定した HTTP ヘッダー x-oracle-mobile-redirect と OracleAS Wireless XML フォームをレスポンスのコンテンツとして送信する必要があります。3 行目で POST リダイレクトが URL http://OracleAS Wireless.oracle.com に送信されます。param1=value1 は、URL に転送データとして渡されます。

```
<%
response.setHeader("x-oracle-mobile-redirect", "true");
response.setHeader("Content-Type", "tex/vnd.oracle.mobilexml");  // [3]
%>
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>
<!DOCTYPE SimpleResult PUBLIC "-//ORACLE//DTD SimpleResult 1.1.0//EN"
"http://xmlns.oracle.com/ias/dtds/SimpleResult_1_1_0.dtd">
<SimpleResult>
   <SimpleContainer>
      <SimpleForm name="ProcessSignOnForm" mimetype="text/vnd.oracle.mobilexml"
target="http://OracleAS Wireless.oracle.com/MyApp" method="post">
         <SimpleFormItem name="param1" value="value1" type="hidden"/>
      </SimpleForm>
   </SimpleContainer>
</SimpleResult>
```

MCS は、HTTP ヘッダー Referer の参照 URL を送信します。コンテンツ・プロバイダはこのメカニズムを使用して、現行のリクエストのコンテキストをトレースします。デフォルトでは、Referer ヘッダーは送信されませんが、後述する OracleAS Wireless XML の属性 *sendreferer* を使用して Referer ヘッダーの送信が必要であることが示されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<SimpleResult>
    <SimpleContainer>
        <SimpleHref target="HelloWorld.xml" sendreferer="true">Send
Referer</SimpleHref>
         <SimpleHref target="HelloWorld.xml" sendreferer="false">Don't Send Referer
</SimpleHref>
    </SimpleContainer>
</SimpleResult>
```

MCS では、HTTPS プロトコル・ベースの URL にアクセスできます。HTTPS を使用する前に、システム・マネージャのサイト構成を使用してクライアント資格証明を構成する必要があります。MCS では、HTTP プロキシ・サーバーを介して外部 URL にアクセスできます。プロキシ設定は、システム・マネージャのサイト構成を使用して指定できます。

MCS Runtime では、Netscape によるバージョン 0（ゼロ）の Cookie 仕様が実装されます（http://www.netscape.com/newsref/std/cookie_spec.html）。また、MCS Runtime では、外部 URL から送信された Cookie が MCS Runtime セッションに格納され、セッションから取り出された関連 Cookie が、HTTP URL リクエストとともにコンテンツ・プロバイダに送

信されます。この Cookie は、MCS セッションが有効であるかぎり有効です。永続 Cookie をサポートするには、永続的な MCS セッションを使用可能にする必要があります。

MCS では、コンテンツ調整を実行して、標準の XHTML、XForms および CSS マークアップ言語で記述された Web コンテンツをデバイス固有のマークアップ言語に調整します。MCS では、高度なアルゴリズムを使用して、User-Agent、Accept などの HTTP ヘッダーの属性、およびオプションのデバイス識別と物理デバイス・モデル（ユーザーがデバイスを登録するときに指定）を検索することで、デバイスとネットワーク機能を判別します。また、MCS は、新しいデバイスがリクエストとともに発行したデバイス機能があるかどうかも確認します。MCS は、デバイスのナレッジ・ベースにあるデバイス・モデルのレパートリを使用して、最適なコンテンツ調整をレンダリングします。

Wireless and Voice Portal

OracleAS Wireless には、自己完結型ポータルを作成するために必要なコンポーネントとして、ユーザーとデバイスのプロビジョニング、ユーザー管理、コンテンツとサービスの管理、シングル・サインオン認証、認可、デバイス登録、ユーザー作業環境管理、エンド・ユーザーのパーソナライズ・ポータル、請求システム統合などの機能も備えています。Wireless and Voice Portal は、ユーザー・リポジトリ、Oracle Internet Directory およびシングル・サインオン資格証明を Oracle Application Server Portal 製品と共有し、メインのポータルと相互運用できます。Wireless and Voice Portal Runtime では、デバイスからの各リクエストは、有効な Runtime セッションのコンテキスト内で処理されます。セッション所有者は Guest ユーザー、つまり匿名ユーザーの場合がありますが、匿名デバイスからのリクエストも追跡され、個々の Runtime セッションに割り当てられます。

デバイスの識別

OracleAS Wireless と音声ポータル Runtime では、デバイスで使用可能な識別子を使用して各デバイスを一貫して識別できるように、Oracle Application Server OID ユーザー・リポジトリ内で仮想ユーザーが自動的にプロビジョニングされます。デバイス識別子がリクエストに存在する場合は、仮想ユーザーの Runtime セッションがオープンされます。デバイス識別子の基礎として、Mobile Identification Number（MIN）、Mobile Subscriber ISDN（MSISDN）、Ipv6 アドレス、Electronic Serial Number（ESN）などの固有のデバイス識別子を使用できます。また、デバイス識別子は WAP ゲートウェイによってデバイスにプロビジョニングされることもあります。WAP Client ID Specification（http://www.wapforum.org/）には、デバイス識別子をサポートするための標準スキームが定義されています。リクエストにデバイス識別子が指定されていない場合、OracleAS Wireless Runtime では可能であれば永続 Cookie を使用してデバイスに識別子がプロビジョニングされます。

Wireless and Voice Portal Runtime では、仮想ユーザーを使用したパーソナライズを容易にするためにのみデバイス識別子が使用されます。仮想ユーザーを使用してオープンされた Runtime セッションは、リポジトリ内のパーソナライズ済プリセットや作業環境プロファイルなどの情報にアクセスできます。また、デバイス識別子を使用すると、セッションが失効しないかぎり、デバイスでユーザー用の同じ Runtime セッションに再接続できます。デバイス識別子は、ワイヤレス・デバイスや音声デバイスのセッション管理の堅牢さを高め、断続

的な接続障害の発生時にもサービスを継続可能にします。また、ユーザーはポータルへの接続と接続の間に、コンテキストを失わずに通話できます。

デバイス識別子は認証手段ではありません。仮想ユーザー用の Runtime セッションは認証されませんが、ユーザーはパーソナライズされたポータルにアクセスできます。ユーザーが認証済セッションを確立できるのは、Wireless and Voice Portal に登録している場合のみです。ユーザー名とパスワードは登録時に指定できます。ユーザーのパーソナライズ・プロファイルとプリセットは、登録後も使用できます。また、登録には、E-Wallet や金融取引サービスなど、セキュアなサービスへのアクセス権を付与する認証を可能にするという利点があります。

仮想ユーザーの概念

Wireless and Voice Portal Runtime では、デバイスで使用可能な識別子を使用して各デバイスを一貫して識別できるように、Wireless リポジトリ内で仮想ユーザーが自動的にプロビジョニングされます。仮想ユーザー・オプションにより、デバイス所有者は、ユーザー操作を強化するポータルのパーソナライズ機能に即時にアクセスできます。この機能によって、新たな WAP クライアント ID 規格を使用している電話会社やエンタープライズ・ポータル管理者のプロビジョニング処理が自動化されます。

デバイス所有者は Wireless and Voice Portal に登録し、認証を通じてセキュアなサービスへのアクセス権を取得できます。この登録は、デバイス所有者がセットアップ・メニューから実行できます。この自己プロビジョニング登録機能により、管理タスクがさらに簡素化されます。仮想ユーザー・サポート機能を持つデバイスを使用すると、登録済ユーザーは、セキュアなサービスから認証を要求されるまでシステムにサイン・オンし続けなくても、Wireless and Voice Portal に接続し、パーソナライズされたサービスにアクセスできます。仮想ユーザー機能によって、デバイスのアクティビティを仮想識別子で識別できるため、ポータルのアクセス可能性が改善されるのみでなく、ポータル操作のデータ・マイニング機能も強化されます。

仮想ユーザー機能は、サイト単位の構成パラメータ設定 `wireless.virtualuser.enabled=false` で使用禁止にできます。このプロパティは、「システム・マネージャ」→「サイト」→「ユーザー・プロビジョニング」コントロール・パネルで「仮想ユーザーを使用可能にする」オプションを使用して変更できます。 仮想ユーザー機能を使用禁止にした場合、またはデバイスでデバイス識別子がサポートされていない場合、セッションは Guest ユーザーを使用してオープンされます。このユーザーは、リポジトリ内でプロビジョニングされている必要があります。Wireless and Voice Portal ブートストラップ・リポジトリには、匿名ユーザーである Guest が含まれています。

Wireless and Voice Portal Runtime オブジェクトに直接アクセスするアプリケーションでは、`oracle.panama.model.User` の `getUserType()` メソッドから戻される `oracle.panama.model.UserType` の値をチェックできます。Runtime セッションの User は、`oracle.panama.rt.Session` の `getUser()` メソッドから取得できます。外部コンテンツ・プロバイダは、ユーザー・タイプ情報を HTTP ヘッダー属性 *x-oracle-user.userkind* から取得できます。この属性の可能な値は anonymous、virtual または registered です。

認証と認可

認証済セッションが必要なサービスのアプリケーション・プログラムでは、URL に *PAlogin=true* パラメータを追加する必要があります。Wireless and Voice Portal Runtime は、Runtime セッションが未認証の場合に、サービス・リクエスト内の URL パラメータの中から *PAlogin=true* パラメータが検出されると、ユーザーの認証を試行します。この認証プロセスは、リクエストされたサービスが Runtime によって起動される前に実行され、通常はユーザーが Oracle Application Server Single Sign-On（SSO）Server にユーザー名とパスワードを入力する必要があります。*PAlogin* パラメータで認証プロセスが起動した後も、セキュアなサービス用のアプリケーション・プログラムではセッションが認証されていることを検証する必要があります。Wireless and Voice Portal Runtime オブジェクトに直接アクセスするアプリケーションでは、`oracle.panama.rt.Session` インタフェースの `isUserAuthenticated()` メソッドを使用できます。外部コンテンツ・プロバイダは、HTTP ヘッダー属性 *x-oracle-user.authkind* から情報を取得できます。この属性の値は authenticated または unauthenticated です。

また、アプリケーションでは、セッションが SSL、TLS、WTLS のうちどのチャネルで保護されているかもチェックできます。Wireless and Voice Portal Runtime オブジェクトに直接アクセスするアプリケーションでは、`oracle.panama.rt.Request` インタフェースの `isSecure()` メソッドを使用できます。外部コンテンツ・プロバイダは、HTTP ヘッダー属性 *x-oracle-device.secure* を介して条件 `isSecure()` を取得できます。この属性の値は true または false です。

サービスへのアクセスの認可は、すべての認証済または未認証セッションについてリクエストごとに実行されます。この認証により、セッション・ユーザーがサービスへのアクセス権限を持っていることが確認されます。デフォルトの認可ポリシーでは、セッションが認証済かどうかは区別されません。仮想または登録済ユーザーによる未認証セッションは、認証済セッションと同様の可視性を持ちます。したがって、アプリケーションでは認証を規定するために *PAlogin* パラメータを適用する必要があります。

グローバリゼーション（NLS）サポート

マルチチャネル・サーバーは、デバイス・ブラウザとバックエンド Web アプリケーションの間にデプロイされる中間層です。マルチチャネル・サーバーは、HTTP（HTTPS）プロトコルを介してデバイスやアプリケーションと通信します。

マルチチャネル・サーバーは、リクエスト / レスポンスの文字コード / デコードを正しく行うために、デバイスとアプリケーションの両方で使用する文字コードを認識する必要があります。マルチチャネル・サーバーは、各 HTTP ホップの文字コードを次の順序で処理します。

1. デバイスからの最初のリクエスト : MCS は、デバイス・ブラウザの「受け入れた文字コード」属性を使用して、ブラウザから送信されたリクエスト・パラメータをデコードします。
2. MCS からアプリケーションへの最初のリクエスト : MCS は、アプリケーションで必要な文字コードを認識しません。この場合、MCS は、アプリケーションへの最初のリクエストで UTF-8 文字コードを使用します。
3. アプリケーションからの最初のレスポンス : MCS は、Hypertext Transfer Protocol（HTTP）1.1 仕様に従って、受信者に送信されるエンティティ本体のメディア・タイプを検出します。送信者は、この仕様に従って *Content-Type* HTTP ヘッダーを追加し、コンテンツのキャラクタ・セットを示す *charset* 値を追加する必要があります。次に例を示します。

   ```
   Content-Type: application/vnd.oracle.xhtml+xforms; charset=ISO-8859-4
   ```

   Content-Type ヘッダーつまり *charset* 値を指定しない場合、MCS は ISO-8859-1 文字コードを使用します。

   > **注意：** この文字コード値は MCS で記憶され、アプリケーションへの後続のリクエストで使用されます。

4. MCS からの最初のレスポンス : MCS は、デバイス・ブラウザの「受け入れた文字コード」属性を使用して、ブラウザから送信されたリクエスト・パラメータをデコードします。
5. デバイスからの後続のリクエスト : MCS は、デバイス・ブラウザの「受け入れた文字コード」属性を使用して、ブラウザから送信されたリクエスト・パラメータをデコードします。
6. MCS からアプリケーションへの後続のリクエスト : MCS は、前のレスポンスでアプリケーションから送信された内容と同じ文字コードを使用します。
7. アプリケーションからの後続のレスポンス : MCS は、ステップ 3 で説明したロジックを使用します。アプリケーションでは、文字コードを再指定する必要があります。アプリケーションで別のコードを使用することは可能ですが、一般的ではありません。

8. MCS からの後続のレスポンス : MCS は、デバイス・ブラウザの「受け入れた文字コード」属性を使用して、ブラウザから送信されたリクエスト・パラメータをデコードします。

最初のリクエスト / レスポンスの後は、後続のすべてのリクエスト / レスポンスについてステップ 5 ～ 8 を繰り返します。

つまり、MCS はデバイスとの通信時に、常に「受け入れた文字コード・デバイス」属性を使用して文字コード / デコードを行います。また、アプリケーションとの通信時は、常に Content-Type HTTP ヘッダーの *charset* 値を使用します。このルールの唯一の例外は、MCS で UTF-8 文字コードを使用している場合の、MCS からアプリケーションへの最初のリクエストです。

データ・モデルの変更

OracleAS Wireless サービスの概要

様々なサービスによって、エンド・ユーザーは OracleAS Wireless の機能にアクセスできます。これらのサービスは、コンテンツ・ソースと配信ターゲットの間のリンクを表します。サービスにより、特定のデータ・ソースが（アダプタを介して）様々なデバイスに連結されます。

次のように様々なタイプのサービスがあります。

- MasterService - サービスの実際の実装を提供します。MasterService では、サービスに使用されるアダプタとサービス固有のパラメータが指定されます。
- Link - サービスへのポインタです。ほとんどの場合、Link は MasterService をエンド・ユーザーに公開し、MasterService のパラメータをカスタマイズするために使用されます。
- Module - 既知の URL を持つ MasterService へのポインタです。
- Folder - 他の Folder など、他のサービスのコンテナです。サービス・ツリーの作成に使用されます。
- ExternalLink - 外部リソースを指し示すサービスです。

MasterService

MasterService はサービスの実際の実装であり、基本的な Wireless 機能を提供します。各 MasterService は 1 つのアダプタに基づいています。MasterService により、アダプタの初期パラメータ、入力パラメータおよび出力パラメータの値が設定されます。各 MasterService では、使用するアダプタの独自インスタンスが作成されます。したがって、複数のサービスが同じタイプのアダプタを使用し、それぞれが独自サービス固有の引数値を渡すことができます。

すべての MasterService の作成に HTTP アダプタを使用することをお薦めします。これにより、JSP や他の Web テクノロジを使用して、サービスのビジネス・ロジックを柔軟に実装できます。

Link

Link は、パラメータの値をオーバーライドして既存のサービスをさらにカスタマイズするために使用されます。

Link サービスが起動すると、OracleAS Wireless サーバーはパラメータを Link が指し示すサービスのパラメータとマージして、そのサービスを起動します。

Link は、サービスをユーザー・サービス・ツリー形式で適切に編成するためにも使用されます。これにより、同じサービスに異なる名前を付けて、異なるフォルダ（サービス・ツリーの異なるレベル）で柔軟に公開できます。パラメータ値をオーバーライドしない場合は、Link を起動すると、それが指すサービスが起動します。

Module

Module は、予約済の仮想 URL（OMP URL、つまり omp://my.module）を使用する Wireless サービスです。

Module は、任意のアプリケーションまたはモジュールからコールでき、他のアプリケーションまたはモジュールに制御を戻すように指示できます。コールは任意のレベルまでネストできます。この双方向リンク・メカニズムにより、アプリケーションの迅速なアセンブリが可能になります。

Module と通常のサービスとの重要な違いは、Module は完了後に戻るために必要なサービスに関する情報を受け取ることです。これは、常に Module のコール元であるとはかぎりません（Module のコール元は、異なるサービスに戻るための Module を必要とする場合があります）。

Folder

Folder は、他のサービスのコンテナです。Folder は、ユーザーがアクセス可能なサービスをサービス・ツリー形式で適切に編成するために使用されます。Folder コンテンツは、そのレンダリング・サービス、つまり各 Folder に関連付けられている特殊サービスを起動することで表示されます。

システム・レンダリング・サービスでは、指定のソート規則に従って Folder の子サービスが表示されます。

オプションで、アイコンとオーディオ・ファイルを指定して、フォルダ・コンテンツにサービス・リンクが表示されるときやサービスの起動時に表示または再生できます。

ExternalLink

ExternalLink は、外部リソースを指す Wireless サービスです。通常、外部リソースは、ターゲット・デバイスでサポートされているフォーマットでコンテンツをサービスする Web ページです。

OracleAS Wireless では、ExternalLink のターゲットのコンテンツは処理されません。そのため、他の Wireless サービスと同様に、ExternalLink サービスを使用できないターゲット・デバイスがあります。ほとんどの場合、ExternalLink はエンド・ユーザーがサービス・デザイナではなく Customization ポータルで設定します。

アクセス制御

アクセスに関連するサービスには、次の 2 種類があります。

- ユーザー・プライベート・サービス : 個々のユーザーのみがアクセス可能です。
- 共有サービス : 複数のユーザーがアクセス可能です。

この 2 種類のサービスには、異なる規則が適用されます。

- ユーザー・プライベート・サービスは、ユーザーのホーム・サービス・ツリーに常駐するサービスです。ユーザーは、この種のすべてのサービスにアクセスできます。他のユーザーのプライベート・サービスにはアクセスできません。
- これに対して、共有サービスには複数のユーザーがアクセスします。アクセスは、ユーザー・グループ・サービスの関係により制御されます。グループにサービスを割り当てると、そのグループのユーザーはすべてサービスにアクセスできます。

フォルダ・レンダラ

概要

フォルダ・レンダラは、フォルダのコンテンツをレンダリングする、OracleAS Wireless の Runtime コンポーネントです。エンド・ユーザーがカスタマイズできるように、フォルダ・レンダラ・コンポーネントのロジックはフォルダ・レンダラ・フックの形式で具体化されています。

詳細は、`oracle.panama.rt.hook.FolderRendererHook` Java インタフェースを参照してください。

OracleAS Wireless には、デフォルトでフォルダ・レンダラ・フックの実装が用意されています。デフォルト実装の構成は次のとおりです。

- OC4J アダプタに基づいたシステム・マスター・サービス。`omp://oracle.iasw.folder.renderer` の仮想 URL が含まれます。
- マスター・サービスの検索と起動を行う Java クラス。この Java クラスの名前は `oracle.panama.rt.hook.FolderRendererPolicy` で、インタフェース FolderRendererHook を実装します。
- 一連の JSP ページ。マスター・サービスは、`iaswfr/FolderRenderer.jsp` の相対 URL を使用して 1 つの JSP を指し示します。この JSP は、フォルダのコンテンツをレンダリングできる一連の JSP ページへのエントリ・ポイントとして機能します。
- JSP ページによって起動するメソッドのライブラリが含まれるユーティリティ Java クラス。この Java クラスの名前は `oracle.panama.rt.hook.FolderRendererUtil` です。

JSP ベースのフォルダ・レンダラのフレームワークを使用すると、エンド・ユーザーは簡単にカスタマイズできるようになります。JSP で記述されたフォルダ・レンダラのロジックによって、エンド・ユーザーは、Java コードを再コンパイルしないで JSP ページを簡単に変更できます。JSP ベースのフォルダ・レンダラを使用すると、次に示すように、いくつかのレベルでカスタマイズを実行できます。

- ユーザーは、デフォルト実装に含まれている JSP を変更できます。
- ユーザーは、一連の JSP ページを新たに作成し、その新しい JSP ページを指し示すようにシステム・マスター・サービスの相対 URL を変更できます。
- ユーザーは、マスター・サービスと一連の JSP ページを新たに作成できます。この場合、ユーザーは、仮想 URL の `omp://oracle.iasw.folder.renderer` が指し示すデフォルトのマスター・サービスを新しいマスター・サービスに置換します。
- ユーザーは、インタフェース `FolderRendererHook` の独自の実装を記述できます。このインタフェースのデフォルト実装は `oracle.panama.rt.hook.FolderRendererPolicy` です。

ユーザーは、フォルダ・レンダラをカスタマイズするために、JSP ページの構造と実行フローを理解する必要があります。

JSP ページの構造

JSP ページの論理構造を次に示します。

- エントリ・ポイントとして機能するトップレベルの JSP ページが FolderRenderer.jsp です。この FolderRenderer.jsp から、接続デバイスのデバイス・カテゴリをチェックし、第 2 レベルの適切な JSP ページを追加します。
- 第 2 レベルの JSP ページは XXXRenderer.jsp という名前で、XXX は接続デバイスのデバイス・カテゴリです。XXXRenderer.jsp 自体には、XXXHeader.jsp、XXXBody.jsp および XXXFooter.jsp の 3 つの JSP ページが含まれています。

たとえば、WAP 電話からのリクエストの場合、FolderRenderer.jsp には MicroBrowserRenderer.jsp が含まれ、その MicroBrowserRenderer.jsp には、MicroBrowserHeader.jsp、MicroBrowserBody.jsp および MicroBrowserFooter.jsp が含まれます。

実行フロー

フォルダ・レンダラのデフォルト実装の実行フローは、次のとおりです。

1. OracleAS Wireless Runtime は、フォルダをレンダリングする準備ができると、`oracle.panama.rt.hook.FolderRendererPolicy invoke()` メソッドをコールします。
2. `invoke()` メソッドは、仮想 URL の `omp://oracle.iasw.folder.renderer` のハードコード値を使用し、マスター・サービスを検索して起動します。マスター・サービスは JSP として実装され、`iaswfr/FolderRenderer.jsp` の相対 URL が含まれます。
3. FolderRenderer.jsp がコールされ、次の処理が実行されます。
   - 必要なすべての情報（ServiceContext、セッション、ユーザー、デバイスなど）をリクエストから取得して格納します。この情報は、FolderRenderer.jsp に含まれる他の JSP ページで使用されます。
   - 接続デバイスが属しているデバイス・カテゴリをチェックし、適切な JSP ページ（前述の説明を参照）を追加します。
4. 追加された JSP ページでは、現行フォルダのコンテンツがレンダリングされます。

ブックマーク

OracleAS Wireless サーバーを使用すると、サーバー側のユーザー・ブックマークを管理できます。各ブックマークは、デバイス固有のマークアップ言語で戻されたコンテンツを持つデータ・ソース（URL）を参照します。OracleAS Wireless サーバーでは、ユーザーと管理者はブックマークをユーザー・サービス・ツリーの任意の位置に配置できます。

各ブックマークは 1 つの論理エントリで、異なるマークアップ言語に対応した複数の URL を含めることができます。たとえば、Yahoo ブックマークには、HTML マークアップ言語用の URL `http://www.yahoo.com`、および WML マークアップ言語用の URL `http://wap.yahoo.com` を含めることができます。ユーザーが、WML マークアップ言語をサポートするデバイスから Yahoo ブックマークにアクセスすると、`http://wap.yahoo.com` URL のコンテンツがデバイスに戻されます。また、ユーザーが、HTML 言語をサポートするデバイスから Yahoo ブックマークにアクセスすると、`http://www.yahoo.com` URL のコンテンツが戻されます。

各マークアップ言語は、MIME タイプによって一意に識別されます。OracleAS Wireless サーバーでブックマークを格納するとき、内部的には、<URL, markup language> の組合せを <URL, MIME-type> の組合せとして格納します。

ブックマークは Wireless トランスコーディング API と統合されています。この API によって、WML コンテンツ（`text/vnd.wap.wml` MIME タイプ）を MobileXML に変換し、さらに OracleAS Wireless サーバーでサポートされているデバイス固有のマークアップ言語に変換できます。

URL に関連付けられた MIME タイプに応じて、一部の URL はユーザーのデバイスから直接アクセスされ、他の一部は OracleAS Wireless を介してアクセスされます。現在、OracleAS Wireless サーバーを介してアクセスできるのは、`text/vnd.wap.wml` MIME タイプに関連付けられた URL のみです。トランスコーディング API でサポートする入力マークアップ言語が増えるに従って、アクセスできる URL も増える予定です。

さらに、ブックマークの任意の URL をデフォルト URL としてマークできます。デバイスでサポートされるマークアップ言語に対応した URL がないブックマークにデバイスがアクセスすると、OracleAS Wireless サーバーはデフォルト URL を起動し、そのコンテンツを MobileXML にトランスコードし、さらに、デバイスでサポートするマークアップ言語に変換します。

デフォルト URL から戻されたコンテンツのマークアップ言語は、Wireless トランスコーディング API によるサポートが必要です（つまり、デフォルト URL で使用できるのは、WML の `text/vnd.wap.wml` MIME タイプのコンテンツのみです）。

ユーザー・デバイスとブックマークでサポートされる MIME タイプ（つまり、関連付けられた URL がある MIME タイプ）に応じて、次のいくつかの方法で、ブックマークに格納された URL にアクセスできます。

- ユーザーのデバイスが WML コンテンツ・タイプをサポートし、text/vnd.wap.wml MIME タイプに関連付けられた URL がある場合。この場合、ユーザーのデバイスは OracleAS Wireless サーバーを介して URL にアクセスします。関連するすべての URL が OracleAS Wireless サーバーを指し示すようにリライトされた場合を除き、WML コンテンツは変更されません。後続のすべてのリクエストは、OracleAS Wireless サーバーを介してアクセスします。
- ユーザーのデバイスが WML 以外のマークアップ言語をサポートし、その MIME タイプに対応する URL がブックマークにある場合。この場合は、OracleAS Wireless サーバーを介さずに（つまり、ユーザーは Wireless ポータルを離れて）ユーザーのデバイスから URL に直接アクセスします。
- ユーザーのデバイスが WML 以外のマークアップ言語をサポートし、そのマークアップ言語に関連付けられたブックマークに URL はないが、デフォルト URL（text/vnd.wap.wml MIME タイプ）がある場合。この場合、ユーザーのデバイスはリクエストを OracleAS Wireless サーバーに送信します。サーバーでは、WML コンテンツをフェッチし、WML を MobileXML にトランスコードします。次に、その MobileXML をデバイス固有のマークアップ言語に変換し、レスポンスをデバイスに返信します。後続のすべてのリクエストでこれを繰り返します。
- ユーザーのデバイスが、対応する URL がブックマークにないマークアップ言語をサポートし、デフォルト URL がない場合。この場合、デフォルトの FolderRenderer はブックマークを起動するリンクを表示しないため、ユーザーに対してブックマークは表示されません。

OracleAS Wireless Tools を使用したブックマークの作成と編集

ユーザーは、OracleAS Wireless Tools（または Customization Portal）を使用して、ブックマークを作成、編集および削除できます。

Model API: 一般的な使用方法

OracleAS Wireless リポジトリは Model-View-Control（MVC）アーキテクチャのモデルで構成されていますが、OracleAS Wireless Runtime レイヤーは MVC のコントローラに該当します。`oracle.panama.model` パッケージ内の Repository Model API を使用すると、OracleAS Wireless リポジトリ内の永続オブジェクトを作成、削除、変更および問い合せるアプリケーションを開発できます。

OracleAS Wireless リポジトリでは、オブジェクト間に組織構造が適用されます。たとえば、1 人のユーザーを複数のグループに所属させることができます。各ユーザーは、1 つ以上のロールに割り当てられます。ユーザーは、所属するグループからアクセス可能なサービスにアクセスできます。ただし、ユーザー・インタフェースの実装では、Oracle Internet Directory（OID）や Oracle Applications のユーザー・リポジトリ（AOL）など、外部のプロビジョニング・システムやリポジトリにアクセスし、エンタープライズ・ユーザー情報を管理し、ユーザーのロール、グループ・メンバーシップおよびそのユーザーがアクセス可能な特定のサービスを指定できます。

フォルダは、サービス・ツリーを構成するサービスのコンテナとして使用される特殊なサービスです。サービスまたはフォルダは、1 つ以上のグループに割り当てることができます。ユーザーは、DeviceAddress のコレクション、LocationMark のコレクション、カスタマイズ Profile のコレクション、高度なカスタマイズで使用される Presets の 1 つ以上のコレクションを所有できます。デフォルトの LocationMark とデフォルトの Profile は、ユーザーごとに割り当てることができます。Model API の Device インタフェースでは、対象となるデバイス・プロトコル（WAP、SMS、EMAIL など）を定義します。また、対象となるデバイスの物理的な特性（画面の幅と高さ、画面の列数と行数、ソフトキーの数など）も指定します。この特性はアダプタとトランスフォーマで使用できます。

Model API の対象ユーザーは、Customization Portal、ポートレット、カスタム・フック、リスナーおよびアプリケーションの開発者です。この場合のアプリケーションには、JSP、サーブレット、モジュール、および HTTP アダプタを介して起動する他の（URL アドレス可能な）リソースなどがあります。開発者は Model API を使用して、永続オブジェクトを操作するスタンドアロン・アプリケーションを開発することもできます。この種のインタフェースではリポジトリ内のデータ整合性が保たれますが、アクセス制御セキュリティは規定されません。Model API を介してリポジトリにアクセスするアプリケーションは、OracleAS Wireless Runtime レイヤー内の同じ認証および認可メカニズムでは認証または許可されません。Model API は、認証および認可ポリシーを開発してカスタマイズするために、トラステッド・コンポーネントによって使用されます。OracleAS Wireless Tools は、リポジトリに認証および認可されたアクセス制御を提供します。開発者は、Model API のインタフェースを使用してサービスを開発するときに十分な注意を払い、エンド・ユーザーがサービスを起動したときに予期しない問題が発生しないように、適切な対策を講じる必要があります。

データ・モデルのキャッシュと同期化

リポジトリ・オブジェクトは、Data Model API からアクセスされると、Java インスタンスのメイン・メモリーにキャッシュされます。これらのオブジェクトがメイン・メモリーから削除されるのは、TTL 間隔中に API を介してアクセスされなかった場合のみです。この間隔は、OracleAS Wireless Tools の「システム・マネージャ」→「サイト」→「ランタイム構成」コントロール・パネルの「キャッシュ・オブジェクト・ライフ時間（秒）」プロパティから構成できます。リポジトリ・オブジェクトが変更され、Java インスタンスの 1 つからリポジトリにコミットされると、他のすべての Java インスタンスでは変更後のオブジェクトがリポジトリから自動的に再ロードされます。キャッシュ同期スレッドの数は、OracleAS Wireless Tools の「システム・マネージャ」→「サイト」→「オブジェクト・キャッシュ同期」コントロール・パネルで指定できます。

インタフェースとインタフェース階層

ModelObject は、すべてのリポジトリ・オブジェクトに共通する動作とプロパティを表すルート・インタフェースです。このオブジェクトは、oracle.panama.model パッケージに格納されています。

Model API の継承階層

oracle.panama.model パッケージには、モデル・オブジェクトにアクセスできるように次の 3 つのロケータおよびファクトリ・オブジェクトも用意されています。

- MetaLocator: Model API 使用の開始ポイント。このコールから、ModelFactory および ModelServices インタフェース実装への参照を取得できます。
- ModelFactory: モデル・オブジェクトを作成するためのファクトリ。
- ModelServices: モデル・オブジェクトにアクセスするためのロケータ。

次に、様々なモデル・インタフェースについて簡単に説明します。インタフェースの詳細は、Model API 仕様を参照してください。

- Adapter: RuntimeAdapter のリポジトリ・コンテナであり、すべてのカスタム・アダプタで実装されるインタフェースです。この Adapter によって、RuntimeAdapter クラスがリポジトリに取り込まれ、RuntimeAdapter のロードと初期化がサポートされます。
- DeviceV2: 対象となるロジカル・デバイス・プロトコルの定義です。たとえば、WML 固有デバイスに対して WML11 を指定できますが、Nokia 固有の WML に対して WML_Nokia7110 を指定することもできます。その他の例には、SMS と EMAIL があります。DeviceV2 には Transformer オブジェクトが含まれています。
- User: ユーザー ID を表し、OracleAS Wireless ポータルでのパーソナライズを容易にします。
- Profile: ユーザーは、サービス・ツリーのカスタマイズを含む 1 つ以上の Profile を持つことができます。ユーザーの Profile では、フォルダ内のサービスの優先順位を指定できます。

- Group: ユーザーのコレクションです。Group は、特定のサービスをグループ・メンバーに対して公開するために使用されます。ユーザーは、所属するグループからアクセス可能なサービスにアクセスできます。
- Role: Group と同様に、Role もユーザーのコレクションです。ただし、Group は実行時にアクセスを制御するために使用されますが、Role は異なる Webtool へのアクセスを制御するために使用されます。
- Service: 抽象インタフェースで、サービス全般を処理します。Service には、次のサブインタフェースが含まれています。
  - MasterService: 最終的なサービスです。この MasterService は、他のすべてのサービスのテンプレートです。外部ソースとの通信には、常に Adapter を使用します。
  - Folder: ファイル・システムのディレクトリに類似しており、サブフォルダを含む他のサービスが格納されています。
  - ExternalLink: 外部 URL への論理参照です。1 つの ExternalLink では、チャネルごとに 1 つの URL を参照するか、または複数のチャネルで 1 つの URL を共有できます。
  - LocalModule: モジュール化可能な MasterService へのポインタです。
  - Link: 別の Link を含む他のサービスへのポインタです。Link は、MasterService のカスタマイズやアクセス可能な MasterService のプライベート・ツリー構造の作成に使用されます。Link は、最終的なマスター・サービスに至るサービス連鎖が保持するアクセス可能なパラメータをオーバーライドできます。
- LocationMark: 名前が付けられ、ジオコーディングされた物理アドレスを表す永続オブジェクトです。
- Transformer: すべての変換サブクラスに対するベース・インタフェースです。また、実際の変換実装（Java または XSL）のリポジトリ・コンテナです。Transformer は、oracle.panama.rt.xform.RtTransformer インタフェースを実装するカスタム・トランスフォーマ・クラスのロードと初期化を実行します。また、XSLT スタイルシート用の XSLT トランスフォーマを提供します。

  Transformer には、次のサブインタフェースがあります。

  - JavaTransformer: Transformer インタフェースを実装するクラスで、デバイスに依存しないマークアップ言語からデバイス固有のマークアップ言語への変換を処理します。また、oracle.panama.xform.RtTransformer クラスをリポジトリに取り込みます。Transformer は、oracle.panama.rt.xform.RtTransformer インタフェースを実装するカスタム・トランスフォーマ・クラスのロードと初期化を実行します。
  - XSLTransformer: デバイスに依存しないマークアップ言語からデバイス固有のマークアップ言語への変換を処理する XSLT スタイルシートを使用します。カスタム XSLT スタイルシートをリポジトリに取り込みます。また、XSLT スタイルシート用の XSLT プロセッサを提供します。

Data Model API を使用するサンプル・コード

次のサンプル・コードに、Model API のインタフェースを使用して OracleAS Wireless リポジトリに新規オブジェクトをプロビジョニングする方法を示します。この例ではサンプル・コードを取り込むためにスタンドアロン・クラスを使用していますが、アダプタ、フック、リスナーおよびサーブレットなど、他のタイプのコンポーネントを使用して Model API を示すことができます。この例には、Model API での検索、作成、削除およびコミット操作のみが示されていますが、必要なビジネス・ロジックは含まれていません。

サンプル・コードの詳細リストの横には、大カッコで囲まれた参照番号があります。この番号は、説明文にある大カッコで囲まれた参照番号に対応しています。

```
Use MetaLocator to get the ModelFactory and ModelServices (line [1]).
Use ModelFactory to create a new object.
Use ModelServices to search for an object.
MetaLocator metaLocator = MetaLocator.getInstance();
modelFactory = metaLocator.getModelFactory();
modelServices = metaLocator.getModelServices();
```

MetaLocator インタフェースを使用して、ModelFactory と ModelServices を参照します。このインタフェースの `getInstance()` メソッドは、MetaLocator のシングルトン・インスタンスを取得します。`getModelFactory` メソッドと `getModelServices` メソッドは、ModelFactory と ModelServices を参照します。

通常、新規オブジェクトを作成するには、オブジェクトがすでに存在しているかどうかを最初にチェックします。任意のオブジェクトを参照するには、ModelServices インタフェースとメソッド lookupX(Java.lang.String name) を使用します。X はオブジェクトのインタフェース名です。このサンプル・コードでは、新規ユーザーを作成するとき（新規ユーザー作成のコード・セクションは、2 行目から始まります）、次のコード行のように、最初に ModelServices インタフェースの lookupUser(userName) メソッドを使用してユーザーを参照します（3 行目）。

```
modelServices.lookupUser(userName);
```

この参照操作は、リポジトリ内で新規永続オブジェクトを作成する前の最初の手順です。lookupUser(userName) メソッドは、名前でユーザーを検索し、ユーザー名が見つかると、User オブジェクトを戻します。ユーザー名が見つからない場合、メソッドは PanamaRuntimeException をスローします。

次に、ユーザーが所属する（または所属する必要がある）グループがすでに存在しているかどうかをチェックします（4 行目）。任意のオブジェクトを参照する場合の規則に従って、ModelServices インタフェースと lookupGroup(groupName) メソッドを使用し、グループ名を指定してグループを参照します。グループが見つかると、メソッドは Group オブジェクトを戻します。グループが見つからない場合、メソッドは PanamaRuntimeException をスローします。

ユーザーとグループがすでに存在しているかどうかをチェックした後、次のように新規ユーザー・オブジェクトを作成します（5 行目から 6 行目まで）。

```
{
     user = modelFactory.createUser(userName, groups);
} else {
     user = modelFactory.createUser(userName);
}
user.setPassword(userPassword);
user.setEnabled(true);
```

新規に作成したユーザーは必ず保存してください。新規にオブジェクトを作成した後は、次のように、そのオブジェクトを保存する必要があります（7 行目）。

```
modelFactory.save();
```

Save は、現行スレッド内で作成または変更されたすべてのオブジェクトに適用されます。オブジェクトは永続記憶域に保存され、トランザクションがコミットされます。作業内容を保存できない場合は、メソッドが PanamaException をスローします。

サンプル・コードの searchUser() メソッド（8 行目）は、User オブジェクトの検索方法を示しています。一連のユーザー（名前が文字 B で始まるすべてのユーザーなど）を列挙するには、findUsers メソッド（10 行目）から戻される ResultSetEnumeration（9 行目）を使用します。メソッド findUsers は、名前にパターン一致を使用します。サンプル・コードの詳細リストの 11 行目と 12 行目も参照してください。

ResultSetEnumeration（13 行目）をクローズして、データベース・カーソルを解放します。クローズしないと、ResultSetEnumeration はオープンしたままになります。

ユーザーを削除するには、14 行目のサンプル・コード・セクションに従って deleteUser メソッドを使用します。ユーザー名は、15 行目で正しく指定してください。ModelServices.lookupUser() メソッドは、例外をスローしてパターン一致テンプレートを拒否します。ユーザー・オブジェクトは 16 行目で削除されます。

```
import Java.util.Vector;

import oracle.panama.PanamaException;
import oracle.panama.PanamaRuntimeException;

import oracle.panama.model.MetaLocator;
import oracle.panama.model.ModelFactory;
import oracle.panama.model.ModelServices;
import oracle.panama.model.ResultSetEnumeration;
import oracle.panama.model.User;
import oracle.panama.model.Group;


/**
 * This is a sample program demonstrates the usage of the model API.
 */
public class SampleModelClient {
```

```
    private ModelFactory modelFactory;
    private ModelServices modelServices;

    public SampleModelClient() {
        MetaLocator metaLocator = MetaLocator.getInstance();       [1]
        modelFactory = metaLocator.getModelFactory();
        modelServices = metaLocator.getModelServices();
    }

    /**
     * Get all group names
     */
    private String[] getGroupNames() throws PanamaException, PanamaRuntimeException
{
        String[] names;
        ResultSetEnumeration result = null;
        try {
            // Find all user groups - use a wildcard for the name expression
            result = modelServices.findGroups("*");
            Vector buffer = new Vector();
            while (result.hasMoreElements()) {
                Group group = (Group)result.next();
                String name = group.getName();
                buffer.addElement(name);
            }
            names = new String[buffer.size()];
            buffer.copyInto(names);
        } catch (PanamaRuntimeException ex) {
            throw ex;
        } finally {
            if (result != null) {
                result.close();
                result = null;
            }
        }
        return names;
    }

    /**
     * Create a new user.
     */
    private void createUser(String userName, String userPassword, String groupName)
[2]
                            throws PanamaException, PanamaRuntimeException {
        try {
            // First check if the user does not already exists
```

```
            modelServices.lookupUser(userName);                    [3]
            // If we are here the user must already exists
            return;
        } catch (PanamaRuntimeException ignore) {}
        Group group = null;
        try {
            // Get the group to add the user
            group = modelServices.lookupGroup(groupName);          [4]
        } catch (PanamaRuntimeException ex) {
            // A PanamaRuntimeException is thrown if the group is not found
            group = null;
        }
        User user;
        // modelFactory.createUser() will automatically create a
        // home folder for the new user.
        if (group != null) {
            Group[] groups = new Group[1];
            groups[0] = group;
            user = modelFactory.createUser(userName, groups);      [5]
        } else {
            user = modelFactory.createUser(userName);
        }
        user.setPassword(userPassword);
        user.setEnabled(true);                                     [6]

        // save the newly created object
        modelFactory.save();                                       [7]
    }

    /**
     * Search for users.
     */
    private User[] searchUser(String userNamePattern)              [8]
                            throws PanamaException, PanamaRuntimeException {
        User[] users;
        ResultSetEnumeration result = null;                        [9]
        try {
            result = modelServices.findUsers(userNamePattern);     [10]
            Vector buffer = new Vector();
            while (result.hasMoreElements()) {                     [11]
                User user = (User) result.next();                  [12]
                buffer.addElement(user);
            }
            users = new User[buffer.size()];
            buffer.copyInto(users);
        } catch (PanamaRuntimeException ex) {
            throw ex;
```

```
            } finally {
                if (result != null) {
                    result.close();                                    [13]
                    result = null;
                }
            }
            return users;
        }

        /**
         * Delete a user.
         */
        private void deleteUser(String userName)                       [14]
                                throws PanamaException, PanamaRuntimeException {
            try {
                if (userName != null && userName.length() > 0) {
                    User user = modelServices.lookupUser(userName);    [15]
                    user.delete();                                     [16]

                    // Save the changes
                    modelFactory.save();
                }
            } catch (PanamaRuntimeException ex) {
                throw ex;
            }
        }

    }
```

10

メッセージ・アプリケーションの作成

この章では、メッセージ・アプリケーションのアーキテクチャと、これらのアプリケーションを使用してモバイル・アプリケーションを作成してデプロイする方法について説明します。項ごとに様々なトピックを記載しています。各項の内容は、次のとおりです。

- メッセージの概要とアーキテクチャ
- メッセージの送受信
- 非同期アプリケーションの作成
- XMS メッセージ・センター
- デバイス・チャネルの選択
- トランスポート・コンポーネント
- プレミアム SMS と逆課金 SMS のサポート

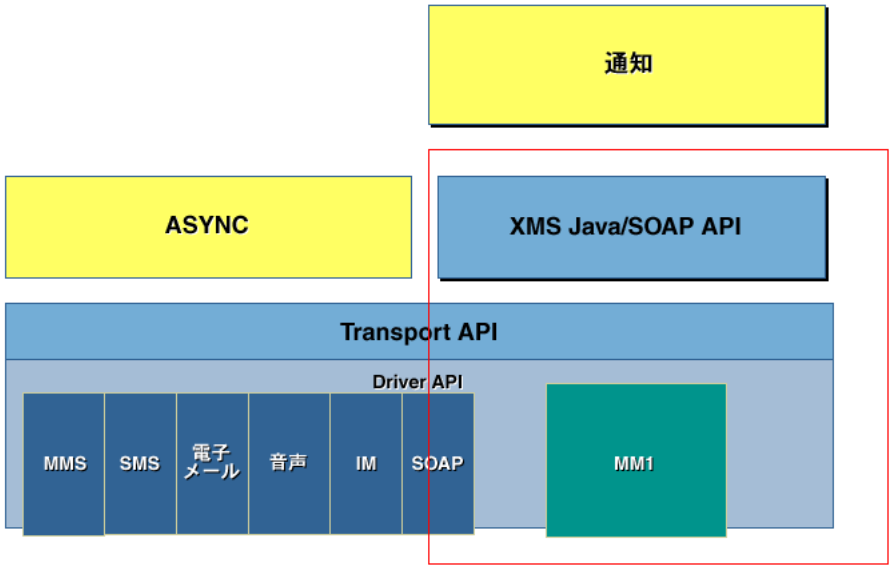
メッセージの概要とアーキテクチャ

概要

メッセージ・サービスは、モバイル・ユーザー間でのメッセージの送受信をサポートすることでモバイル・アプリケーションの機能を拡張する重要なコンポーネントです。OracleAS Wireless のメッセージ・サービスは、メッセージをすべてのモバイル・デバイスに配信するための高度にスケーラブルなメカニズムを提供します。メッセージは、デバイス固有のプロトコルを使用してモバイル・デバイスに配信されます。たとえば、携帯電話には SMS を介して、双方向ポケットベルには電子メールで、通常の電話にはオーディオ・メッセージで、IM クライアントには Instant Messaging（IM）メッセージで、FAX マシンには FAX でメッセージが送信されます。

OracleAS Wireless のメッセージ・サービスは、HTTP を介して SOAP を使用する Web サービス・インタフェース（WSDL を使用して指定）も提供します。SOAP サービスによって、アプリケーションは HTTP プロトコルを介してリモート・オブジェクトのメソッドを起動できます。そのため、アプリケーションでは、インターネット上のどこからでも任意のプログラミング・モデルを使用してメッセージ・サービスを起動できます。OracleAS Wireless のメッセージ・サービスによって、アプリケーションではメッセージとその受信者の両方を指定できます。アプリケーションは、SOAP と HTTP を使用して OracleAS Wireless 内のメッセージ・サービスと通信します。OracleAS Wireless はメッセージを受信し、SMS、電子メール、音声などの適切なプロトコルを使用して、モバイル・デバイスにメッセージを配信します。

図 10-1 OracleAS Wireless のメッセージ・アーキテクチャ



OracleAS Wireless のメッセージ・サービスはスケーラブルで、多数のデバイスに対して大量のメッセージを処理できます。メッセージ・サービスは拡張可能なアーキテクチャと設計に基づいており、多様なデバイスとプッシュ・プロトコルをサポートできます。プッシュ・プロトコルは、XMS メッセージ API で処理されます。メッセージ・サブシステムはドライバ・ベースのアーキテクチャをサポートしています。ドライバは、すべてのデバイス固有または通信プロトコル固有のルーチンを処理する Wireless メッセージ・システムのコンポーネントです。

メッセージの主な機能

OracleAS Wireless は、次の機能を備えたインテリジェント・メッセージを提供します。

- デバイスの自動選択
- メッセージ・コンテンツの調整
- フェイルオーバー配信制御
- 豊富なメッセージ機能を使用できる MMS（マルチメディア・メッセージ）機能
- アクション可能通知メッセージ
- 連絡ルールの統合

マルチチャネル対応メッセージ

一般的に XHTML または OracleAS Wireless XML で作成されたメッセージは、OracleAS Wireless がデバイスにあわせて自動的に調整します。調整機能によって、メッセージは一度で記述され、受信デバイスの機能に応じて自動的に最適化されます。たとえば、イメージをフルカラーからモノクロに変更するイメージ変換を使用して、マルチメディア・コンテンツをデバイスに応じて最適化できます。

マルチメディア・メッセージ

OracleAS Wireless では、グラフィックス、ビデオ、オーディオなどの豊富なモバイル・メッセージ用のマルチメディア・メッセージ（MMS）がサポートされます。MMS メッセージは、SMIL、XHTML または OracleAS Wireless XML で固有に作成できます。

トランスポート・フレームワーク

メッセージ・サブシステムは、デバイス・アドレスと SMS、MMS、IM、音声、電子メールなどのトランスポート・タイプに基づいており、メッセージを適切なトランスポート・プロトコル・ドライバにディスパッチします。ドライバ・インタフェースは、メッセージをデバイス固有のプロトコルでデバイスに配信します。メッセージ・サブシステムは、1 つのインスタンスで複数のドライバをサポートできます。

OracleAS Wireless のメッセージ・ドライバは、デバイス固有または通信プロトコル固有の処理ルーチンを実装する交換可能なモジュールです。OracleAS Wireless には、SMS（SMPP と UCP）、MMS、IM、音声、電子メール、FAX などの通信プロトコルをサポートするビルトイン・ドライバが組み込まれています。

OracleAS Wireless には特殊なドライバ実装が用意されており、他の OracleAS Wireless インストールや、OracleAS Wireless が定義した Web サービス・インタフェースを使用するサービスのクライアントとして、Wireless インスタンスを動作できます。この特殊なドライバは、SOAP インタフェースを（XMS API として）使用してメッセージを送信します。デフォルトでは、このドライバはインターネット上で Oracle がホスティングしている OracleAS Wireless インスタンスの XMS クライアントとして機能するように構成されています。インスタンス管理者は、このデフォルト設定を、OracleAS Wireless が定義した XMS WSDL インタフェースを使用するサーバーを指すように変更できます。そのため、OracleAS Wireless インストールは、デフォルトで SMS、MMS、電子メール、音声、FAX メッセージを送信する機能があります。

割当て制限があらかじめ設定されています。事前に割り当てられている制限を超えて割当て制限を引き上げる必要がある場合は、Oracle の管理者にお問合せください。

MMS センター

XMS コンポーネントには、デフォルトで MMS センター（MMSC）機能が組み込まれており、MM1 メッセージ通知プロトコルをサポートします。受信者のデバイスに MMS ブラウザがある場合、通知メッセージは MMS ブラウザに送信され、HTTP を使用してメッセージを取得します。コンテンツは OracleAS Wireless によって格納および配信されます。唯一必要な外部コンポーネントは、通知メッセージを送信するための標準 SMSC です。OracleAS Wireless では、ホスティングされているサーバーを使用してデフォルトで SMS を送信できることに加え、デフォルトで MMS も送信できます。

アクション可能メッセージのフレームワーク

通知フレームワークは、柔軟なメッセージ・テンプレート、メッセージ偽装を防止するセキュリティ、メッセージ優先順位のサポート、および大量の通知を処理する柔軟性を備えています。

メッセージの送受信

一方向のメッセージ・アプリケーション API の概要

OracleAS Wireless のメッセージ・サービスは、トランスポート・レイヤーとして HTTP を介した SOAP を使用して、Web サービスとしてデプロイされます。WSDL（Web Services Definition Language）は、Web サービス・アプリケーションを定義する標準的な XML インタフェースです。明確に定義した WSDL を使用することによって、開発者は Java や VB などのプログラミング言語を使用してアプリケーションを作成し、インターネットを介して OracleAS Wireless のメッセージのインタフェースと通信できます。また、任意の WSDL ツールキットを使用して一方向（プッシュ）アプリケーションを迅速に実装し、インターネット上で任意の OracleAS Wireless インスタンスを使用してモバイル・デバイスにメッセージを送信できます。

OracleAS Wireless は、Web サービス・インタフェースに加えて、一方向のメッセージ（プッシュ）アプリケーションを作成するための単純な Java API（XMS API）もサポートしています。XMS API は、HTTP を介した SOAP を使用して OracleAS Wireless サーバー・インスタンスと通信します。ただし、XMS API は、アプリケーションの Java コードから任意のプロトコル固有の（SOAP）実装の詳細を抽象化します。XMS は、明確で単純なメッセージ配信用インタフェースを必要とするアプリケーション開発者にとっての優先 API です。

XMS API では、SMS、MMS、IM、音声、電子メールおよび FAX など、あらゆる種類のデバイスにメッセージを配信できるように汎用インタフェースがサポートされています。この API により、アプリケーションでは配信リクエストを 1 つ使用するのみで、単一メッセージに複数の受信者を指定できます。

また、メッセージの宛先アドレスには、異なる通信チャネルを使用するデバイスを指定できます。たとえば、単一のメッセージ配信リクエストで、アプリケーションから電子メール・マシンと FAX マシンにメッセージを送信できます。アプリケーションでは、1 つの配信リクエストで、SMS または MMS デバイス、電子メールまたは IM クライアントおよび音声デバイスのユーザー・リストにメッセージを送信できます。

OracleAS Wireless では、配信用に各種のコンテンツがサポートされています。テキスト文字のみで構成されるメッセージでも、複数の部分を含む複雑なメッセージでもかまいません。メッセージ・タイプは MIME に基づいて識別されるため、ターゲット・デバイスにメッセージの MIME タイプがサポートされている場合は、Microsoft Word や Adobe PDF などの文書を配信できます。OracleAS Wireless には、2 種類の異なる XMS API が用意されています。`XMSSimpleSender` API は、テキストのみのメッセージをサポートします。一方、`XMSSender` 拡張 XMS API は、任意の MIME タイプのメッセージをサポートします。

次に、XMS API の機能の概要を簡単に示します。詳細は、XMS の JavaDoc を参照してください。

XMSSimpleSender

`oracle.panama.messaging.xms.XMSSimpleSender`

XMSSimpleSender は、OracleAS Wireless の以前のリリースでの PushLite API に相当します。文字列パラメータのみを処理する単純な API です。この API は軽量で、使用方法も簡単です。

sendMsg

```
public String[] sendMsg(String[] recipients, String message)
```

テキスト・メッセージを（件名なしで）複数のトランスポート・タイプの複数の受信者に送信します。メッセージのエンコードとコンテンツは、メッセージのコンテンツと受信デバイスによって異なります。テキスト・メッセージを送信するには、このメソッドを使用するのが最も簡単です。オーバーロードされる他の `send()` メソッドを使用すると、件名、返信先、コンテンツ・タイプのエンコーディング（MIME タイプ）および関連する主要パラメータを設定できます。

recipients: 受信者のアドレス（電子メール・アドレス、電話番号またはユーザー名など）。

アドレスの書式は、次のとおりです。

```
<address string ><:transport type > [;<address string>:<transport type >]*
<address string> = <email address>|<phone number> |< subscriber ID > |< user address
>
<user address> = < brand name > ~ < user name >
<phone number> = < country code > - < area code > - < local phone number >
```

1 行に 1 人の受信者を指定します。受信者は複数のフェイルオーバー・アドレスを保持している可能性があります。また、各アドレスには少なくとも 1 つのトランスポート・タイプが必要です。コロン（:）を使用して、アドレスとトランスポート・タイプを区切ります。カンマ（,）を使用して、同じアドレスの各トランスポート・タイプを区切ります。セミコロン（;）を使用して各アドレスを区切ります。

例 1: `SMS:1-650-5551234,Voice:1-650-5551234;Fax:1-408-3456789`

例 2: `Email:myemail@foo.com;Voice:mary,Email:mary,Fax:mary;Email:bob`

[transport] は、`oracle.panama.messaging.common.TransportType` に定義されているタイプの 1 つです。

message: メッセージの本文。メッセージ・テキストには、プレーン・テキストまたはリッチ・テキスト（XHTML または OracleAS Wireless XML）を指定できます。XMS は、メッセージのコンテンツを調べてリッチ・テキストが使用されているかどうかを判断します。リッチ・テキストは、ターゲット・デバイスに適したマークアップに変換されます。

getStatus

```
public String[] getStatus(String[] messageIDs)
```

一連のメッセージ ID に対する現行のステータスを取得します。

戻り値 : テキスト・ステータス文字列の配列。

XMSSender

`oracle.panama.messaging.xms.XMSSender`

XMSSender は、OracleAS Wireless の以前のリリースでの Push API に相当します。複雑な（複数の部分に分かれた）メッセージを送信できる API です。

sendMsg

```
public WorkOrder[] sendMsg(Packet pkt)
```

メッセージ・パケットを送信します。

pkt: 配信するメッセージ・パケット。Packet クラスについては後の項で説明します。

戻り値 : XMS サーバーがリクエストを受け入れると、一連の WorkOrder が戻されます。WorkOrder は、受信者アドレスのインスタンスごとに 1 つ戻ります。

getStatus

```
public Status getStatus(WorkOrder workOrder)
```

WorkOrder の現行のステータスを取得します。1 つの WorkOrder には、アドレス 1 つと、そのアドレスのメッセージ ID が含まれます。

```
public Status[] getStatus(WorkOrder[] workOrders)
```

一連の WorkOrder に対する現行のステータスを取得します。

```
oracle.panama.messaging.push.Packet
```

Packet クラスは、現実の一般的なメッセージ（電子メールなど）を表します。件名、1 つの本文または一連のメッセージ本文（複数の部分）を含めることができます。同じメッセージを、複数のトランスポート・タイプ（デリバリ・タイプ）の複数の受信者に配信できます。たとえば、同じメッセージを、同じパケットで 2 人の電子メール受信者、3 人の SMS 受信者および 4 台の FAX マシンに配信できます。

各トランスポート・タイプには、送信者、代替返信先アドレスおよび受信者グループを指定できます。パケットには、優先順位、登録済など、一連の配信指示を任意に含めることができます。

そのためには、最初に空の Packet インスタンスを構成します。次に、パケットのメッセージ、メッセージ情報、送信者、返信先および受信者を設定します。詳細は、後述のサンプル・コードを参照してください。

XMS API には、メッセージのプロパティを設定して OracleAS Wireless インスタンスにディスパッチするメソッドが用意されています。API インタフェースの詳細は、OracleAS Wireless XMS の Javadoc（`oracle.panama.messaging.xms`）を参照してください。プッシュ・メッセージを送信するには、次の情報を指定する必要があります。

- プッシュ Web サービスが稼働している OracleAS Wireless サーバー。（アプリケーションを OracleAS Wireless VM で実行している場合を除いて）リモート OracleAS Wireless Web サービスへのアクセスに必要なユーザー名、パスワードおよび HTTP プロキシを指定します。
- 実際に送信するメッセージとそのコンテンツ（MIME）タイプ。

テキスト・ベースのメッセージ

XMS API を使用する最も簡単な方法は、テキストのみのメッセージを送信することです。この方法の場合、XMS API は OracleAS Wireless の以前のリリースでの Push API と同じように機能しますが、重要な違いが 1 つあります。WAP プッシュなど、プレーン・テキストを表示できないデバイスにメッセージを送信すると、選択されたデバイスに適切なマークアップで入力テキストが埋め込まれます。

マルチメディア・メッセージ

XMS にはリッチ・テキスト（つまり、OracleAS Wireless XML または OracleAS Wireless がサポートする XHTML マークアップ言語でマークアップされたテキスト）を使用できます。入力メッセージのコンテンツは、選択されたターゲット・デバイスに適合するように変換されます。リッチ・テキストには、埋込みイメージまたはサウンド・ファイルに加え、文書自体の構造とレイアウトが含まれます。また、ターゲット・デバイスに応じて URL が適切に処理されます。特に、ターゲット・デバイスで HTTP ハイパーリンクを表示できない場合は、かわりに逆非同期（RevAsync）形式を使用してメッセージが再度書き込まれます。これによって、エンド・ユーザーはリクエスト / 返信メッセージ交換を使用してリンクを起動できるようになります。

XMS は、入力マークアップのトランスコーディングに加え、イメージやオーディオも適切に変換します。イメージの場合、変換はイメージ形式の変更のみでなく、必要に応じてターゲット・デバイスのイメージに対するサイズ変更やリサンプリングも実行されます。ハンドセットのプロビジョニングをサポートするチャネル（SMS など）を介して単一のメッセージを送信する場合、メッセージはイントール可能なオペレータ・ロゴとして送信されます。同様に、単一のオーディオ・クリップを送信する場合は、ターゲット・デバイスに適した形式で着信音に変換されます。

逆非同期の透過的なサポートの提供によって、XMS では、ユーザーが対話可能なリッチ・コンテンツ（アクション可能通知と呼ばれます）を他のアプリケーション（通知フレームワークなど）から送信できます。これらの通知には OracleAS Wireless サービスの出力が含まれています。ユーザーはメッセージに返信することによって、サービスと対話できます。アクション可能通知は、ユーザーではなく、サーバーによって開始される RevAsync セッションとみなすことができます。

XMS では、マークアップ言語要素（メッセージのターゲット・デバイスに依存）のマッピングに加え、標準的な OracleAS Wireless XML マークアップを使用できます。MMS ブラウザなどのリッチ・ターゲット・デバイスは、マルチメディア要素すべてを受け取り、マークアップのタイミング情報を使用してマルチメディア・コンテンツの存続期間も制御します。イメージは JPEG、GIF または BMP 形式で提供できます。イメージ形式がターゲットのデバイスでサポートされていない場合は、ターゲット・デバイスに適切な別の形式に変換されます。また、XMS では OracleAS Wireless XML の `SimpleImage` タグの available 属性もサポートしています。これは、イメージを複数の形式で格納でき、使用可能なイメージ形式のうち、できるかぎり最適な形式が選択されることを意味します。自動変換では計算上のオーバーヘッドが生じ、最終イメージへの変換が意図しない結果となる可能性もあるため、現実的にはここで説明したアプローチをお薦めします。指定のイメージに対して複数のバージョ

ンを事前に生成することによって、色の深度を制限した低解像度のイメージをサポートするデバイス用に単純なイメージのバージョンを作成できます。

その他のコンテンツ

プレーン・テキストとリッチ・テキストに加え、XMS は任意のコンテンツを送信できます。その際、トランスコーディングは実行されません。また、クライアント・プログラマの責任で、メッセージを適切にフォーマットおよびパッケージ化する必要があります。

この場合、明示的な MIME タイプを指定する必要があります。つまり、このクラスのメッセージに使用する適切な API は `XMSSender` API です。

双方向メッセージ Transport API

Transport API は、送受信の両方に使用できる豊富な API セットです。

Transport API は、クライアント側のメッセージのインタフェースです。この項では、トランスポート・システムのカスタマイズに使用できる主な構成メンバーと機能など、Transport API の詳細を説明します。

XMS API は Transport API を介して作成されます。XMS API は送信のみを処理します。Transport API は、送信に関するメッセージ変換を提供しません。ただし、Transport API は、ステータスのトラッキング、ヒント、微調整されたメッセージのルーティングなど、XMS API にはないいくつかの特別な機能を提供します。

メッセージ配信リクエストが送信されると、トランスポート・システムは受信者の分析を実行し、メッセージを配信用の適切なプロトコル・ドライバにルーティングします。

メッセージを受信するには、アプリケーションでリスニング・エンドポイントとメッセージ・コールバック・リスナーをトランスポート・システムに登録する必要があります。エンドポイントは、電話番号などのアドレスの形式を取ります。エンドポイントは、トランスポート・システムに対してメッセージのディスパッチ方法を識別します。ターゲット・アドレスへのメッセージを受信すると、一致するアドレスを持つエンドポイントに関連付けられているリスナーにディスパッチされます。

主要なインタフェースは、`oracle.panama.messaging.transport.Messenger` です。このインタフェースのインスタンスは、`TransportLocator` クラスを介して取得できる `oracle.panama.messaging.transport.MessengerController` の `Get` メソッドを介して戻されます。これによって、パッケージの残りの部分にアクセスしてメッセージ・アプリケーションを作成できます。API の詳細は、Javadoc を参照してください。

宛先分析

1 つのメッセージを通信プロトコルの異なる複数の受信者に配信できます。たとえば、会議のアラームを数人には SMS を使用して送信し、他の人々については電子メール・アドレスに送信できます。トランスポート・システムでは、メッセージをドライバにルーティングする前に、受信者を分析してデリバリのカテゴリ別にグループ化します。通常、トランスポート・システムはすべての宛先を分析し、それに従ってグループ化することで内部処理を開始します。

メッセージのルーティング

メッセージを送信するために、トランスポート・システムでは送信用の適切なドライバを特定する必要があります。適切なドライバを検索する処理は、メッセージのルーティングと呼ばれます。トランスポート・システムには、特定の時点で多数のメッセージング・サーバーとプロトコル・ドライバが構成されている場合があります。様々なドライバ・インスタンスで異なるカテゴリのメッセージを処理できます。

たとえば、ドライバで送信できるのが SMS メッセージのみの場合があります。また、電子メールと FAX メッセージしか送信できない場合もあります。したがって、トランスポート・システムは、SMS メッセージの送信には SMS 機能を持つドライバ、電子メール・メッセージの送信には電子メール機能を持つドライバを使用する必要があります。同じカテゴリのメッセージを処理できるドライバが複数存在することもあります。たとえば、複数の SMS ドライバがあり、一方は ATT の SMSC、他方は Cingular の SMSC で通信するとします。トランスポート・システムでこの 2 種類の SMS ドライバに SMS メッセージを使用する場合、ATT のデバイスには ATT の SMS ドライバを使用し、Cingular のデバイスには Cingular の SMS ドライバを使用する必要があります。

これらの決定はいずれも、2 組の情報に基づいてトランスポート・システムにより行われます。1 組目は、デリバリ・タイプ、スピード、コスト、エンコーディングなど、アプリケーションが指定する送信基準です。この中で、デリバリ・タイプは必須であり、クラスの接続先で指定できます。2 組目の情報は、使用可能なドライバのセットから提供されます。ドライバのプロパティは、ドライバのスピード、ドライバ・コスト、エンコーディングおよびデリバリのカテゴリなどで、管理者により構成されます。

前述のように、ルーティングによって最適なドライバが検索されます。デリバリのカテゴリのように一致する必要があるプロパティと、コストやスピードのように単に最も近似の一致が検索されるプロパティがあります。

トランスポート・システムでは、次の情報がルーティングに使用されます。

- デリバリのカテゴリ
- プロトコル
- 電話会社
- スピード
- コスト

ルーティングでは、属性のエンコーディングは使用されません。

トランスポート・システムでは、メッセージは次の情報が最も一致したドライバにルーティングされます。

1. SMS や EMAIL など、デリバリのカテゴリ。
2. UCP や SMPP など、プロトコル。
3. Cingular や Telia など、電話会社。
4. `(speed_requested >= 0` かつ `cost_requested >= 0)` の場合、最小 `(driver_speed -speed_requested)**2 + (driver_cost - cost_requested)**2`。

   または

   `cost_requested < 0` の場合、最小 `abs(driver_speed - speed_requested)`。

   または

   `speed_requested < 0` の場合、最小 `abs(driver_cost - cost_requested)`。

複数のドライバが前述の基準を満たしている場合、トランスポート・システムではその 1 つがランダムに選択されます。

トランスポートの内部処理を容易にするためのヒントの提供

アプリケーションでは、ルーティングと宛先分析の所要時間を短縮するヒントを提供できます。たとえば、すべての受信者に対してデリバリのカテゴリに「電子メール」を指定すると、トランスポート・システムでは各受信者を調べてカテゴリを判別する必要がなくなります。

原則として、メッセージ配信（`Messenger.send()` メソッド）に必須のパラメータは、*Destination* と *Message* です。他のパラメータ（*SenderInfo*、*MessageInfo* および *DeviceInfo*）はすべてオプションです。オプションのパラメータを指定すると、すべての受信者に共通するプロパティを記述するヒントとして解析されます。たとえば、*DeviceInfo* を指定した場合に、この DeviceInfo インスタンスの `getDeliveryType()` が `DeliveryType.EMAIL.getName()` を戻すと、トランスポート・システムでは、この戻り値はすべての受信者が電子メール・アドレスであることを示すヒントとして使用され、宛先分析は実行されません。

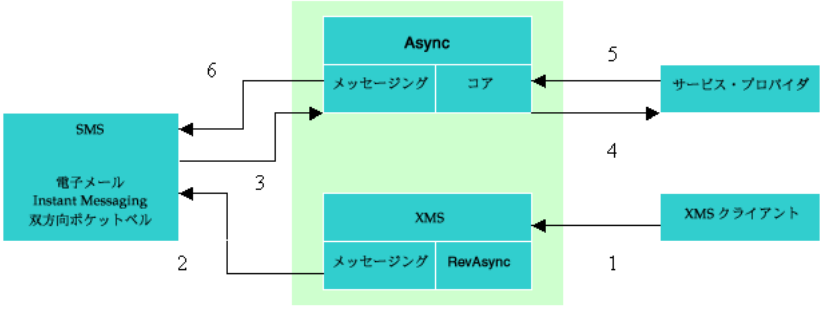
アクション可能メッセージ

以前の OracleAS Wireless では、メッセージ・デバイスはサーバーから開始されたプッシュ・メッセージにレスポンスできませんでした。たとえば、OracleAS Wireless から送信された通知メッセージは、ユーザーのデバイスが受信した時点で最終結果とみなされ、それ以上の相互作用はありませんでした。メッセージにレスポンスする機能がなかったために、プッシュ・メッセージを介してデバイスのユーザーに提供される情報およびオプションは制限されていました。

コンポーネントの概要

XMS と Async の統合によって API が公開され、コンテンツ・プロバイダが、SMS、電子メール、Instant Messaging などのメッセージ・チャネルに作用するプッシュ・メッセージを作成できるようになりました。図 10-2「アクション可能メッセージ」に、関係のあるコンポーネントとその関連を示します。

図 10-2 アクション可能メッセージ



XMS クライアント

XMS クライアントは、メッセージとコンテンツを配信するために XMS API をコールするコンポーネントです。プッシュ・メッセージをアクション可能にするために生成できる文書の種類には、いくつかの要件があります。

- 文書は、OracleAS Wireless でサポートしているコンテンツ・マークアップ形式であることが必要です。このリリースでは、アクション可能メッセージには OracleAS Wireless XML のみがサポートされます。
- 文書に使用する URL リンクは、HTTP または OMP プロトコルであることが必要です。OMP は、OracleAS Wireless インスタンスのサービスを識別する仮想 URL です。OMP URL を使用すると、OracleAS Wireless で作成されたサービスを一意に識別できるという利点があります。したがって、サービス OID の変更がユーザー・プログラムに影響することはありません。

XMS

アクション可能なメッセージに対する XMS の役割は、送信される文書が永続的な状態であるかどうかを判断することです。たとえば、URL が埋め込まれている文書を SMS デバイスにプッシュします。デバイスには URL リンクを処理するためのブラウザがないため、状態をサーバーにキャッシュする必要があります。XML は、基礎となる RevAsync コンポーネントをコールすることによって、これを実行します。その後、文書はデバイス結果に変換され、送信されます。

RevAsync

RevAsync は、XMS コンポーネントの下に新しく作成されたインスタンスです。RevAsync の主な機能は、OracleAS Wireless でサポートしているマークアップ文書を入力として受信し、キャッシュが必要なすべての要素と属性を特定するために文書を分析することです。キャッシュされたオブジェクトは、データベースに永続的に格納され、デバイス・ユーザーが返信した後すぐに、Async がセッションの状態を再構成するために使用されます。

アクション可能メッセージの流れ

アクション可能メッセージを有効にするには、XMS に提供するコンテンツが OracleAS Wireless でサポートするマークアップ文書であることが必要です。このリリースでは、OracleAS Wireless XML のみがサポートされます。

アクション可能メッセージをトリガーするには、次の 2 つの方法があります。

- ユーザー指定の条件を満たすたびにサービスが起動されるように、通知サービスを作成します。サービスからは、いくつかのハイパーリンクが文書内に埋め込まれている OracleAS Wireless XML 結果が戻ります。
- XMS API を介して OracleAS Wireless XML 文書を直接プッシュします。文書には、ユーザー相互作用を適用できるように、いくつかのメニューまたはフォームの構成メンバーが含まれている必要があります。ユーザーの返信で起動されるサービスにアドレスを指定する場合は、文書の URL 値に OMP プロトコルを使用すると、開発者がサービスの OID 変更について考慮する必要はありません。

OracleAS Wireless XML の入力を受け取ると、XMS は、文書内でユーザー相互作用をさらにトリガーできる要素を検索します。典型的な例としては、デバイス・ユーザーが選択するハイパーリンクを表す `SimpleMenu` 要素があります。このような状況を検出すると、永続的な状態および対応するトランザクション ID が生成されます。RevAsync は、ユーザーの返信に使用する書式とトランザクション ID を示す結果メッセージに特別な指示の行を追加します。レスポンスすることをユーザーが決定した場合は、返信パラメータの一部としてトランザクション ID が必要です。この方法で、アクション可能メッセージの返信を受け取ると、Async はトランザクション ID を使用してセッション状態をリストアできます。

永続的なユーザーの状態を取得するために、Async は、次の方法を使用してアクション可能メッセージのユーザー返信と典型的な非同期リクエストを区別します。

1. ユーザーは、サイト全体で一意のアクション可能メッセージ（AM）短縮名で返信します。AM 短縮名は、サイト全体で一意のシステム・パラメータです。Async は、コマンドラインが AM 短縮名で開始しているかどうかによって、リクエストがアクション可能メッセージへの返信であることを区別します。ユーザー返信はこのオプションを使用し、次の書式に従う必要があります。

   ```
   <AM short name> <transaction ID> <parameters>
   ```

   たとえば、ユーザーは、次の株価通知メッセージを受信できます。

   ```
   To respond, type 'am 2 <link selector>'
   [orcl] L 15.25 B 15.20 A 15.30 O 15.10
   1 News
   2 Detail Quote
   ```

   ユーザーは、コンテンツ「`am 2 2`」でこのメッセージに返信し、「Detail Quote」オプションを取得します。「`am`」はサイト全体で一意の短縮名であるため、Async は、この短縮名の直後のパラメータをトランザクション ID と解釈し、これを使用して永続的な状態すべてを取得します。

2. プッシュ・メッセージの送信者アドレスをアクション可能メッセージ専用のアクセス・ポイントとして設定します。非同期アクセス・ポイントを AM 専用に構成できるため、アクセス・ポイントへのすべてのリクエストがアクション可能メッセージへの返信として解釈されるようになります。したがって、ユーザーによるアクション可能メッセージ短縮名の指定は不要になります（前述の例では、「2 2」を返信して「Detail Quote」を取得します）。

アクション可能メッセージの有効化

アクション可能メッセージを有効化するには、次の手順に従います。

1. 送信する OracleAS Wireless 文書を準備します。

   前述のように、文書は、通知サービスの出力として、または XMS クライアントがプッシュする文書として生成できます。文書にはアクション要素（デバイス・ユーザーがアクションを行う SimpleMenu または SimpleForm など）が含まれている必要があります。サンプル文書を次に示します。

```
<SimpleResult>
   <SimpleContainer>
      <SimpleText>
          <SimpleTextItem>Approve/Disapprove John Doe's expense report #1234</SimpleTextItem>
          <SimpleTextItem>Upgrade 256 MB memory on desktop pc - $30<SimpleBreak/></SimpleTextItem>
      </SimpleText>
      <SimpleMenu name="" title="">
         <SimpleMenuItem
target="omp://expensereply?answer=approved&amp;reportNo=1234">Approve</SimpleMenuItem>
         <SimpleMenuItem
target="omp://expensereply?answer=disapproved&amp;reportNo=1234">Disapprove</SimpleMenuItem>
      </SimpleMenu>
   </SimpleContainer>
</SimpleResult>
```

   omp://expensereply は、ユーザーの返信メッセージでこれらのアクション項目がトリガーされたときに起動するサービスを識別します。

2. プッシュされる文書にアドレス指定するアプリケーション・リンクを作成します。

   プッシュされる文書にアドレス指定するアプリケーション・リンクは、OMP の値セットで作成する必要があります。前述の例では、omp://expensereply が OMP の値です。

3. アクセス・ポイントを作成します。

   サポートされる各配信チャネルに対して、アクセス・ポイントを少なくとも 1 つ作成する必要があります。アクセス・ポイントをアクション可能メッセージに対するユーザー返信のエントリ・ポイントとしてのみ確保する場合は、アクセス・ポイントをアクション可能メッセージ専用として設定できます。あるいは、通常のアクセス・ポイントとして設定することもできます。この場合は、アクション可能メッセージの非同期リクエストと返信の両方のメッセージに使用できます。ユーザー返信時に非同期エントリ・ポイントにルーティングできるように、プッシュ・メッセージの送信者アドレスを構成したアクセス・ポイントに設定できます。同じチャネルに対して複数のアクセス・ポイントのセットがある場合は、専用フラグが設定されているアクセス・ポイントが優先されます。

構成パラメータ

次の表は、アクション可能メッセージでサポートしている構成パラメータです。設定手順の詳細は、『OracleAS Wireless 管理者ガイド』を参照してください。

表 10-1 アクション可能メッセージの構成パラメータ

パラメータ名	説明
アクション可能メッセージへの返信用のエイリアス名	これは、アクション可能メッセージの非同期リクエストと返信の両方が同一のアクセス・ポイントをエントリ・ポイントとして共有する場合に、アクション可能メッセージの返信を識別するための名前で、サイト全体で一意の短縮名です。
デバイスに対するアクティブ・トランザクションの最大数	このパラメータは、デバイス当たりの永続的なアクティブ・トランザクションの最大数を定義します。ユーザーがアクション可能メッセージに返信すると、メッセージによってアドレス指定された永続トランザクションは削除されます。
アクティブでないトランザクションの有効期限	このパラメータは、アクティブでないトランザクションの有効期限（日数）を指定します。期限が切れると、トランザクションは永続格納領域から削除されます。
アクション可能メッセージへの返信専用アクセス・ポイント	アクセス・ポイントがアクション可能メッセージへの返信専用かどうかを示すフラグです。このフラグが設定されている場合、アクション可能なプッシュ・メッセージすべてのアクセス・ポイントには、送信者アドレスが設定されます。プッシュ・メッセージに追加された、アクション可能メッセージに対する返信方法の指示では、短縮名を省略します。ユーザーに必要な操作は、トランザクション ID とアプリケーション・パラメータを使用して返信することのみです。

非同期アプリケーションの作成

非同期リスナー

非同期リスナーのアーキテクチャ

OracleAS Wireless は、WAP または XHTML ブラウザを備えたデバイスなどのブラウザ・ベースのデバイスを介して、または SMS を備えた携帯電話などのメッセージ・プロトコル・ベースのデバイスを介してアクセスされる Wireless アプリケーションと音声アプリケーションを開発するためのフレームワークと実行環境を提供します。Async は、メッセージ・プロトコル・ベースのデバイスが、これらの Wireless アプリケーションにアクセスできるようにする Wireless コンポーネントです。

規則上、アプリケーション・サーバーへのエントリ・ポイントは HTTP プロトコルを通ります。このため、アプリケーション・サーバー上で作成されるアプリケーションは、Web 機能を持つクライアントのみに制限されます。大多数のモバイル・ユーザーは Web アクセス権を持っていないか、Web アクセスを使用できないため、このサーバー制限はモバイル・ユーザーにとって問題です。この種のユーザーがある種のメッセージ機能（電子メール、SMS など）を使用する必要があることはほぼ確実です。そのため、開発者は、この種のユーザーの機能に応じて専用アプリケーションを作成するか、単にアプリケーション・サーバーがモバイル市場に対処できないため無視するかというジレンマに直面します。

OracleAS Wireless によって、開発者が何もしなくても、このジレンマが解消されます。Async の導入によって、通常の HTTP プロトコルを介してモバイル・アプリケーションにアクセスできるのみでなく、他のメッセージ・プロトコル（電子メールや SMS など）でもモバイル・アプリケーションにアクセスできます。開発者は、特定のプロトコルに適合させることに気遣いながらアプリケーションを作成するのではなく、アプリケーション・ロジックの構築に専念できます。適切な接続、セッション管理およびユーザー・リクエストの解析は、OracleAS Wireless によって行われます。モバイル・アプリケーションは、受信リクエストがどのプロトコルで処理されるかに関係なく同じ方法で起動します。したがって、アプリケーション開発者は完全に透過的にサービスへのアクセスを許可できます。

主要な課題

複数のメッセージ・トランスポート・プロトコルのサポート

最も明白な課題の 1 つは、複数のプロトコルをサポートすることです。電子メール、SMS および他のプロトコルを順に処理するように同じ機能を構築するのは望ましくありません。OracleAS Wireless は、クライアントで使用されているプロトコルに関係なく、同じアプリケーションへのアクセスを提供します。このため、直近の課題は、複数のプロトコルを均一にサポートする機能です。

メッセージ・プロトコルの非同期性

一般に同期プロトコルの 1 つである HTTP プロトコルとは異なり、SMS や電子メールなどのメッセージ・プロトコルは非同期です。これは、HTTP とは異なり、リクエストおよびレスポンス・モデルに基づいていないためです。1 つの基本操作は、通常は一方向です。たとえば、Web ブラウザを使用する場合は、URL を入力してリクエストを送信した後、結果が戻されるまで待ちます。メッセージ・プロトコル（SMS など）の場合は、メッセージ自体の送信によって 1 つの操作が完了します。ほとんどのアプリケーションはユーザー・リクエストにレスポンスを返すため、通常は HTTP で十分です。同じアプリケーションに非同期プロトコルを介してアクセスできるようにすることは、この種の動作を SMS や電子メールなどのプロトコルで疑似実行するにはどうすればよいかという課題を表します。

セッションのサポート

もう 1 つの大きな課題は、ほとんどのアプリケーションがセッション・ベースであることです。通常は、タスクを完了するために複数のリクエストとレスポンスが必要になります。クライアントである Web ブラウザには、セッションのセマンティクスを容易にするために Cookie などの機能が組み込まれているため、アプリケーションには Web の分野でセッションを保持する機能があります。この機能は、電子メールまたは SMS のクライアントにはありません。対話型アプリケーションをサポートするこの種の機能は組み込まれていません。

ユーザー・ナビゲーション

Web ブラウザには、アプリケーションをナビゲートできるようにユーザー・インタフェースが用意されています（たとえば、ハイパーリンクをクリックし、メニューや一連の手順を横断して特定の機能を完了する操作などです）。SMS や電子メールなど、他のプロトコルを処理するクライアントには、通常はこの種のナビゲーション機能はありません。ここでの課題は、アプリケーションがプロトコルから独立できるように、この種のクライアントに同様のナビゲーション機能を与えることです。

アプリケーションのネーミングとアドレッシング

Web の分野では、通常、アプリケーションには URL が割り当てられています。URL は、アプリケーションを識別し、リクエストするための手段です。メッセージング・クライアントは通常はプレーン・テキスト・デバイスであり、アプリケーションのネーミング規則はありませんが、プロトコル間での一貫性が必要です。

主要な解決策

Async は HTTP サーバーの機能と Web ブラウザの各部を結合して、その機能を提供します。

複数のトランスポート・プロトコルのサポート

この課題は比較的簡単です。OracleAS Wireless トランスポート・システムの最上位に組み込むことで、トランスポート・システム自体の性質により、複数のトランスポート・プロトコルがサポートされます。Async は、メッセージを送受信するトランスポート・システムにアプリケーションとして登録します。さらにプロトコルごとに 1 つ以上のアドレスを登録すると、その順序で各プロトコルを使用してユーザーとの対話が処理されます。たとえば、電子メール用に async@yourcomany.com、SMS 用に 1234567 を登録できます。これによって、async@yourcompany.com および 1234567 は、Web における http://yourcompany.com と同様に、それぞれのプロトコルの URI となります。

Async 自体は受信プロトコルを考慮せず、使用登録されている手段を使用してメッセージを送受信するように設計されています。メッセージのペイロード（コンテンツ）は、Async によって解析され処理されます。

メッセージ・プロトコルの非同期性

Async は HTTP リスナーに似た論理を構築します。この論理は、非同期プロトコルによる同期のセマンティクスを表します。そのために、デバイスからリクエストされたアプリケーションのクライアントとして機能します。Async はユーザーにかわってアプリケーションに対するリクエストを出し、アプリケーションからのレスポンスを待ってから処理し、レスポンスの書式を設定してユーザーに表示します。ユーザーには、先ほどのリクエストからのレスポンスのように見えます。

セッションのサポート

ユーザーからリクエストを受信すると、Async は、対話型アプリケーションが機能できるように、そのユーザー用のセッションを作成します。セッション情報がブラウザ（または Cookie）で保持される HTTP とは異なり、すべてのセッションのステータスは Async によってバックエンドで保持されます。

ユーザー・ナビゲーション

Async はフォームやメニューなどの要素を変換し、エンド・ユーザー用のナビゲーション・コマンドを表示します。フォームなどの要素がアプリケーションから戻ると、Async はフォームの形式をバックエンドで保持し、必要な他のすべての情報とともにフォームが送信されたときの操作を判別します。このユーザーが（Async 固有のコマンド・セットを使用して）コマンドを入力して送信すると、Async は（Async に格納されている現行のユーザー情報に基づいて）リクエストを送信し、ユーザーのかわりに結果を再処理します。ユーザーがリンクをクリックすると、ハイパーリンクがバックエンドに格納されるとみなすことができます。

アプリケーションのネーミングとアドレッシング

Web でアプリケーションに URL を割り当てるのと同様に、Async を使用するには非同期可能のアプリケーションに短縮名を割り当てる必要があります。たとえば、株価アプリケーションにパス /finance/quote が割り当てられており、http://mycompany.com/finance/stock としてアクセスできるとします。コンテンツ・マネージャを使用して、このアプリケーションに 1 つ以上の短縮名（st など）を割り当てることができます。Async が受信するメッセージが st で始まる場合は、株価アプリケーションに対するリクエストを通知します。ユーザーは、Async がリスニングするように構成されている非同期アクセス・ポイント（電子メールの場合は async@mycompany.com、SMS の場合は 1234567 など）に st orcl（orcl はオラクル社のチッカー記号）を送信し、オラクル社の株価を取得できます。

また、非同期アプリケーションを識別するためにサービス・アクセス・ポイントを作成できます コンテンツ・マネージャを使用すると、電子メールのアクセス・ポイント（stock@mycompany.com）と SMS のアクセス・ポイント（123FINANCE）を株価アプリケーションに関連付けることもできます。これによって、単に orcl を電子メールとして stock@mycompany.com に送信するか、または SMS メッセージとして 123FINANCE に送信するのみで、オラクル社の株価を受信できます。

Async によるリクエストの認可

Async では、リクエストを発行するユーザーがゲストおよび登録済という 2 つのカテゴリで区別されます。Async がユーザー・リクエストを受信すると、リクエスト・メッセージのソース・アドレスを使用して、認証のために OracleAS Wireless ユーザーが逆参照されます。ユーザーのプロファイルにデバイス・アドレスが登録されている場合は、ユーザー・オブジェクトを特定できます。このアドレスは、リクエスト・メッセージのソース・アドレスと同一です。特定されたユーザー・オブジェクトは、リクエストにより作成され新規に認証されたセッションにバインドされます。それ以外の場合は、ゲスト・ユーザー・オブジェクトがセッションにバインドされます。ユーザーに対して許可されたアプリケーションには、デバイスから発行されたリクエストからアクセス可能になります。

ゲスト・ユーザーがアクセスできるのは、ゲスト・グループに属しているアプリケーションのみです。ゲスト以外のアプリケーションにアクセスすると、名前とパスワードの入力を求めるフォームが表示されます。ユーザーが有効な OracleAS Wireless ユーザー名とパスワードを入力すると、前のセッションを認証済セッションにアップグレードし、名前で識別されたユーザー・オブジェクトをバインドできます。また、ゲスト・ユーザーはプロンプトが表示されないように、ログイン・コマンド !L（その後にユーザー名とパスワードが続く）を使用して明示的にログインできます。

ユーザー・インタフェースとナビゲーション・コマンド

前述のように、通常、メッセージング・クライアントではプレーン・テキストのみが表示され、対話型のナビゲーション機能は用意されていません。Async は、この種の機能を使用可能にするために、アプリケーションからのレスポンスを変換して特定の表現に形式を設定します。Async には、Web における Web ブラウザと同様に、表現形式とナビゲーション・コマンドのセットが組み込まれています。このため、ユーザーが Async を使用してアプリケーションを起動すると、レスポンスは Async によって変換された形式で表示されます。それ以降の Async との対話は、Async が予期する形式に準拠する必要があります。

この項では、ユーザーが Async に発行できるコマンドについて説明します。コマンドを発行するには、単に適切な書式でメッセージを送信します。コマンドのテキストは、件名行またはメッセージの本文に挿入できます。

システム・コマンド

- !H:（ヘルプ・コマンド）コマンドの使用方法に関する全般的なヘルプを提供します。
- !E:（エスケープ・コマンド）現行フォームのステータスを消去します。
- !S: コマンド・シーケンスの終了をマークします。メッセージに、一連のコマンドを改行またはコマンド・デリミタで区切って挿入できます。!S はコマンド・シーケンスの終了をマークします。!S マークの後のテキストは解析されません。
- help: アプリケーション・レベルのヘルプです。パラメータを指定しないと、すべての非同期アプリケーション・ヘルプが表示されます。また、アプリケーションの短縮名をパラメータとして指定すると、特定の非同期可能アプリケーションに関するヘルプを取得できます。
- !L <username> <password>: ユーザー名とパスワードを指定してシステムにサイン・オンします。
- !O: セッションを終了します。

アプリケーション起動コマンド

次のコマンドは、アプリケーションの起動、メニューの選択およびパラメータの入力に使用します。アプリケーション起動コマンドとフォーム・コマンドには、予約済のコマンド記号はありません。フォーム・コマンドやメニュー項目選択など、ユーザー・セッションで現行のフォームまたはメニューの状態が保持されている場合にのみ起動できるコマンドがあります。フォームまたはメニューの状態の詳細は後述します。

- [<shortname>|<menuitem>] <parm1><parm2> . . . : アプリケーションを起動します。最初のフィールドには、アプリケーションの短縮名またはメニュー項目番号を指定できます。メニュー項目を指定できるのは、アプリケーション結果からメニュー・メッセージを以前に受信している場合のみです。メニューの状態は、Async のユーザー・セッションで保持されます。メニューに基づいて選択し、さらに操作をトリガーできます。現行のメニュー状態の詳細は後述します。

- <parm1><parm2> . . . : フォームのパラメータを入力します。必須パラメータを入力せずにアプリケーションを起動すると、パラメータ値の入力を求めるフォームが戻されることがあります。これによって、ユーザー・セッション内で現行のフォーム状態が作成され、ユーザーは以降のコマンドでパラメータ・シーケンスを送信するものと期待されます。パラメータ値は、前に戻されたフォームに表示されていたパラメータと同じ順序でコマンド行に指定する必要があります。

構成とカスタマイズ

システム構成パラメータ

次の表は、サイトおよびアプリケーションの構成パラメータです。システム管理者やアプリケーション開発者は、Oracle Enterprise Manager（OEM）コンソールおよび OracleAS Wireless Tools を介してこれらのパラメータを使用し、サイト・レベルおよびアプリケーション・レベルでの Async の動作をカスタマイズできます。詳細は、第 5 章「サービスの開発」の「非同期情報の入力」を参照してください。

表 10-2 サービス構成パラメータ

名前	デフォルト	セマンティクス
非同期コマンドライン構文		アプリケーションおよびその使用方法を説明するためにユーザーに送信するヘルプ・メッセージ。
区切り文字	' '（空白）	各パラメータ値を区切る区切り文字。たとえば、星占いサービスの短縮名を ho、区切り文字を ','（カンマ）と仮定します。ho gemini, aries のコマンドを発行すると、双子座（gemini）と牡羊座（aries）の星座占い結果を取得できます。
サイレント	False	デバイスにレスポンス・メッセージを返信しないアプリケーションに対して、このフラグを選択します。これは、アプリケーションをサイレントとして静的にマークする方法です。アプリケーション結果を動的に使用禁止にするもう 1 つの選択肢は、name 属性が 'ASYNC_NO_RESPONSE'、content 属性が 'true' の meta 要素を持つ結果文書にフラグを設定する方法です。OracleAS Wireless XML のサイレント・メタ要素の一例は、<SimpleMeta name="ASYNC_NO_RESPONSE" content="true" /> です。
可変引数のサポート	False	対応する非同期アプリケーション・パラメータよりも多くのパラメータ値がユーザー・リクエストに含まれている可能性がある場合は、このフラグが役立ちます。余分な値はすべて最後のパラメータの値として追加されます。

表 10-2 サービス構成パラメータ（続き）

名前	デフォルト	セマンティクス
セッションレス	False	このフラグは、対話型のユーザー相互作用を提供しないアプリケーションに対して任意に選択できます。このようなアプリケーションは、アプリケーションの起動に関する最終結果を常に戻します。これは、結果文書にハイパーリンクまたはフォーム・コントロールが含まれていないことを意味します。このようなアプリケーションへのアクセスをリクエストするデバイスでは、セッションは保持されません。その結果、リソースを節約できます。

表 10-3 サイト構成パラメータ

名前	デフォルト	セマンティクス
ワーキング・スレッド	10	ワーキング・スレッドの数。
フィルタされたサブジェクト行の接頭辞	re:、fw:、[fwd:、fwd:	電子メールの件名行に接頭辞のリストを指定することによって、これらの接頭辞で開始するメッセージの件名行を無視し、ユーザー・コマンドとして解釈しないことを示します。たとえば、接頭辞 Re および Fwd を指定します。
システム・ヘルプ・コマンド	!H	コマンドの使用方法について一般的なヘルプを提供します。
エスケープ・コマンド	!E	現在のフォームの状態を消去します。
停止コマンド	!S	コマンド・シーケンスの終了をマークします。
アプリケーション・ヘルプ	Help	サービス・レベルのヘルプを提供します。
ログイン・コマンド	!L	ユーザー名とパスワードを使用してシステムにサイン・オンできます。
ログオフ・コマンド	!O	ユーザー・セッションをサイン・オフします。
コマンド・ライン区切り文字	;	複数のコマンドを使用するリクエストのコマンド・セパレータ。
コマンド接頭辞	.	記号直後のテキストがパラメータ値ではなく非同期短縮名であることを示す記号。これは、エスケープ・コマンドを使用せずにフォームの状態からエスケープする際に役立ちます。たとえば、.stk orcl は、コマンド接頭辞にピリオドを使用したコマンドです。
ヘルプ・ヘッダー	Usage -	アプリケーション・ヘルプ結果のヘッダー。

表 10-3 サイト構成パラメータ（続き）

名前	デフォルト	セマンティクス
ヘルプ・フッター		アプリケーション・ヘルプ結果のフッター。
デフォルト・アプリケーションのエイリアス名		ユーザー・リクエストが空の場合に起動されるアプリケーションの短縮名を指定します。値が指定されていない場合は、アプリケーション・サービスのヘルプ・ページが戻ります。

ユーザー・カスタマイズ・パラメータ

ユーザーは、非同期アプリケーションを起動するために、Wireless Customization Portal を介して各自のエイリアスを短縮名で作成します。ユーザー短縮名は、アプリケーションを指し示すエイリアス、または一連の非同期短縮名を表すエイリアスにできます。

たとえば、株価アプリケーションの短縮名である文字列 stk を表すエイリアス（s）を定義できます。株価アプリケーションを起動するために、ユーザーが s orcl コンテンツ（orcl はオラクル社のチッカー記号）を使用して非同期メッセージを発行すると、オラクル社の株価を受け取ることができます。文字列値 traffic ny;weather ny のエイリアスとして、文字列（tw）を作成できます。このようにすると、2 文字の入力で交通情報（traffic）と気象情報（weather）の 2 種類のアプリケーションを起動できます。

アプリケーションの起動例

アプリケーションの短縮名によるアプリケーションの起動

非同期可能なすべてのアプリケーションには、エンド・ユーザーがアクセスできるように短縮名を割り当てる必要があります。短縮名には、サイト全体でアプリケーションを一意に識別する名前を指定してください。アプリケーションを起動するには、info@oraclemobile.com など、非同期リスナーがリスニングするように構成されているサイトのアクセス・ポイントにメッセージを送信する必要があります。コマンドラインの書式は次のとおりです。

```
<Svc Short Name> <parm1> <parm2> . . .
```

次の例では、メッセージがサイトのアクセス・ポイント info@oraclemobile.com に送信され、短縮名 st の株価アプリケーションを起動します。このアプリケーションには、パラメータとして銘柄コードが必要です（この例では ORCL を指定しています）。

図 10-3 サービスの短縮名による起動

アプリケーションに関連付けられたアクセス・ポイントによる起動

各アプリケーションには、アプリケーションのアクセス・ポイントを関連付けることができます。たとえば、電子メール・アドレス stock@oraclemobile.com を使用して株価アプリケーションを識別できます。このアプリケーションはリクエスト・メッセージの宛先アドレスで識別されるため、コマンドラインにアプリケーションの短縮名を指定する必要はありません。コマンドラインでは、銘柄コードなどのアプリケーション・パラメータのみが必要です。

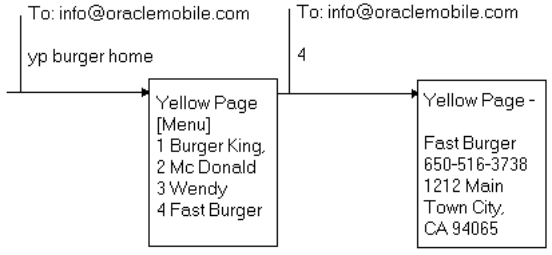
すべてのシステム・コマンド（help など）も、アプリケーションに関連付けられているアクセス・ポイントに対して発行できます。非同期リスナーは、サイトのアクセス・ポイントに送信されるのと同じ方法でこれらのコマンドを解析します。

図 10-4 サービスに関連付けられたアクセス・ポイントによる起動

メニュー機能

メニュー機能として、HTTP モデルに似ている機能が用意されています。アプリケーションを起動すると、メニューを使用したメッセージの戻りがトリガーされます。各メニュー項目の先頭には番号が付いています。コンテンツにメニュー項目番号を含めた別のメッセージを発行することで、ユーザーはその項目を選択できます。これによってアプリケーション機能が拡張され、ユーザー相互作用が効率的になります。短縮名 yp を持つイエロー・ページ・アプリケーションでは、カテゴリと地域の 2 つのユーザー・パラメータが予期されます。ユーザーは、ランドマークである burger（ハンバーガー・ショップ）および home（自宅）などの値を指定してアプリケーションを起動します。アプリケーションでは、ユーザーの最寄りの地域にあるすべてのハンバーガー・ショップが検索されます。アプリケーション結果から戻されるメッセージには、ハンバーガー・ショップの店舗名一覧が含まれています。ユーザーは、選択した店舗の詳細情報を取得する別のメッセージを発行します。

図 10-5 メニュー機能



フォーム機能

フォームは、ユーザーの入力を要求するアプリケーションの起動結果です。Async にとって理想的なユーザー相互作用は、ユーザーが入力パラメータをフォームに入力してメッセージのラウンドトリップ数を増やすのではなく、コマンドラインに入力することです。

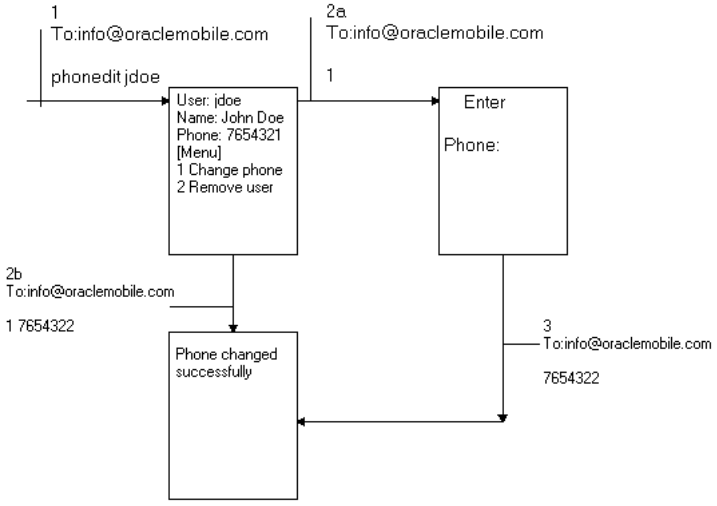
図 10-6「フォーム機能」に、電話帳アプリケーションに可能な相互作用を示します。`phonedit` コマンドを使用すると、ユーザーは特定ユーザーの電話番号を検索し、編集できます。コマンドのパラメータとしては名前が予期されます（次の例では、jdoe がこのパラメータに相当します）。jdoe の情報がメニューとともに戻され、デバイス・ユーザーは電話番号を編集したり、ユーザー jdoe を削除できます。電話番号の編集には、次の 2 つのオプションがあります。

- パラメータを入力せずに選択する方法。図のボックス 2a は、この使用例を表しています。ユーザーに対して、新規電話番号の入力を求めるフォームが戻されます。デバイス・ユーザーは、メッセージ本文に新規電話番号を使用して新規メッセージを作成します。

または

- 必須パラメータを指定して選択する方法。図のボックス 2b は、この使用例を示しています。デバイス・ユーザーには、選択 1（Change phone）へのレスポンスでフォームが戻されることがわかります。したがって、選択を収集するためにパラメータ値（phone number）を指定します。これにより、メッセージのラウンドトリップが減少します。

図 10-6 フォーム機能

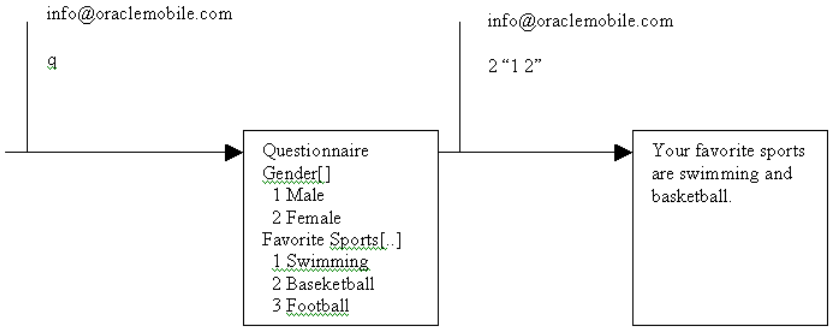


フォームのフィールドと選択オプション

フォームには、1 つまたは複数の選択肢を指定できるフィールドが使用されます。フィールドは、HTML のチェックボックスまたはプルダウン・メニュー構成と似ています。デバイス・ユーザーの選択を簡素化するために、各選択オプションの前には番号が付きます。選択する項目の接頭辞番号でレスポンスすることによって、項目がフィールドに入力されます。複数の選択を許可するフィールドには、入力マーカー [..] が付いています。このマーカーによって、単一の選択を許可するフィールドと複数の選択を許可するフィールドが区別されます。単一の選択を許可するフィールドには、単一選択マーカー [] が付きます。複数の値を選択する場合は、ユーザーが返信する値の周りを引用符で囲む必要があります。

次の図は、フォーム選択フィールドを示しています。Gender は単一選択フィールドで、Favorite Sports は複数選択フィールドです。

図 10-7 フォームのフィールドと選択オプション



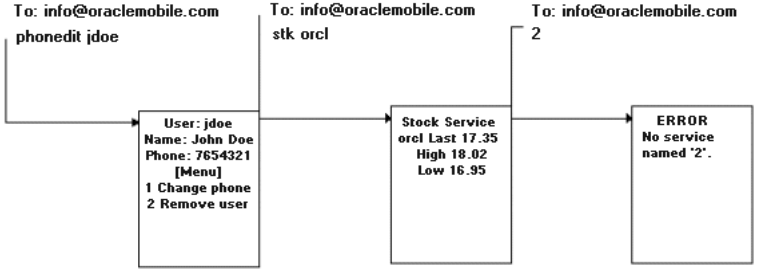
現行のメニューの状態

セッションはユーザーごとに保持されるため、メニュー・ナビゲーションが可能です。現行のメニューとは、ユーザーが Async から受信した最新メニューです。現行のメニューの状態は、ユーザー・セッション中は Async で保持されます。

ユーザーのメニュー選択は、常に現行のメニューに適用されます。ユーザーがまだメニューを受信していない場合、Async ではユーザーが指定した番号と同じ短縮名を持つアプリケーションが検索されます。このアプリケーションが見つからない場合は、エラーが戻ります。

短縮名またはアクセス・ポイントを使用してアプリケーションを起動すると、以前のアプリケーション起動で作成されたメニュー状態が自動的に取り消されます。図 10-8 に示すように、phonedit アプリケーションの起動に対するレスポンスとしてメニューが戻ります。その後、stk サービスをリクエストするメッセージが発行されます。これによって、phonedit アプリケーションの起動時に作成されたメニュー状態が消去されます。メニュー項目を選択しようとすると、Async からエラー・メッセージがトリガーされます。

図 10-8 現行のメニューの状態



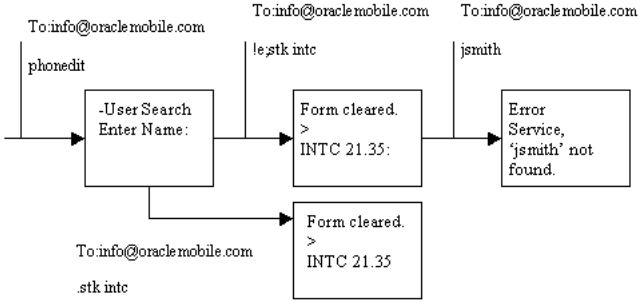
現行のフォームの状態

現行のフォームの状態は、ユーザーがフォーム・メッセージを受信するたびにユーザー・セッションで作成されます。後続するユーザー・リクエストでフォーム・パラメータ値を指定して、以前のフォーム・メッセージでリクエストされたパラメータを入力します。ユーザーがフォームに入力せずに別のアプリケーションを起動することを決めた場合は、エスケープ・コマンドを発行して現行のフォームの状態を取消しできます。フォーム状態を消去した後は、ユーザーが発行したフォーム・パラメータ値はすべて無効とみなされます。現行のフォームの状態がないフォーム・パラメータ値に対するレスポンスとして、エラー・メッセージが戻ります。または、短縮名で識別されるサービスの起動が後続する場合は、ユーザーがショートカットとして短縮名を使用してコマンド接頭辞を発行することによって、フォームの状態が消去されます。

図 10-9「現行のフォームの状態」に、フォーム状態の例を示します。デバイス・ユーザーは、パラメータを指定せずに phonedit アプリケーションを起動します。検索名の入力を求めるフォーム・メッセージが戻されます。デバイス・ユーザーが phonedit アプリケーションを起動せずに別のアプリケーション（stk など）の起動を決めた場合、最初の手順は、Async で stk コマンドが phonedit アプリケーションから予期する名前値として処理されないように、フォーム状態を消去することです。これで新規の stk コマンドを発行できます。2 つのコマンドをデフォルトのコマンド・セパレータ（;）で区切ると、この 2 つの手順を 1 つのメッセージに結合できます。

ピリオド（.）などのコマンド接頭辞を使用して短縮名 stk を発行すると、少ないキーストロークでフォーム状態を消去できます。

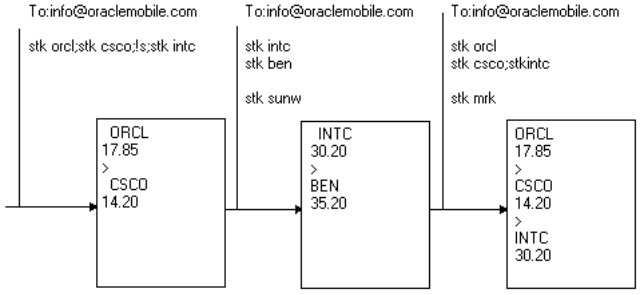
図 10-9 現行のフォームの状態



1 つのメッセージ内の複数のコマンド

1 つのメッセージで複数のコマンドを発行できます。この場合は、各コマンドを構成可能なコマンド・セパレータ（デフォルトは [;]）で区切って 1 行で発行できます。また、各コマンドを別々の行で発行することもできます。最初の空白行または停止コマンド（!s）が検出されると、コマンド・シーケンスの終了がマークされます。このマークより後のコマンドは解析されません。

図 10-10 1 つのメッセージ内の複数のコマンド

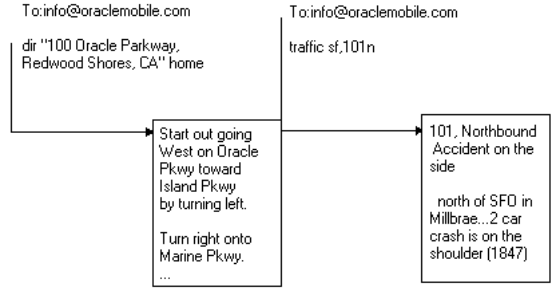


パラメータ・セパレータ

非同期アプリケーションは、複数のパラメータを必要とする場合があります。デフォルトのパラメータ・セパレータは空白です。パラメータ値に空白が含まれている場合は、二重引用符で囲んで 1 つのパラメータ値を表すことができます。パラメータ・セパレータは、アプリケーション・レベルで構成可能です。

図 10-11 に、道案内アプリケーションを示します。このアプリケーションは、from（出発地）と to（行先）の両方の住所を必要とします。from 住所は、値全体を二重引用符で囲んで指定されています。to は、ユーザー・プロファイルからのランドマークである home（自宅）として指定されています。ユーザーから送信される第 2 のメッセージでは、交通情報アプリケーションがリクエストされています。このアプリケーションはパラメータの区切り文字としてカンマ（,）を使用するように構成されており、ユーザーはパラメータ値をカンマ（,）で区切って指定します。

図 10-11 パラメータ・セパレータ



非同期アプリケーションの作成

非同期アプリケーションは、ブラウザ・ベースのアプリケーションと同じ方法で開発されます。アプリケーション・プロバイダは、HTTP プロトコルを介してデバイスからユーザー・パラメータを受信し、結果を OracleAS Wireless XML または XHTML/XForms フォーマットでレスポンスとして返します。非同期クライアントの要件はわずかで、テキスト・メッセージの送受信機能のみが必要です。このため、Async はすべてのタグをサポートしているわけではありません。サポートされるタグの Async 固有の解析に関する詳細およびこのチャネルに対する文書の作成方法は、第 8 章「モバイル・ブラウザおよび音声アプリケーションの作成」を参照してください。第 3 章「OracleAS Wireless Developer Kit」も参照してください。.

開発者は、アカウントまたはロギング目的でサービス・リクエストのデバイス情報を取得することが必要になる場合があります。この情報は、HTTP ヘッダー（サービスが HTTP アダプタに基づいて作成される場合）を介して特定できます。関係する 2 つのユーザー・デバイス・ヘッダーは、次のとおりです。

- x-oracle-user.deviceaddress
- x-oracle-user.deviceaddresstype

XMS メッセージ・センター

XMS コンポーネントには、デフォルトで MMS センター（MMSC）機能が組み込まれており、MM1 メッセージ通知プロトコルをサポートします。受信者のデバイスに MMS ブラウザがある場合、通知メッセージは MMS ブラウザに送信され、HTTP を使用してメッセージを取得します。コンテンツは、OracleAS Wireless によって格納および配信されます。MMS をサポートするために、唯一必要な外部コンポーネントは、通知メッセージを送信するための標準 SMSC です。

MMS ブラウザが通知メッセージを受信する場合は、MM1 仕様に従って、HTTP を使用して XMSC に接続します。メッセージはエンコードされた MMS 形式で提供されます。

XMSC は、別の電話への MO（モバイル発信）メッセージもサポートしています。この場合、メッセージは XMSC に格納され、通知メッセージが前述のようにターゲット・デバイスに送信されます。

XMSC では、他のメッセージ・チャネル用のメッセージ記憶域および通知もサポートされます。たとえば、ユーザーのデバイスは MMS 機能のみを備えていると仮定します。マルチメディア・コンテンツが含まれているメッセージが送信されると、XMS は、URL を含んでいるテキストのみの SMS メッセージを送信します。ユーザーは、この URL を標準的なブラウザに入力して、メッセージを表示できます。

構成

XMSC 機能を使用するには、次の手順を実行します。

サーバー側

- SMS ドライバを設定する必要があります。ドライバの設定方法に関する一般的な情報については、10-35 ページの「トランスポート・コンポーネント」を参照してください。この項では、OracleAS Wireless に付属するドライバの構成方法について詳しく説明します。
- サイトのホスト名とポートを設定します（設定されていない場合）。

  「システム」→「Wireless サーバー」→「サイト管理」→「HTTP、HTTPS の構成」にアクセスします。

  「URL」セクションで、ホスト名とポートを追加します。
- 必要な場合は、XMSC を有効にします（デフォルト状態で有効です）。

  「システム」→「Wireless サーバー」→「サイト管理」にアクセスし、「コンポーネント構成」を展開します。**「XMS 構成」**をクリックします。

  「XMS センター」セクションで、「XMSC の有効化」チェックボックスを選択します。

クライアント（ハンドセット）側

モバイル着信（MT）メッセージ（デバイスに送信するメッセージ）の場合は、前述の設定で十分です。XMSC は完全な MMSC であり、MO（モバイル発信）メッセージもサポートしています。つまり、OracleAS Wireless が中継を担当して、ある電話から別の電話にメッセージを送信します。この場合、次のような電話の構成が必要です。

OracleAS Wireless インスタンスを送信メッセージング・サーバーとして使用するように MMS ブラウザを構成する必要があります。実際の手順は電話によって異なります。次の手順は、Sony Ericsson T68i の場合です。

1. 「Messages」を選択します。
2. 「MMS」（2）を選択します。
3. 「Options」（5）を選択します。
4. 「Message Server」（6）を選択します。
5. MM1 リスナー・サーブレットの URL を入力します。次の形式で入力します。

   ```
   http://<hostname>:<port>/xms/mm1
   ```

注意： ダイアルアップまたは電話に用意されている WAP 設定を使用して、電話からホスト名に接続可能であることが必要です。

デバイス・チャネルの選択

受信者がユーザー名で指定されると、XMS ランタイムによって、ユーザーが使用可能なデバイスおよびユーザーの現在の連絡ルールに基づいて、使用するデバイスとチャネルが判断されます。

デバイスの自動選択

ユーザーが多数の異なるデバイスを持っている場合は、XMS ランタイムによって、使用に最適なデバイスとチャネルが選択されます。チャネルの選択では、メッセージのコンテンツに加え、ユーザーの優先順位も考慮されます。たとえば、SMS と MMS の両方のメッセージ機能がある携帯電話を持っているユーザーを想定してください。プレーン・テキスト・メッセージがユーザーに送信される場合は、SMS チャネルが選択されます。これは、プレーン・テキストのメッセージの場合は SMS で十分であり、一般的に SMS によるメッセージ送信のほうが MMS よりも安くて高速であることから判断されます。一方、マルチメディア・コンテンツを含んでいるメッセージの場合は、MMS を介して XMS メッセージが送信されます。

所在情報の統合

OracleAS Wireless では、各ユーザーの所在設定を保持できます。所在には、ユーザーの位置と関連する連絡ルールを指定します。連絡ルールには、各状況でのユーザーの優先配信チャネルを指定します。連絡ルールの詳細は、第 15 章「ユーザー・カスタマイズの有効化」を参照してください。

ユーザーに連絡ルールが設定されている場合は、前述されているように、そのルールにはチャネルの自動選択全体に優先権が割り当てられます。また、連絡ルールでは、ユーザーがメッセージを受信しないブラックアウト期間を指定できます。連絡ルールにブラックアウト期間が設定されていると、XMS はブラックアウト期間が終了するまでメッセージの配信を延期します。

トランスポート・コンポーネント

トランスポート・レイヤーは OracleAS Wireless メッセージ・システムの基本部分です。トランスポート・レイヤーには、Transport API、トランスポート・サーバーおよび Driver API が含まれています。この項では、ビルトイン・トランスポート・ドライバ、新規ドライバの開発方法、およびトランスポート・サーバーの拡張方法について説明します。

現在の Transport API は、以前のリリースとの互換性があります（当然のことながら、以前のリリースを使用している場合、このリリースで組み込まれた新機能は使用できません）。機能拡張された現在の Driver API は、以前のリリースとの互換性がありません。ただし、以前のリリースのすべてのビルトイン・ドライバは、このリリースにあわせて機能拡張されています。以前のリリースのビルトイン・ドライバは、このリリースのトランスポート・サーバーでは使用できません。現行のビルトイン・ドライバのみを使用してください。インタフェースの拡張に加え、ビルトイン・ドライバには新しい多数の機能が組み込まれています。

ドライバが新規ドライバ・インタフェースを実装すると、以前のリリースでのユーザー独自のカスタム組込みのドライバを現行のトランスポート・サーバーで動作するようにアップグレードできます。ドライバのロジックはそれほど大幅には変わりません。主要な変更はインタフェースに限定されます。

ビルトイン・ドライバ

Nokia MMS ドライバ

このドライバは、Nokia MMSC（Multimedia Messaging Service Center）との MMS メッセージの送受信機能を提供します。このドライバには Nokia MMS Java Library v1.1 が必要です。ドライバは、MMSC への 2 つの TCP 接続（メッセージの送信用に 1 つ、受信用に 1 つ）をオープンします。

必要なサード・パーティ・ソフトウェア このドライバには Nokia MMS Java Library v1.1（MMSLibrary.jar）が必要です。このライブラリは Forum Nokia（http://www.forum.nokia.com）から入手できます。このライブラリを該当するファイルの CLASSPATH に追加する必要があります。対象となるファイルは、
UNIX では `$ORACLE_HOME/opmn/conf/opmn.xml`、
Windows では `$ORACLE_HOME¥opmn¥conf¥opmn.xml` です。

クラス名

```
oracle.panama.messaging.transport.driver.mms.NokiaMMSDriver
```

構成

`mms.nokia.account.id`: Nokia のアカウント ID または電話番号。このパラメータは必須です。

mms.nokia.mmsc.url: Nokia MMSC の URL。このパラメータは必須です。

mms.nokia.debug: 特別なデバッグ情報をファイルに記録できます。オプションは、true（デバッグ可能）、false または空白（デバッグ不可）です。

mms.nokia.log.filename: 特別なデバッグ情報のログ・ファイル名。デフォルトのファイル名は NokiaMMSDriver.log です。

mms.nokia.receive.host: 論理ローカル・ホスト名または IP アドレス。未指定の場合、この値はローカル・ホストから導出されます。

mms.nokia.receive.port: MMSC からの MMS メッセージを受信するポート。デフォルト値は 7000 です。

mms.nokia.receive.mode.async: MMSC からの MMS メッセージを受信するために非同期モードを設定します。オプションは、true（使用可能）、false または空白（使用不可）です。非同期モードの詳細は、Nokia MMS Java Library v1.1 に付属のドキュメントを参照してください。

CMG MMS ドライバ

このドライバは、CMG MMSC（Multimedia Messaging Service Center）との MMS メッセージの送受信機能を提供します。このドライバは CMG MMSC API for VAS v1.01 を使用します。

必要なサード・パーティ・ソフトウェア

このドライバには、CMG MMSC API for VAS v1.01（mmscapi.jar および mmscapi.war）が必要です。これらのファイルは CMG（http://www.cmgwds.com）から入手できます。mmscapi.jar ライブラリを該当するファイルの CLASSPATH に追加する必要があります。対象となるファイルは、
UNIX では、$ORACLE_HOME/opmn/conf/opmn.xml、
Windows では $ORACLE_HOME¥opmn¥conf¥opmn.xml です。

クラス名

```
oracle.panama.messaging.transport.driver.mms.CMGMMSDriver
```

構成

- mms.cmg.account.id

  CMG MMSC のアカウント ID または電話番号。このパラメータは必須です。

- mms.cmg.account.password

  CMG MMSC のアカウント・パスワード。このパラメータは必須です。

- mms.cmg.config.file

  CMG MMSC API のコア構成ファイルのパス。このパラメータは必須です。このファイルの詳細は、CMG MMSC API パッケージに付属のユーザー・マニュアルを参照してください。このドライバには、サンプル構成ファイル（$ORACLE_HOME¥wireless¥messaging¥drivers¥cmg¥CMGMMSDriver.cfg）が収められています。

- mms.cmg.debug

  特別なデバッグ情報をファイルに記録できます。オプションは、true（デバッグ可能）、false または空白（デバッグ不可）です。

- mms.cmg.billing.category

  MMSC 請求カテゴリ（オプション）。この値は、MMSC にカスタム請求カテゴリ情報を送信するために使用されます。請求カテゴリの詳細および例は、CMG MMSC API パッケージに付属のユーザー・マニュアルを参照してください。

- mms.cmg.billing.price

  MMSC 請求価格（オプション）。この値は、MMSC にカスタム請求価格情報を送信するために使用されます。請求価格の詳細および例は、CMG MMSC API パッケージに付属のユーザー・マニュアルを参照してください。

その他の構成

MMS メッセージを受信するドライバを構成するには、次の追加手順を実行する必要があります。

1. 次の手順で、mmscapi.war ファイルを cmgmmsc.ear ファイルにパッケージ化します。
   - $ORACLE_HOME¥wireless¥messaging¥drivers¥cmg¥cmgmmsc.ear.zip を空のディレクトリに解凍します。次のディレクトリ構造が作成されます。
     * ¥META-INF¥application.xml
     * ¥META-INF¥MANIFEST.MF
2. mmscapi.war をこのディレクトリにコピーし、ファイル名を cmgmmsc.war に変更します。最終的に次のディレクトリ構造となります。
   - ¥META-INF¥application.xml
   - ¥META-INF¥MANIFEST.MF
   - ¥cmgmmsc.war
3. このディレクトリ構造を圧縮し、圧縮ファイル名を cmgmmsc.ear に変更します。
4. cmgmmsc.ear を $ORACLE_HOME¥wireless¥j2ee¥applications¥ にコピーします。

5. $ORACLE_HOME¥wireless¥j2ee¥config¥wireless-web-site.xml に、次の記述を追加します。

```
<web-app application="cmgmmsc" name="cmgmmsc" root="/cmgmmsc"
load-on-startup="true"/>
```

6. $ORACLE_HOME¥wireless¥j2ee¥config¥wireless-server.xml に、次の記述を追加します。

```
<application name="cmgmmsc" path="../applications/cmgmmsc.ear" auto-start="true"
/>
```

7. OracleAS Wireless インスタンスを起動します。cmgmmsc.ear ファイルがオート・デプロイされます。

8. オート・デプロイ終了後、$ORACLE_HOME ¥wireless¥j2ee¥applications¥cmgmmsc¥cmgmmsc¥WEB-INF¥web.xml に次のセクションを追加することによって、トレース・ディレクトリおよびファイル名を編集します。

```
<servlet-mapping>
    <servlet-name>
      HttpReceive
    </servlet-name>
    <url-pattern>
      /HR
    </url-pattern>
</servlet-mapping>
```

9. 次のファイルのバックアップを作成（ファイルの拡張子を .jar 以外に変更）します。
$ORACLE_HOME¥wireless¥lib¥log4j-core.jar
$ORACLE_HOME¥wireless¥lib¥log4j.jar

   ファイルをコピーします。
   コピー元：
   $ORACLE_HOME
   ¥wireless¥j2ee¥applications¥cmgmmsc¥cmgmmsc¥WEB-INF¥lib¥log4j-1.2.5.jar
   コピー先：
   $ORACLE_HOME¥wireless¥lib¥log4j-core.jar

10. OracleAS Wireless インスタンスを再起動します。

> **注意：** OracleAS Wireless インスタンス内でサーブレット HttpReceive を実行し、RMI が CMG MMS ドライバと通信する必要があります。詳細は、CMG MMSC API パッケージに付属のユーザー・マニュアルを参照してください。

MM7 ドライバ

このドライバは、3GPP TS 23.140 V5.3.0 の仕様に基づいて MM7 プロトコルを使用し、MMS メッセージを MMSC（Multimedia Messaging Service Center）に送信する機能を提供します。

クラス名

```
oracle.panama.messaging.transport.driver.mms.MM7Driver
```

構成

mms.mm7.url: MMSC/MM7 にアクセスする URL。このパラメータは必須です。

mms.vaspid: （オプション）このアプリケーション（VASP）の MMSC に対する識別子。

mms.vasid: （オプション）起動元アプリケーションの識別子（たとえば、News）。

mms.local.hostname: 論理ローカル・ホスト名または IP アドレス。未指定の場合、この値はローカル・ホストから導出されます。

mms.local.port: ローカル・リスニング・ポート。デフォルト値は 80 です。

mms.default.encoding: デフォルトの MM7/HTTP エンコード形式。デフォルトのエンコードは UTF-8 です。

debug: 特別なデバッグ情報をファイルに記録できます。オプションは、true（デバッグ可能）、false または空白（デバッグ不可）です。

CIMD ドライバ

CIMD（Computer Interface to Message Distribution）は、Nokia 指定の SMS プロトコルです。OracleAS Wireless サーバー製品には、送受信機能を持つドライバの 1 つとして CIMD ドライバのビルトイン実装が用意されています。ドライバは、メッセージ送受信用に SMSC への TCP/IP 接続を 1 つオープンします。このドライバは、テキスト・メッセージとバイナリ・メッセージ（vCard、vCalendar など）の送信を処理できます。このドライバで受信できるのはテキスト・メッセージのみです。

クラス名

```
oracle.panama.messaging.transport.driver.sms.CIMDDriver
```

構成

ドライバを作成するときに、次のパラメータを指定する必要があります。

- sms.account.id

  SMSC のアカウント ID です。通常は、オペレータが割り当てる短縮番号を使用します。このパラメータは必須です。

- `sms.cimd.system.userid` および `sms.cimd.system.password`

  短縮番号に加え、オペレータは SMSC にログインするためのユーザー ID とパスワードを指定します。この 2 つのパラメータは必須です。

- `sms.server.host` および `sms.server.port`

  この情報は、SMSC への TCP/IP 接続をオープンするためにドライバが使用します。

- `sms.message.maxchunks`

  1 つのメッセージの最大許容チャンク数です。この数値を超えた後のチャンクは無視されます。デフォルト値は -1 です。負の値は無制限を意味します。

- `sms.message.chunksize`

  各チャンクの最大バイト数です。デフォルト値は 140 です。

- `sms.server.default.encoding`

  テキスト・メッセージのエンコード・スキームを指定します。デフォルト値は IA-5 です。

- `sms.window.size`

  ウィンドウのサイズを指定します。ウィンドウイングをサポートしない場合は、ウィンドウ・サイズを 1 に設定します。

- `sms.send.alive.packet.interval`

  アライブ・パケットを送信する時間間隔（ミリ秒単位）。負の値はアライブ・パケットが送信されないことを示します。通常は指定時間を経過すると、ユーザーは SMSC から自動的にログアウトされます。このパラメータは、稼働中の SMSC への接続を維持するために必要です。自動ログアウトの設定よりも短い時間間隔を指定します。

VVSP ドライバ

このドライバは、Via Vodafone Services Platform（VVSP）との SMS メッセージおよび MMS メッセージの送受信機能を提供します。このドライバを使用するには、ドライバをアプリケーションとして Vodafone Mobile Office に登録する必要があります。アプリケーションの登録方法や、このドライバの実行に必要な資格証明の取得方法の詳細は、Vodafone Mobile Office（http://www.mobileoffice.vodafone.com）にお問合せください。

クラス名

```
oracle.panama.messaging.transport.driver.vvsp.VVSPDriver
```

構成

vvsp.sms.address: カンマで区切られた SMS ネットワーク ID（MSISDN または短縮コード）のリスト。

例 : *8205, 8206*

vvsp.sms.country: 前述の SMS ネットワーク ID を VVSP に登録する際の対応する 2 文字の国コードがカンマで区切られたリスト。ISO 3166-1-alpha-2 コード要素規則が適用されます。

例 : uk, de

vvsp.sms.url: VVSP SMS ゲートウェイの URL。デフォルト値は、https://vvsp.vodafone.net/gns/sms です。

vvsp.sms.id: VVSP に登録されている SMS アプリケーション・インスタンスの ID。

vvsp.sms.password: SMS アプリケーション・インスタンスのパスワード。

vvsp.sms.onetimepassword: パスワード更改用の SMS ワンタイム・パスワード（OTP）。VVSP でのテスト・フェーズに固定の OTP を使用する場合のみ、このフィールドを使用します。例 : 69696969

vvsp.mms.address: カンマで区切られた MMS ネットワーク ID（MSISDN または短縮コード）のリスト。

例 : 8005, 8006

vvsp.mms.country: 前述の MMS ネットワーク ID を VVSP に登録する際の対応する 2 文字の国コードがカンマで区切られたリスト。ISO 3166-1-alpha-2 コード要素規則が適用されます。

例 : uk, de

vvsp.mms.url: VVSP MMS ゲートウェイの URL。デフォルト値は、https://vvsp.vodafone.net/gns/mms です。

vvsp.mms.id: VVSP に登録されている MMS アプリケーション・インスタンスの ID。

vvsp.mms.password: MMS アプリケーション・インスタンスのパスワード。

vvsp.mms.onetimepassword: パスワード更改用の MMS ワンタイム・パスワード（OTP）。VVSP でのテスト・フェーズに固定の OTP を使用する場合のみ、このフィールドを使用します。例 : 69696969

vvsp.admin.url: VVSP 管理ゲートウェイの URL。デフォルト値は、https://vvsp.vodafone.net/gns/admin です。

vvsp.admin.log: 管理ゲートウェイによって発行される新しいパスワードを格納するログ・ファイル。デフォルト値は、VVSPAdmin.log です。

vvsp.local.hostname: VVSP から通知を受信するためのバインド先論理ローカル・ホスト名 /IP。未指定の場合、この値はローカル・ホストから導出されます。通知の受信に使用される完全な URL は、http://vvsp.local.hostname:vvsp.local.port/ です。

vvsp.local.port: ローカル・リスニング・ポート。デフォルト値は 80 です。

vvsp.ssl.trustStore: SSL trustStore ファイル（jks 形式）に対するパス。空白の場合は、組込み VVSP による証明書ファイルが自動的にロードされます。ほとんどの場合、このパラメータを変更する必要はありません。

vvsp.ssl.trustPassword: trustStore ファイルのパス句。ない場合は空白（デフォルト）のままにします。

vvsp.ssl.keyStore: SSL keyStore ファイル（pkcs12 形式）に対する完全パス。VVSP からの SSL クラス 3 クライアント証明書がある場合は、このパスを指定する必要があります。

vvsp.ssl.keyPassword: keyStore ファイルのパス句。ない場合は空白のままにします。

vvsp.default.encoding: デフォルトの HTTP エンコード形式。デフォルト値は UTF-8 です。

vvsp.sms.maxchunks: SMS メッセージの最大長を超えた場合に許可する SMS メッセージの最大チャンク数 （1 つのチャンク当たりの最大長は、テキスト SMS の場合は 160 文字、バイナリ SMS の場合は 140 文字)。チャンクを制限しない場合は -1 を入力します。デフォルト値は -1 です。

vvsp.retrieve.flush.freq: 通知が行方不明なために VVSP で待機している未取得のメッセージを取得およびフラッシュするために周期的にポーリングする頻度（秒単位)。ポーリングを無効にする場合は 0 を入力します。デフォルト値は 600 秒です。

vvsp.debug: 特別なデバッグ情報を記録できます。オプションは、true（デバッグ可能)、false または空白（デバッグ不可）です。

その他の構成 : パスワードの更改

VVSP からの SSL クラス 3 クライアント証明書がない場合、アプリケーション・インスタンスのパスワードはプラットフォームによって定期的に失効します。パスワードが失効すると、VVSP は、登録されているアプリケーション・オペレータのハンドセットに、ワンタイム・パスワード（OTP）を指定して SMS を送信します。VVSP ドライバがパスワード更改手続きを完了し、VVSP からアプリケーション・インスタンスの新規パスワードを受信できるように、アプリケーション・オペレータはこの OTP を入力する必要があります。オペレータが OTP を入力できるように、ユーティリティが用意されています。このユーティリティは、OTN（http://otn.oracle.com/tech/wireless/integration/）にあります。説明を参照しながらユーティリティをダウンロードし、OracleAS Wireless インスタンスにインストールできます。インスタンスにインストールした後は、http://<instance_hostname>:<instance_port>/vvsp/vvspdriverotp.jsp でユーティリティにアクセスできます。

このユーティリティを使用してオペレータが OTP を入力すると、VVSP ドライバはパスワードを更新し、通常の操作を続行します。

オペレータが短時間（およそ 4 分以内）で OTP を入力できない場合、VVSP は OTP が失効したとみなして、ドライバによるパスワードの更改処理が再度開始され、VVSP から有効な新しい OTP がオペレータのハンドセットに送信されます。その後、オペレータはユーティリティを使用して、この新しい OTP を入力する必要があります。

注意： VVSP ドライバは、OracleAS Wireless Tools のドライバ・インスタンス構成では、新しいアプリケーション・インスタンスのパスワードを自動的に更新できません。その結果、メッセージング・サーバーが再開される時点で、新しいパスワードは消失します。したがって、操作が正常に実行されるように、OracleAS Wireless インスタンス（またはメッセージング・サーバー・プロセスのみ）を再開する前に、VVSP ドライバ・インスタンス構成を更新する必要があります。対応するアプリケーション・インスタンス ID および新しいパスワードは、OTP の入力に使用するユーティリティで使用できます。これらの値は、`VVSPAdmin.log` ファイル（または `vvsp.admin.log` パラメータで指定したファイル名）にも記録されます。管理者またはオペレータは、このファイルから新しいパスワードをコピーし、VVSP ドライバ・インスタンス構成でパスワードを手動で更新する必要があります。この処理は、今後のリリースで自動化される予定です。

定期的にパスワード更改に関係する手順を手動で実行せずに済むようにするには、VVSP から SSL クラス 3 クライアント証明書を取得する必要があります。詳細は、Via Vodafone Developer Site（http://www.via.vodafone.com）からアクセスできる『VVSP Gateway Overview Guide』を参照してください。

オラクル社では、定期的にパスワード更改が要求されないように、VVSP から SSL クラス 3 クライアント証明書を取得することをお薦めします。詳細は、Vodafone Mobile Office（http://www.mobileoffice.vodafone.com）からアクセスできるガイドを参照してください。

WCTP ドライバ

WCTP（Wireless Communication Transfer Protocol）の主な目的は、ワイヤー・ライン・システムとモバイル・デバイス間で英数字およびバイナリのメッセージを簡単に受渡しできる方法を提供することです。HTTP 1.1 は WCTP に推奨されているトランスポート・プロトコルです。組込みドライバは、エンタープライズ・ホストとして動作し、WCTP ゲートウェイに接続してメッセージを送受信します。このドライバでは、テキスト・メッセージの送受信を処理できます。ドライバは、HTTP リスナーを実装して着信メッセージをリスニングします。メッセージの送信に対するステータス・レポートもサポートしています。

クラス名

```
oracle.panama.messaging.transport.driver.sms.WCTPDriver
```

構成

ドライバを作成するときに、次のパラメータを指定する必要があります。

- `send-host`、`send-port` および `send-page`

  WCTP ゲートウェイのホスト、ポートおよびページです。

- `receive-host`、`receive-port` および `receive-page`

  WCTP ドライバによって実装される HTTP サーバーが着信メッセージをリスニングするホスト、ポートおよびページです。

- `receive-proxy-host` および `receive-proxy-port`

  この情報は、プロキシ・サーバーがある場合に必要です。WCTP XML メッセージを解析する際に必要となります。これらのパラメータはオプションです。

- `maxchunks`

  1 つのメッセージの最大許容チャンク数です。この数値を超えた後のチャンクは無視されます。デフォルト値は -1 です。負の値は無制限を意味します。

- `chunksize`

  各チャンクの最大バイト数です。デフォルト値は 160 です。

- `notify-when-queued`

  メッセージがキューされたときに通知が必要な場合は、true を指定します。

- `notify-when-delivered`

  メッセージが配信されたときに通知が必要な場合は、true を指定します。

- `notify-when-read`

  メッセージが読み込まれたときに通知が必要な場合は、true を指定します。

- `multi-recipient-message-support`

  1 つのメッセージを複数の受信者に送信する場合は、true を指定します。ture でない場合は、個別のメッセージが受信者リストの各受信者に送信されます。

データ通信ドライバ

OracleAS Wireless では、データ通信ドライバによって、Nokia 社製電話を介して SMS メッセージを送受信できます。ドライバ・コードでは、テキストおよびバイナリのメッセージを送信できます。データ・ケーブルを介して PC に接続されている場合は、テキスト・メッセージの受信のみ可能です（5110 用ドライバでテスト済）。

設定の詳細

Windows プラットフォームでデータ通信ドライバを設定するには、次の手順に従います。

設定を開始する前に、次の準備が整っていることを確認してください。

- Java 通信パッケージ（http://Java.sun.com/products/Javacomm/ からダウンロード可能）
- Nokia 社製携帯電話（ドライバは 5100 および 6100 シリーズの電話でテスト済）
- Nokia 社製電話用データ・ケーブル（5100/6100 シリーズには 9 ピン RS-232C シリアル・ケーブル DAU-9P が必要）
- データ・パッケージのインストール CD（電話に組込みモデムがない場合）

設定手順は、3 つのカテゴリに分割されます。

- Java 通信パッケージのインストール
- データ・パッケージのインストール（この手順は、Nokia 5110 など、電話に組込みモデムがない場合にのみ必要）
- OracleAS Wireless メッセージング・サーバーの構成

前述の各カテゴリの詳細を、次に示します。

- Java 通信パッケージのインストール

  http://Java.sun.com/products/Javacomm/ から Javacomm20-win32.zip をダウンロードします。

  PlatformSpecific.html ファイル（Javacomm20-win32.zip ファイルを解凍すると参照できます）に指定されているすべての手順に従います。

  サンプルを実行して構成が完全に行われたことを確認します。このサンプルは zip ファイルに含まれています。

  > **注意：** サンプル・プログラムでシリアル・ポートがリストされない場合は、ディレクトリ <Java_HOME>¥jre¥lib の Javax.comm.properties ファイルをコピーしてください。<Java_HOME> は jdk のインストール位置を示します。<Java_HOME>¥bin¥Java にインストールされた Java は、OracleAS Wireless コンポーネントの実行に使用される Java です。

- データ・パッケージのインストール

  （この手順は、組込みモデムがない Nokia 社製の電話をデータ通信に使用する場合のみ必要）

  Nokia データ・パッケージをインストールする前に、「コントロール パネル」→「ポート」にアクセスします。システムのポートを確認します。

  1. 適切なデータ・ケーブルを使用して、PC のいずれかの COM ポートと Nokia 社製携帯電話を接続します。
  2. Nokia データ・パッケージをインストール（標準インストール）します。
  3. ソフトウェアのインストールが正常に終了した後、マシンを再起動します。
  4. 「コントロール パネル」→「ポート」にアクセスし、ポートがリストに追加されていることを確認します。たとえば、インストール前に COM1 と COM2 がリストされていた場合、インストール後は COM1、COM2 および COM3 がリストされます。この場合、COM3 は仮想ポートで、データ・パッケージによって電話と PC 間のデータ転送用に構成されたポートです。

ドライバの構成

クラス名

```
oracle.panama.messaging.transport.driver.datacommunication.DataCommunicationDriver
```

構成

- sms.datacommunication.port

  ケーブルを介して接続している電話と通信するためにドライバが使用するポートです。データ・ケーブルによる電話と PC の接続に使用するポート名を入力します。電話に組込みモデムがない場合は、データ・パッケージのインストール時に作成された仮想ポート名を入力する必要があります（COM3 など）。

- `sms.datacommunication.phone-no`

  メッセージング・サーバーが実行しているコンピュータにデータ・ケーブルを使用して接続されている携帯の電話番号です（たとえば、1-650-576-8055）。

- `sms.message.maxchunks`

  メッセージ・サイズがチャンク・サイズよりも大きい場合に送信するメッセージ・チャンクの最大数です。デフォルトでは、すべてのチャンクが送信されます。

- `sms.message.chunksize`

  チャンクの最大サイズです。デフォルト値は 150 です。

メッセージング・サーバーを実行する前に、`opmn.xml` ファイルの CLASSPATH に `comm.jar` のパスが指定されていることを確認してください。

WAP プッシュ PAP ドライバ

PAP（Push Access Protocol）は WAP プッシュ・ゲートウェイにアクセスするためのプロトコルです。OracleAS Wireless には、WAP プッシュの送信機能を提供する PAP ドライバのビルトイン実装が用意されています。

このドライバでは、次のコンテンツ・タイプを処理できます。

- `ContentTypes.WAP_PUSH`: WAP プッシュ・ゲートウェイがサポートするコンテンツ・タイプすべて。
- `ContentTypes.URL`（関連するリソースのコンテンツ・タイプが前述の 1 つである場合のみ）：ドライバはすべてのコンテンツ・タイプを受け入れ、プッシュ・ゲートウェイにメッセージを送信します。

PAP ドライバによって送信されたメッセージを WAP プッシュ・ゲートウェイが理解しない場合、PAP ドライバは失敗であることを示す致命的でない例外をスローします。

クラス名

```
oracle.panama.messaging.transport.driver.wap.PAPDriver
```

構成

- `pap.ppg.url`

  WAP プッシュ・ゲートウェイの URL。このパラメータは必須です。

- `pap.notifyto.url`

  WAP プッシュ・ゲートウェイが通知を送信できる URL。この値はオプションです。

- `pap.listento.notify`

  値を true または false に設定します。このフラグは、このドライバが通知をリスニングするかどうかを示します。

- `pap.source.reference`

  WAP プッシュ・ゲートウェイに対するこのドライバのソース参照。

- `pap.ppg.hostname`

  PPG ゲートウェイの論理ホスト名。PPG の電話会社 ID として使用されます。未指定の場合、この値は PPG URL から導出されます。

- `pap.local.hostname`

  論理ローカル・ホスト名または IP アドレス。未指定の場合、この値は、ローカル通知 URL またはローカル・マシン（存在する場合）から導出されます。

- pap.default.encoding

  デフォルトのコンテンツ・エンコード形式。未指定の場合は UTF-8 が使用されます。

- pap.version

  PPG がサポートする PAP バージョン。1.0 または 2.0 であることが必要です。

Instant Messaging（IM）ドライバ

Instant Messaging ドライバは、リアルタイム Instant Messaging（IM）を介して OracleAS Wireless アプリケーションにアクセスするために、エンド・ユーザーに一方向および双方向アクセスを提供します。IM ドライバは、Jabber を使用するオープンな XML ベースの Instant Messaging プラットフォームです。Yahoo、MSN、AOL、ICQ などの独自の IM ネットワークとも統合できます。このドライバによって、エンド・ユーザーは、選択した IM クライアントを介して、アラート通知を受け取り、アプリケーションを対話形式で使用できます。

Jabber の概要 Jabber は、Instant Messaging および所在情報用のオープンな XML ベースのプロトコルです。Jabber ベースのソフトウェアは、インターネットを介して何千ものサーバーにデプロイされ、世界中の 100 万人を超える人々に使用されています。Jabber は、クライアント・サーバー・アーキテクチャで構成され、ユビキタスな電子メール・ネットワークと共通点があります。Jabber サーバーは、完全に分散化されており、誰でも独自のサーバーを設定できます。メッセージは、受信者がユーザー名とホスト名（たとえば、username@hostname）でアドレス指定される電子メール・ネットワークなどで達成されます。Jabber ネットワークでは、ユーザーが接続する特定の Jabber サーバーのユーザー名とホスト名で構成される Jabber ID でユーザーが識別されます。Jabber のエンド・ユーザーは、他の Jabber ユーザーにインスタント・メッセージを送信するために、Jabber クライアントを使用して Jabber サーバーに接続します。ただし、これはインスタント・メッセージを届ける唯一の方法ではありません。Jabber には、拡張可能なモジュール化アーキテクチャが採用されています。Jabber は、Yahoo、MSN、AOL、ICQ などの独自の IM ネットワークに接続できるトランスポート・ゲートウェイを使用して、これらのネットワークを統合します。これによって、Jabber ユーザーは Yahoo、MSN、AOL または ICQ のユーザーと通信できます。

OracleAS Wireless で IM ドライバを使用するには、Jabber サーバーおよび OracleAS Wireless インスタンスの Jabber アカウントにアクセスする必要があります（エンド・ユーザー間でメッセージを送受信するために、IM ドライバが Jabber にログインする際の ID を使用）。また、IM ドライバには、OracleAS Wireless インスタンスが Yahoo、MSN、AOL または ICQ の IM ネットワークのユーザーと通信できるようにする構成パラメータが含まれています。IM ドライバでは、OracleAS Wireless を使用して接続している独自の IM ネットワークに対するアカウントがさらに必要です。それによって、これらのネットワークのエンド・ユーザーが OracleAS Wireless と通信できます。

サード・パーティの Jabber ソフトウェア Instant Messaging ドライバは、JabberBeans Java ライブラリを使用して Jabber Instant Messaging サーバーに接続します。そのためには、次のサード・パーティ・ソフトウェアが必要です。

表 10-4 Instant Messaging ドライバに必要なサード・パーティ・ソフトウェア

名前	指示	バージョン
JabberBeans	OracleAS Wireless には JabberBeans（バージョン 0.9.1）のコピーが含まれています。これよりも新しいバージョンの JabberBeans にアップグレードするには、次の説明に従ってください。 http://jabberbeans.jabberstudio.org から最新の `jabberbeans.jar` をダウンロードしてコピーします。コピー先は、UNIX の場合は $ORACLE_HOME/wireless/lib、Windows の場合は %ORACLE_HOME%¥wireless¥lib です。 ORACLE_HOME 値の例 : Solaris の場合 : ORACLE_HOME=/u01/iaswv904。 NT の場合 : ORACLE_HOME=d:¥oracle¥iaswv904。	0.9.1
Jabber Server（jabberd）	オプション。ユーザー独自の Jabber サーバーをダウンロードしてインストールするには、http://www.jabber.org の Jabber サーバーのインストレーション・ガイドに従います。	1.4.2
Yahoo TransportGateway	オプション。トランスポートのインストレーション・ガイドに従います。http://yahoo-transport.jabberstudio.org を参照してください。	2.0.0
MSN Transport Gateway	オプション。トランスポートのインストレーション・ガイドに従います。http://msn-transport.jabberstudio.org を参照してください。	1.1.0 以上
AOL & ICQ Transport Gateway	オプション。トランスポートのインストレーション・ガイドに従います。http://aim-transport.jabberstudio.org を参照してください。	0.9.25

注意 : 既存の Jabber サーバーにアクセスできる場合は、ユーザー独自の Jabber サーバーをインストールする必要はありません。公開されている Jabber サーバーのリストは、http://www.jabberview.com を参照してください。

メッセージング・サーバーおよび Instant Messaging ドライバの構成

メッセージング・サーバーおよび Instant Messaging ドライバを構成するには、OracleAS Wireless Tools を使用します。

Instant Messaging ドライバを追加または有効化するには、次の手順を実行します。

1. 「管理」タブをクリックします。
2. Wireless サイトに特別な構成の下の「メッセージング・サーバー」セクションを展開します。**「メッセージング・サーバー・ドライバ」**をクリックします。「メッセージング・サーバー・ドライバ」画面が表示されます。
3. 「構成」セクションで**「メッセージング・サーバー・ドライバ」**をクリックします。「メッセージング・サーバー・ドライバ」画面が表示されます。
4. IM ドライバのエントリが表示されない場合は、ステップ 5 に進みます。IM ドライバのエントリが表示されている場合は、次の手順でドライバを有効化します。
   a. IM ドライバのエントリを選択して**「編集」**をクリックします。IM ドライバのプロパティ画面が表示されます。
   b. **「有効」**ボックスを選択し、**「適用」**をクリックします。
   c. ドライバ・インスタンスの構成の項に進みます。
5. **「ドライバの追加」**をクリックします。「ドライバの追加」画面が表示されます。ここに記載されている値のみを入力してください。他の値は不要です。
   a. **「ドライバ名」**フィールドに、「IMDriver」と入力します。
   b. **「デリバリのカテゴリ」**から「IM」を選択します。
   c. **「スピード・レベル」**ドロップダウン・リストから「8」を選択します。
   d. **「コスト・レベル」**ドロップダウン・リストから「1」を選択します。
   e. **「機能」**ドロップダウン・リストから「両方」（あるいは一方向のみのメッセージを希望する場合は「送信」または「受信」）を選択します。
   f. **「メッセージ・キューの数」**フィールドに「1」を入力します。
   g. **「有効」**ボックスを選択します。
   h. **「ドライバ・クラス名」**フィールドに、「oracle.panama.messaging.transport.driver.instantmessaging.InstantMessagingDriver」を入力します。

i. **「1 行追加」**をクリックして、次の各パラメータについて行を追加します。

* Jabber サーバーの場合 : `im.server.host`、`im.server.port`、`im.server.username`、`im.server.password`

  独自の IM ネットワーク（Yahoo、MSN、AOL、ICQ など）のユーザーをサポートするには、次のように各ネットワークのパラメータを追加します（これらのパラメータの追加は必要ですが、ドライバ・インスタンスを構成する時点でのパラメータへの値の入力はオプションです）。

* Yahoo の場合 : `im.yahoo.enable`、`im.yahoo.username`、`im.yahoo.password`
* MSN の場合 : `im.msn.enable`、`im.msn.username`、`im.msn.password`
* AOL の場合 : `im.aol.enable`、`im.aol.username`、`im.aol.password`
* ICQ の場合 : `im.icq.enable`、`im.icq.username`、`im.icq.password`
* 冗長なデバッグ出力を印刷する場合 : `im.debug`
* ドライバが Jabber サーバーから切断するときに使用されるリトライ・パラメータを調節する場合 : `im.retry.limit`、`im.retry.interval`

注意 : これらのパラメータの詳細は、次の表で説明します。

j. **「OK」**をクリックして、ドライバを作成します。

ドライバ・インスタンスの構成

Instant Messaging ドライバ・インスタンスを構成するには、次の手順を実行します。

1. **「Wireless サーバー」**タブをクリックします。「サーバー」画面が表示されます。
2. プロセス表の**「messagingserver1」**をクリックします。messagingserver1 プロセス画面が表示されます。
3. **「ドライバ・インスタンスの追加」**をクリックします。「ドライバ・インスタンスの追加」画面が表示されます。
4. 「ドライバ・インスタンスの追加」画面に次のように入力します。
   a. **「ドライバ・インスタンス名」**フィールドに「IMDriver Instance」と入力します。
   b. **「ドライバ名」**ドロップダウン・リストから IM ドライバを選択します。
   c. **「実行」**をクリックします。

d. **送信スレッドの数**フィールドに、「2」を入力します。

e. **受信スレッドの数**フィールドに、「1」を入力します。

f. **「ドライバ指定パラメータ」**フィールドに、次の値を入力します。

表 10-5 ドライバ指定パラメータの値

パラメータ名	入力する値
Jabber IM 用パラメータ	Jabber IM の場合はこれらの値を入力します。
`im.server.host`	localhost（この値を Jabber サーバーのホスト名に置換します。複数のサーバーを指定するには、カンマ区切りのリストを使用します。1 つのホスト名のみを指定すると、それがすべてのアカウントに使用されます。例 : my1.host.com, my2.host.com）。ファイアウォールの外側に Jabber サーバーがある場合、IM ドライバは、OracleAS Wireless インスタンスに構成されているサイト全体の HTTP プロキシ設定を使用して Jabber サーバーに接続します。
`im.server.port`	5222（デフォルトの Jabber ポート。複数のサーバーを指定するには、対応するポートのカンマ区切りのリストを使用します。例 : 5222, 5222）。
`im.server.username`	oracleagent（この値を OracleAS Wireless インスタンスの Jabber ユーザー名に置換します。IM ドライバは、このユーザー名で Jabber に接続します。Jabber ID の形式は、username@hostname です。このパラメータはユーザー名部分のみが必要です。複数のアカウントを指定するには、ログインする Jabber ユーザー名をカンマ区切りのリストで入力します。前述のパラメータで複数のサーバーを入力した場合は、サーバーと同数のユーザー名（1 つのサーバーに 1 つのユーザー名）を指定する必要があります。前述のパラメータでサーバーを 1 つのみ入力した場合は、ここに指定したすべてのユーザー名がそのサーバーを使用します。例 : oracleagent1, oracleagent2）。Jabber サーバーにアカウントがない場合、IM ドライバは新しいアカウントとして登録を試みます。
`im.server.password`	test（この値を、前述の各ユーザー名に対応するパスワードのカンマ区切りのリストに置換します）。
Yahoo IM 用オプション・パラメータ	Yahoo の IM ネットワークに接続する場合のみ、これらの値を入力します。
`im.yahoo.enable`	Yahoo トランスポートを有効化するには、`im.yahoo.enable` を true に設定します。Yahoo の使用を無効化するには、このフィールドを空白にし、ユーザー名とパスワードのフィールドを無視します。前述のパラメータで複数のアカウントを入力した場合は、対応する値をカンマ区切りのリストで指定します。
`im.yahoo.username`	前述の各ユーザー・アカウントに対して、Yahoo のアカウント ID（これらの ID が Yahoo に登録されている必要があります）をカンマ区切りのリストで入力します（Yahoo IM を使用しない場合はアカウントのエントリを空白のままにします）。有効な Yahoo アカウント情報を入力することによって、Yahoo ユーザーは Yahoo Messenger を介して OracleAS Wireless アプリケーションにアクセスできるようになります。
`im.yahoo.password`	Yahoo アカウントに対応するパスワードをカンマ区切りのリストで入力します。
MSN IM 用オプション・パラメータ	MSN の IM ネットワークに接続する場合のみ、これらの値を入力します。

表 10-5 ドライバ指定パラメータの値（続き）

パラメータ名	入力する値
im.msn.enable	MSN トランスポートを有効化するには、im.msn.enable を true に設定します。MSN の使用を無効化するには、このフィールドを空白にし、ユーザー名とパスワードのフィールドを無視します。前述のパラメータで複数のアカウントを入力した場合は、対応する値をカンマ区切りのリストで指定します。
im.msn.username	前述の各ユーザー・アカウントに対して、MSN のアカウント ID（これらの ID が MSN に登録されている必要があります）をカンマ区切りのリストで入力します（MSN IM を使用しない場合はアカウントのエントリを空白のままにします）。有効な MSN アカウント情報を入力することによって、MSN ユーザーは MSN Messenger を介して OracleAS Wireless アプリケーションにアクセスできるようになります。
im.msn.password	MSN アカウントに対応するパスワードをカンマ区切りのリストで入力します。
AOL IM 用オプション・パラメータ	AOL の IM ネットワークに接続する場合のみ、これらの値を入力します。
im.aol.enable	AOL IM（AIM）トランスポートを有効化するには、im.aol.enable を true に設定します。AOL の使用を無効化するには、このフィールドを単に空白にし、ユーザー名とパスワードのフィールドを無視します。前述のパラメータで複数のアカウントを入力した場合は、対応する値をカンマ区切りのリストで指定します。
im.aol.username	前述の各ユーザー・アカウントに対して、AOL のアカウント ID（これらの ID が AOL に登録されている必要があります）をカンマ区切りのリストで入力します（AOL IM を使用しない場合はアカウントのエントリを空白のままにします）。有効な AOL アカウント情報を入力することによって、AOL ユーザーは AOL Instant Messenger を介して OracleAS Wireless アプリケーションにアクセスできるようになります。
im.aol.password	AOL アカウントに対応するパスワードをカンマ区切りのリストで入力します。
ICQ のオプション・パラメータ	ICQ の IM ネットワークに接続する場合のみ、これらの値を入力します。
im.icq.enable	ICQ トランスポートを有効化するには、im.icq.enable を true に設定します。ICQ の使用を無効化するには、このフィールドを空白にし、ユーザー名とパスワードのフィールドを無視します。前述のパラメータで複数のアカウントを入力した場合は、対応する値をカンマ区切りのリストで指定します。
im.icq.username	前述の各ユーザー・アカウントに対して、ICQ のアカウント ID（これらの ID が ICQ に登録されている必要があります）をカンマ区切りのリストで入力します（ICQ IM を使用しない場合はアカウントのエントリを空白のままにします）。有効な ICQ アカウント情報を入力することによって、ICQ ユーザーは ICQ Instant Messenger を介して OracleAS Wireless アプリケーションにアクセスできるようになります。
im.icq.password	ICQ アカウントに対応するパスワードをカンマ区切りのリストで入力します。
その他のパラメータ	その他の設定のために次の値を入力します。

表 10-5 ドライバ指定パラメータの値（続き）

パラメータ名	入力する値
im.debug	冗長なデバッグ出力を有効化するには、この値を true に設定します。これによって、ドライバが追加の通知メッセージを出力するようになります。ただし、通知メッセージを出力するには、OracleAS Wireless インスタンスのシステム・ログ構成で、「通知」ログ・レベルが使用可能に設定されている必要があります。
im.retry.limit	20（Jabber サーバーから切断された場合にドライバが再接続を試行する回数。デフォルトの制限は 20 です）。
im.retry.interval	20（再接続を試行する秒単位の時間間隔。デフォルトの間隔は 20 秒です）。

g. **「OK」**をクリックして、ドライバ・インスタンスを作成します。

注意： ドライバ固有の属性を変更するたびに、メッセージング・サーバーを再起動する必要があります。

IM ドライバと非同期サーバーの併用 Instant Messaging を介して OracleAS Wireless の非同期可能アプリケーションへのアクセスを提供するために、IM ドライバを非同期サーバーとともに使用できます。IM ドライバを非同期サーバーと併用するには、非同期サーバーでアクセス・ポイントを設定する必要があります。

非同期サーバーを構成するには、次の手順を実行します。

1. **「Wireless サーバー」**タブをクリックします。
2. **「管理」**タブをクリックします。
3. Wireless サイトに特別な構成の下の「非同期サーバー」セクションを展開します。**「アクセス・ポイント」**をクリックします。「アクセス・ポイント」画面が表示されます。
4. **「アクセス・ポイントの追加」**をクリックします。
5. **「名前」**フィールドに、IM のエントリ・ポイントを入力します。
6. **「デリバリ・タイプ」**から「IM」を選択します。
7. 「アドレス」フィールドに、*jabber*|<*Jabber ID*> を入力します。<Jabber ID> は IM ドライバが使用する ID（たとえば、oracleagent@localhost）です。つまり、アドレス全体では <im.server.username>@<im.server.host> の形式になります。
8. **すべてのサービスへのアクセスを許可**ボックスを選択します。
9. **「OK」**をクリックして、アクセス・ポイントを追加します。

10. Yahoo または MSN など、他の IM ネットワーク用の IM ドライバを設定している場合は、該当する各ネットワークのアクセス・ポイントを特別に追加する必要があります。アドレスは *<network-name>*|*<userid>* の形式で指定する必要があります。<network-name> には、yahoo、msn、aim または icq を指定します。たとえば、MSN 用の IM ドライバを設定した場合は、*msn*|*<im.msn.username>* のアドレスを使用してアクセス・ポイントを追加する必要があります。<im.msn.username> 部分には、IM ドライバ・インスタンスに設定したパラメータの値を指定します。同様に、AOL IM（AIM）用の IM ドライバを設定した場合は、*aim*|*<im.aol.username>* のアドレスを使用してアクセス・ポイントを追加します。

これらの手順で、非同期サーバーの構成は完了です。非同期可能アプリケーションを起動するには、次のようにします。

- IM ドライバとして構成した IM システム（Jabber、Yahoo、MSN、AOL、ICQ など）に従って、対応するユーザーを IM クライアントの名簿または連絡先リストに追加する必要があります。このユーザーは IM エージェントと呼ばれます。

  例 : IM ドライバには、im.msn.username の値で msnimdemo@oracle.com を使用して、MSN トランスポートがすでに構成されています。ここでは、MSN Messenger クライアントを起動して、ユーザー自身で（自分の MSN アカウントを使用して）ログインし、MSN の IM エージェント msnimdemo@oracle.com をユーザーの連絡先リストに追加します。

- メッセージの本文に「help」と入力したインスタント・メッセージを IM エージェントに送信します。このメッセージを送信すると、起動できるアプリケーションのリストが記載されたインスタント・メッセージが返信されます。

  例 : hello のテキストを入力したメッセージを IM エージェントに送信し、hello アプリケーションを起動します。エージェントから Hello World レスポンスが戻ります。

  注意 : 前述の例では、hello アプリケーションを非同期可能アプリケーションと仮定しています。

- ユーザーの入力を必要とするアプリケーションの場合は、エージェントとチャットして必要な値を入力します。

  例 : sm を入力して、Short Messaging アプリケーションを起動します。「0 メニュー、1 [] タイプ」などのオプション・メニューが返信されます。「1」を入力して「タイプ」を選択します。選択するタイプのリストが返信されます。「3」を入力して「ボイス」を選択します。同様の手順で「受信者」、「件名」および「本文」など、他のフィールドを入力し、最終的にメッセージを送信できます。

XMS サービスの使用　XMS サービスを使用してインスタント・メッセージを送信するために、XMSSimpleSender と XMSSender の 2 つの API オプションがあります。

- XMSSimpleSender の使用

  受信者のアドレスを指定するには、IM: タグを IM アドレスに付加します。つまり、次の書式を使用します。

  ```
  IM:<address>
  ```

  <address> には、受信者の IM アドレスを指定します。

  <address> には、次の書式も指定できます。

  ```
  <network-name>|<userid>,
  ```

  <network-name> には、IM ネットワーク名（たとえば、jabber、yahoo、msn、aim、icq）を指定します。

  <userid> には、そのネットワークのユーザー ID を指定します。

  注意：　フィールド・セパレータには、パイプ（|）文字を使用します。

例を次に示します。

表 10-6 異なる IM アドレスのアドレス書式

IM アドレス	XMSSimpleSender のアドレス書式
ID が foo@jabber.org の Jabber ユーザー	IM:jabber|foo@jabber.org
ID が foo@msn.com の MSN ユーザー	IM:msn|foo@msn.com
ID が foo の Yahoo ユーザー	IM:yahoo|foo
ID が foo の AOL IM ユーザー	IM:aim|foo
ID が 12345 の ICQ IM ユーザー	IM:icq|12345

- XMSSender の使用

  受信者のアドレスを指定するには、IM: タグなしで、前述のアドレス書式を使用します（後述の例を参照してください）。IMAddressData クラスのインスタンスとしてアドレスを作成し、「IM」トランスポート・タイプを使用します。これとは別に、XMS クライアントをユーザーの通常のクライアントとして使用します。XMS の例は、10-60 ページの「XMS ドライバ」を参照してください。

IMAddressData のインスタンスを作成するには、2 種類のコンストラクタから 1 つを使用できます。

```
IMAddressData(String address)
```

address には、前述されている XMSSimpleSender アドレス書式の <*address*> が入ります。

例 :

```
IMAddressData("msn|foo@msn.com")
IMAddressData("aim|foo")
```

```
IMAddressData(String userid, String network)
```

userid には、<userid> が入ります。

network には、<network-name> が入ります。

例 :

```
IMAddressData("foo@msn.com", "msn")
IMAddressData("foo", "aim")
IMAddressData("foo@jabber.org", "jabber")
```

次の例は、両方のコンストラクタを使用して IM アドレスの指定方法を示すサンプル XMS クライアント・コードです。

```
// IM recipients
AddressData imRecipients[] = new AddressData[4];

/* using the first constructor */
// specify a regular Jabber user
imRecipients[0] = new IMAddressData("jabber|foo@jabber.org");
// specify a Yahoo user (by prepending the "yahoo|" tag)
imRecipients[1] = new IMAddressData("yahoo|foo");

/* using the second constructor */
// specify an MSN user
imRecipients[2] = new IMAddressData("foo@msn.com", "msn");
// specify an AOL IM user
imRecipients[3] = new IMAddressData("foo", "aim");

// Packet object
Packet pkt = new Packet();
AddressData imSender = new IMAddressData("jabber|oracle@jabber.org");
pkt.setFrom(TransportType.IM, imSender);
pkt.addRecipients(TransportType.IM, imRecipients);
```

IM ドライバのスタンドアロン・テスト IM ドライバをテストするには、ローカル・マシンでいくつかの追加手順を実行する必要があります。具体的には、Jabber サーバーおよび Jabber クライアントをインストールします。

> **注意：** 次の説明は、Jabber クライアントからテストする場合を意図しています。Yahoo または AOL など、独自のネットワークからテストする場合は、Jabber サーバーに適切なトランスポートを追加して、適切な IM クライアント（Yahoo Messenger または AOL Instant Messenger など）をインストールする必要があります。

Windows で Jabber サーバーをインストールするには、次の手順を実行します。

1. http://jabberd.jabberstudio.org/downloads/JabberD-1.4.2.exe から JabberD をダウンロードします。
2. インストール処理を開始するファイルをオープンします。デフォルト値を使用してインストールを進めて、インストールを終了します。
3. Windows の「スタート」→「プログラム」→「JabberD」→「JabberD」を選択して Jabber サーバーを起動します。DOS の「コマンド プロンプト」ウィンドウがオープンし、`[notice] (-internal): initializing server` を示すログ出力が表示されます。このメッセージが表示された場合は、Jabber サーバーが正常に起動しています。

UNIX で Jabber サーバーをインストールするには、次の手順を実行します。

1. Jabber サーバーをダウンロードします。

   Solaris 2.6 の場合 :
   `http://jabberd.jabberstudio.org/downloads/jabber-1.4.2a-sparc-solaris-2.6.tar.gz`

   Solaris 7 の場合 :
   `http://jabberd.jabberstudio.org/downloads/jabber-1.4.2a-sparc-solaris-7.tar.gz`

2. tar ファイルを抽出します。

   ```
   # mkdir jabber
   # cd jabber
   # gtar xzvf jabber-1.4.2a-sparc-solaris-<ver>.tar.gz
   ```

3. jabberd を実行します。

   ```
   # jabberd/jabberd
   ```

4. Jabber サーバーが起動され、`[notice] (-internal): initializing server` などのログ・メッセージの出力が開始されます。このメッセージが表示された場合は、Jabber サーバーが正常に起動しています。

Windows で Jabber クライアントをインストールするには、次の手順を実行します。

1. `http://www.jabber.org/user/clientlist.php` から該当する Jabber クライアントをダウンロードし、インストールします。Jabber 初心者には Rival（http://rival.chote.net/）をお薦めします。
2. インストール終了後、Jabber サーバーに接続するクライアントをローカルホストに構成します。ユーザー自身の新規 Jabber アカウントを作成します。次に、IM ドライバに設定した OracleAS Wireless インスタンスの Jabber ID を連絡先リストに追加します。たとえば、IM ドライバに事前構成済のデフォルトの設定を使用する場合は、oracleagent@localhost の ID を連絡先リストに追加します。

UNIX で Jabber クライアントをインストールするには、次の手順を実行します。

1. `http://www.jabber.org/user/clientlist.php?Platform=Linux` から該当する Jabber クライアントをダウンロードし、コンパイルしてインストールします。すべてのクライアントが UNIX でコンパイルできるとはかぎりません。各自の責任で試行してください。`http://prdownloads.sourceforge.net/gabber/gabber0.8.7_solaris.tar.gz` に、Gabber（一般に普及している Linux Jabber クライアント）のプリコンパイル済 UNIX バイナリがあります。
2. クライアントを起動した後は、前述の Windows の項のステップ 2 に従います。

テスト・メッセージを OracleAS Wireless に送信するには、次の手順を実行します。

1. Jabber クライアントから、先ほど追加した連絡先（oracleagent@localhost）をダブルクリックしてインスタント・メッセージを送信します。
2. 本文に「help」と入力して、メッセージを送信します。IM ドライバおよび非同期アクセス・ポイントが正しく構成されている場合は、起動できる非同期アプリケーションをリストした OracleAS Wireless からの即時レスポンスを受け取ります。

AOL Instant Messenger 用 IM ドライバのクイック構成 AOL Instant Messenger 用の IM ドライバをクイック構成するには、次の手順を実行します。

1. AOL スクリーンネームを作成します。

   ユーザー自身用（ここでは *<my_screenname>*）と IM ドライバ用（ここでは *<imdriver_screenname>*）の 2 つの AOL スクリーンネーム（アカウント）が必要です。スクリーンネームが 2 つない場合は、http://www.aim.com でスクリーンネームを作成します。

   注意： AOL の Web サイトでは、指定の電子メール・アドレスまたは IP アドレスに対して作成できるアカウントの数に制限があります。アカウントを作成しようとしてエラーが発生した場合は、別の電子メール・アドレスまたは異なるマシンを使用してください。

2. IM ドライバと非同期アクセス・ポイントを構成します。

   a. messagingserver1 プロセスに IM ドライバのインスタンスを追加します。次の値を入力します。

```
im.server.host = myjabber.net (or any Jabber host that has an AOL IM (AIM)
transport)
im.server.username = <imdriver_screenname>
im.server.password = <password of imdriver_screenname> (or whatever you
want)
im.aol.enable = true
im.aol.username = <imdriver_screenname>
im.aol.password = <password of imdriver_screenname>
```

   b. 「Wireless Tools」→「サイト管理」→「コンポーネント構成」→「アクセス・ポイント」を選択します。表示された画面で、IM アクセス・ポイントを追加します。

      * 「名前」に IM エントリを入力します。
      * 「デリバリ・タイプ」に「IM」を選択します。
      * 「アドレス」に *aim|<imdriver_screenname>* を入力します。
      * **「すべてのアプリケーションへのアクセスを許可」**を選択します。
      * messagingserver1 プロセスを再起動します。

3. AOL Instant Messenger（AIM）をインストールします。

   a. http://aim.aol.com/aimnew/NS/congratsd2.adp から AIM をダウンロードし、インストールします。

   b. AIM を起動し、**「Setup」**をクリックします。HTTP プロキシ設定（`host = www-proxy.us.oracle.com, port = 80, protocol = HTTP`）を構成するために、**「Connection」**をクリックします。

   c. ユーザーのスクリーンネーム *<my_screenname>* を使用して、AIM にサイン・オンします。

   d. *<imdriver_screenname>* をメンバー・リストに追加して、チャットを開始します。

XMS ドライバ

このドライバはホスティングされている XMS サービスを使用し、デフォルトでオラクル社がホスティングしているデモ用サービスを使用します。このドライバは、インターネット上でホスティングされている OracleAS Wireless サーバーの XMS クライアントとして機能し、OracleAS Wireless の XMS Web サービスをサポートするサービスを指すように構成できます。XMS ドライバは、HTTP を介した SOAP を使用します（OracleAS Wireless の XMS Web サービス）。オラクル社がホスティングしている XMS サーバーでは、アクセス用のアカウントは不要です。XMS は、HTTP を介した SOAP を使用します。試用目的で、ユーザー名に "" （一対の二重引用符）、パスワードに "" を使用して、ホスティングされているデモ用

OracleAS Wireless インスタンスを使用できます。ただし、メッセージ送信のみに用途が制限されています。その際の URL は、http://messenger.us.oracle.com/xms/webservices です。

クラス名

`oracle.panama.messaging.transport.driver.push.PushDriver`

構成

- `messaginggatewayURL`

  ホスティングされている XMS Web サービスへの URL。このパラメータは必須です。例 : `http://messenger.oracle.com/xms/webservices`

- `username`

  XMS サービスに対する認証に使用する名前。XMS Web サービスでは、ユーザー名とパスワードが必要かどうかを判別できます。XMS Web サービスにユーザー名が必要でない場合は、空白（空の文字列）にしてください。ユーザー名が存在しない場合や、そのユーザー名のパスワードが正しくない場合は、XMS Web サービスから「`Bad username and password`」が戻されます。

  例 :`messaginguser`

- `password`

  username フィールドに指定したユーザーのパスワード。XMS サービスに対する認証に使用されます。XMS Web サービスでは、ユーザー名とパスワードが必要かどうかを判別できます。ユーザー名とパスワードが必要でない場合は、password を空白（空の文字列）にしてください。ユーザー名が存在しない場合や、そのユーザー名のパスワードが正しくない場合は、XMS Web サービスから「`Bad username and password`」が戻されます。

  例 :`8Uh42g`

XMS ドライバは、すべてのコンテンツ・タイプを処理できます。処理方法は、ホスティングされている XMS Web サービスによって決まります。ドライバは HTTP 接続で動作します。XMS ドライバは OracleAS Wireless のプロキシ設定を継承するため、明示的な HTTP プロキシ設定は不要です。

このドライバは送信のみを実行します。ホスティングされている XMS サービスと同数のトランスポート・タイプがサポートされます。実際にサポートされるタイプは、どのホスティング環境（OracleAS Wireless インスタンス）のサービスを実行しているかに応じて異なります。OracleAS Wireless のサービスでは、インスタンスでサポートされている正確なトランスポートを記述する API がサポートされています。

電子メール・ドライバ

電子メール・ドライバでは、メッセージの配信用に SMTP、メッセージの受信用に IMAP または POP3 がサポートされます。このドライバでは、メッセージの送信と受信を処理できます。メッセージの受信用に、IMAP プロトコルと POP3 プロトコルの両方がサポートされます。

クラス名 `oracle.panama.messaging.transport.driver.email.EmailDriver`

構成

- `server.incoming.protocol`

  これは電子メール受信プロトコル用の値です。可能な値は IMAP と POP3 です。ドライバ・インスタンスで電子メールの受信がサポートされている場合にのみ必要です。

- `server.incoming.host`

  受信メール・サーバーのホスト名。ドライバ・インスタンスで電子メールの受信がサポートされている場合にのみ必要です。

- `server.incoming.receivefolder`

  ドライバによるメッセージのポーリング元となるフォルダの名前。デフォルト値は INBOX です。

- `server.incoming.usernames`

  ドライバ・インスタンスのポーリング元であるメール・アカウントのユーザー名のリスト。名前はそれぞれ foo,bar など、カンマで区切る必要があります。ドライバ・インスタンスで電子メールの受信がサポートされている場合にのみ必要です。

- `server.incoming.passwords`

  前述のユーザー名に対応するパスワードのリスト。各パスワードはカンマで区切り、ユーザー名のリストに対応するユーザー名と同じ順序で指定する必要があります。ドライバ・インスタンスで電子メールの受信がサポートされている場合にのみ必要です。

  例 : `foopwd,barpwd`

- `server.incoming.emails`

  前述のユーザー名に対応する電子メール・アドレスのリスト。各電子メール・アドレスはカンマで区切り、ユーザー名のリストに対応するユーザー名と同じ順序で指定する必要があります。ドライバ・インスタンスで電子メールの受信がサポートされている場合にのみ必要です。

  例 : `foo@oracle.com,bar@oracle.com`

- `server.incoming.checkmailfreq`

  メール・サーバーからメッセージを取り出す頻度。値は秒単位で、デフォルト値は 5 秒です。

- `server.incoming.deletefreq`

  削除されたメッセージを永続的に削除する頻度。値は秒単位で、デフォルト値は 300 秒です。負の値は、メッセージを削除しないことを示します。POP3 プロトコルの場合、メッセージは処理された後に削除されます。

- `server.incoming.autodelete`

  この値は、ドライバが処理済のメッセージに削除のマークを付ける必要があるかどうかを示します。可能な値は true または false で、デフォルト値は false です。POP3 プロトコルの場合、メッセージは常に処理の直後に削除されます。

- `server.outgoing.host`

  SMTP サーバー名。電子メールを送信する必要がある場合にのみ必要です。

  例 :`smtp05.oracle.com`

- `default.outgoing.from.address`

  送信メッセージに指定されていない場合のデフォルトの FROM アドレス。

電子メール・ドライバでは、メッセージの配信用に SMTP、メッセージの受信用に IMAP4 または POP3 がサポートされます。このドライバでは、メッセージの送信と受信を処理できます。メッセージの受信用に、IMAP4 プロトコルと POP3 プロトコルの両方がサポートされます。

注意： ドライバを送信専用にする場合は、`server.outgoing.host` のみを構成してください。

音声ドライバ

音声ドライバでは、VoiceGenie がサポートしている Out Bound Call プロトコルがサポートされます。現在は、音声ゲートウェイでの動作テストのみが完了しています。このドライバではメッセージの送信のみが処理されます。このドライバはメッセージの送信のみを実行できますが、ドライバを動作させるには送信機能と受信機能の両方を使用するように構成する必要があります。

このドライバでは、次のコンテンツ・タイプを処理できます。

- `text/plain`
- `ContentTypes.MOBILE_XML_URL`
- `ContentTypes.MOBILE_XML_URL_REMOTE`
- `ContentTypes.MOBILE_XML`
- `ContentTypes.URL`（URL リソースのコンテンツ・タイプが前述のいずれかの場合のみ）

他のコンテンツ・タイプの場合、ドライバは致命的でないドライバ例外をスローします。

クラス名 `oracle.panama.messaging.transport.driver.voice.VoiceGenieDriver`

構成

- `voicegenie.outbound.servlet.uri`

  VoiceGenie Outbound Call Servlet の URL。このパラメータは必須で、デフォルト値はありません。サンプルを次に示します。

  ```
  http://rossini.us.oracle.com/servlet/com.voicegenie.outboundcallservlet.OutboundCallServlet
  ```

  ドライバは、サイト・レベルのプロキシ構成を使用して、この URL にアクセスします。

- `voicegenie.outbound.servlet.username`

  VoiceGenie Outbound Call のユーザー名。

- `voicegenie.outbound.servlet.password`

  VoiceGenie Outbound Call のパスワード。

- `voicegenie.outbound.servlet.dnis`

  コール元として設定する電話番号。この値はオプションです。デフォルト値は 12345678 です。

- `voicegenie.urlservice.path`

  ビルトイン VoiceGenie サービスへのサービスパス。このドライバは、HTTP アダプタに準拠する OracleAS Wireless サービスに依存します。OracleAS Wireless インストールには、デフォルトで、この音声ドライバをサポートするための HTTP アダプタ・サービス VoiceGenieURLService が用意されています。これは必須パラメータです。デフォルト値はなく、特定の OracleAS Wireless インストールを調べて値を取得する必要があります。サンプルを次に示します。

  ```
  foo.oracle.com:9000/ptg/rm?PAservicepath=/VoiceGenieURLService&PAsubmit=Submit
  ```

- `voicegenie.driver.receive.host` と `voicegenie.driver.receive.port`

  この 2 つは、HTTP アダプタに Mobile XML フォーマットでコンテンツを送信させるための IP ホストとポートです。ポートは、このドライバ専用にする必要があります。この 2 つのパラメータは必須です。

注意： このドライバは、ポート（voicegenie.driver.receive.port で指定）をオープンして、HTTP 通信量をリスニングします。voicegenie.driver.receive.host と voicegeneie.driver.receive.port を使用して、ドライバを接続する OracleAS Wireless HTTP アダプタの URL が構成されます。OracleAS Wireless の XML を含むメッセージを送信する場合、このパラメータは必須です。ドライバを機能させるには、適切なホスト名と一意のポート番号を指定してください。

UCP ドライバ

UCP（Universal Communication Protocol）は、最も普及している GSM SMS プロトコルの 1 つです。OracleAS Wireless には、送受信機能を提供する UCP ドライバのビルトイン実装が用意されています。

UCP ドライバは EMI/UCP 4.0 を実装します。リスニング・モードと非リスニング・モードの両方をサポートしています。非リスニング・モードでは、ドライバが SMSC への接続を 1 つオープンします。送信と受信でこの接続を共有します。UCP ドライバは、ほどんどの SMSC を非リスニング・モードで処理します。リスニング・モードでは、ドライバはポートをリスニングします。SMSC がドライバへの接続をオープンして、デバイスから受信したメッセージをドライバに配信します。ドライバは、ターゲット・デバイスに送信するために、必要に応じて HTTP 接続をオープンし、SMSC にメッセージを配信します。HTTP 接続は、可能な場合は再使用されます。

このドライバでは、次のコンテンツ・タイプを処理できます。

- text/plain
- ContentTypes.RING_TONE
- ContentTypes.GRAPHICS
- ContentTypes.WAP_SETTINGS
- ContentTypes.WAP_PUSH
- ContentTypes.VCARD
- ContentTypes.EMAIL_SETTING
- ContentTypes.VCALENDAR
- ContentTypes.MM1_NOTIFICATION
- ContentTypes.EMS

- ContentTypes.ENCODED_SMS
- ContentTypes.URL（関連するリソースのコンテンツ・タイプが前述のいずれかの場合のみ）

他のコンテンツ・タイプの場合、ドライバは致命的でないドライバ例外をスローします。

クラス名

oracle.panama.messaging.transport.driver.sms.UCPDriver

構成

- sms.account.id

  SMSSC のアカウント ID です。通常は、オペレータが割り当てる短縮番号を使用する必要があります。このパラメータは必須です。

- sms.account.password

  オペレータが割り当てるパスワードです。このパスワードは、UCP コマンド 60 で SMSC へのセッションをオープンするために使用されます。

- sms.ucptype

  メッセージの送信に使用するコマンドを指定します。可能な値は 01 と 51 です。デフォルト値は 51 で、メッセージの送信に UCP コマンド 51 を使用することを意味します。

- sms.server.host and sms.server.port

  ドライバが TCP/IP 接続をオープンするために使用する SMSC サーバー情報。

- sms.server.default.encoding

  テキスト・メッセージのデフォルトのコード体系。デフォルト値は IA5 です。

- sms.local.port

  ドライバが送信用の接続を確立するために使用するローカル・ポート。非リスニング・モードのドライバに適用可能です。

- sms.local.address

  ドライバを実行する論理ホスト名または IP アドレス。非リスニング・モードのドライバに適用可能です。

- sms.window.size

  ウィンドウイング・サイズ。非リスニング・モードのドライバにのみ適用可能です。

- sms.receiver.listener.port

  ドライバがリスニング・モードになっている場合は、このポートが SMSC で使用されて TCP/IP 接続が初期化され、受信したメッセージがドライバに渡されます。これは、sms.server.url を指定した場合に使用されます。それ以外の場合は無視されます。

- `sms.server.url`

  ドライバから SMSC にアクセスして HTTP 接続を介してメッセージを送信するための URL です。このパラメータを指定すると、`sms.server.host` と `sms.server.port` は無視されます。このパラメータを指定しない場合は、`sms.server.host` と `sms.server.port` が必要です。

- `sms.bulk.sending`

  バルク送信（可能な場合）の実行にコマンド 02 を使用するかどうかを決定します。デフォルト値は true です。

- `sms.message.maxchunks`

  1 つのメッセージの最大許容チャンク数です。この数値を超えた後のチャンクは無視されます。デフォルトは -1 で、無制限を意味します。

- `sms.message.chunksize`

  各チャンクの最大バイト数です。デフォルト値は 160 です。

  注意：

  - SMSC へのダイレクト TCP 接続を使用する場合、ドライバではコマンド 60 を使用して SMSC とのセッションが開始されます。これにより、ドライバと SMSC は 1 つのソケット接続で通信し、ステータスを送受信できます。この場合、`sms.server.url` は使用されません。
  - SMSC への接続が HTTP ベースの場合は、`sms.server.url` の値を指定する必要があります。これは、ドライバ・インスタンスでメッセージの送信に使用される URL です。また、ドライバ・インスタンスが受信メッセージ用にこのポートへのバインドをオープンできるように、`sms.receiver.listener.port` も指定する必要があります。HTTP 接続の場合、`sms.server.host` と `sms.server.port` は使用されません。
  - `sms.message.chunksize` では、メッセージの合計サイズが 1 つの SMS メッセージより大きい場合に、各メッセージのサイズが制御されます。`sms.message.maxchunks` では、各メッセージの最大許容チャンク数が制御されます。最大許容数を超えた後のチャンクは破棄されます。

- `sms.alert.interval`

  コマンド 31 が起動される間隔（秒単位）。負の値を設定すると、コマンド 31 は起動されません。このコマンドは、ドライバが非リスニング・モードの場合、接続状態を保持するための SMSC へのシグナルとして主に使用されます。

SMPP ドライバ

SMPP（Short Message Peer-to-Peer）は、最も普及している GSM SMS プロトコルの 1 つです。OracleAS Wireless には、送受信機能を提供する SMPP ドライバのビルトイン実装が用意されています。ドライバは、送信機能が使用可能な場合は、送信装置として SMSC への TCP 接続を 1 つオープンします。受信機能が使用可能な場合は、受信装置として SMSC への接続をもう 1 つオープンします。したがって、送受信両方の機能を使用できる場合は、ドライバと SMSC 間のすべての通信には 2 つの接続（ドライバによって開始された）が必要です。このドライバは SMPP 3.4 を実装します。したがって、SMPP 3.4 をサポートする SMSC のみをサポートします。

このドライバでは、次のコンテンツ・タイプを処理できます。

- `text/plain`
- `ContentTypes.RING_TONE`
- `ContentTypes.GRAPHICS`
- `ContentTypes.WAP_SETTINGS`
- `ContentTypes.WAP_PUSH`
- `ContentTypes.VCARD`
- `ContentTypes.EMAIL_SETTING`
- `ContentTypes.VCALENDAR`
- `ContentTypes.MM1_NOTIFICATION`
- `ContentTypes.EMS`
- `ContentTypes.ENCODED_SMS`
- `ContentTypes.URL`（関連するリソースのコンテンツ・タイプが前述のいずれかの場合のみ）

他のコンテンツ・タイプの場合、ドライバは致命的でないドライバ例外をスローします。

クラス名

`oracle.panama.messaging.transport.driver.sms.SMPPDriver`

SMPP ドライバ : 構成

- `sms.account.id`

  SMSSC のアカウント ID です。通常は、オペレータが割り当てる短縮番号を使用する必要があります。このパラメータは必須です。

- `sms.server.host`

  ドライバが TCP/IP 接続をオープンするために使用する SMSC サーバー情報。

- `sms.smpp.transmitter.system.id`
  `sms.smpp.transmitter.system.type`
  `sms.smpp.transmitter.system.password`
  `sms.server.transmitter.port`

  これらの属性は SMSC に依存します。送信装置として（つまり、SMS を送信するために）SMSC にログインできるように、オペレータは、短縮番号の割当てとともにシステム ID、タイプ、パスワードおよびポートも指定できます。

- `sms.smpp.receiver.system.id`
  `sms.smpp.receiver.system.type`
  `sms.smpp.receiver.system.password`
  `sms.server.receiver.port`

  これらの属性は SMSC に依存します。受信装置として（つまり、SMS を受信するために）SMSC にログインできるように、オペレータは、システム ID、タイプ、パスワードおよびポートを指定できます。

- `sms.server.default.encoding`

  テキスト・メッセージのデフォルトのコード体系。デフォルト値は IA5 です。

- `sms.local-sending.port`

  ドライバが送信用の接続を確立するために使用するローカル・ポート。

- `sms.local-receiving.port`

  ドライバが受信用の接続を確立するために使用するローカル・ポート。

- `sms.local.address`

  ドライバを実行している論理ホスト名または IP アドレス。

- `sms.server.source.ton`

  送信者アドレス（アカウント ID）の TON。

- `sms.server.source.npi`

  送信者アドレス（アカウント ID）の NPI。

- `sms.server.destination.ton`

  宛先アドレスの TON。

- `sms.server.destination.npi`

  宛先アドレスの NPI。

- `sms.window.size`

  ウィンドウイング・サイズ。デフォルト値は 1 です。

- `sms.bulk.sending`

  バルク送信（可能な場合）の実行にコマンド `submit_multi` を使用するかどうかを決定します。デフォルト値は true です。

- `sms.payload.sending`

  `short_message` フィールドを使用できる場合に、`message_payload` フィールドを使用して送信するかどうかを決定します。デフォルト値は false です。

- `sms.message.maxchunks`

  長いメッセージを分割できる最大チャンク数。この数値を超えた後のチャンクは無視されます。負の値は無制限を意味します。デフォルト値は -1 です。

- `sms.message.chunksize`

  各チャンクの最大サイズ（バイト単位）。デフォルト値は 160 です。

- `sms.enquire-link.interval`

  照会リンクがコールされる間隔（秒単位）。間隔が 0（ゼロ）未満の場合、この機能は無効です。デフォルト値は -1 です。

- `sms.throttling.delay`

  SMSC からスロットル・エラーを受け取った後、再開を送信するまでの遅延秒数。遅延が 0（ゼロ）未満の場合、この機能は無効です。デフォルト値は 15 です。

- `sms.extra.error-code`

  メッセージの再送信を必要とする SMSC が送信できるカンマ区切りのエラー・コードのリスト（たとえば、0x45, 0x50）。デフォルトでは、エラー・コードをステータスとして受信した場合、メッセージは廃棄されます。

- `sms.bind.retry.delay`

  SMSC でバインドを再度試行するまでに、ドライバが照会リンクのレスポンスを待機する遅延秒数。このパラメータまたは `sms.enquire-link.interval` パラメータが 0（ゼロ）未満の場合、この機能は無効です。このパラメータのデフォルト値は -1 です。

- `sms.registered.delivery.mark`

  アプリケーションの要件に基づいて SMSC にメッセージを送信する場合、ドライバは登録済配信フラグを設定できます。ただし、SMSC が有効なフラグすべてをサポートしているとはかぎりません。このような場合に、送信リクエストが SMSC によって拒否されないように、このパラメータを使用してフラグの特定のビットを使用不可にできます。

FAX ドライバ（RightFax）

このドライバは、FAX メッセージと RightFax（Captaris 提供）FAX プロトコルをサポートします。このドライバは、RightFax ソフトウェア・パッケージと、FAX メッセージの配信に RightFax FAX サーバーを使用できるかどうかに依存します。このドライバにはメッセージ送信機能しかありません。

このドライバでは、すべてのコンテンツ・タイプを処理できます。次の MIME タイプが認識されます。

- `text/plain`
- `application/msword`
- `application/msexcel`
- `application/msppt`
- `application/postscript`
- `application/octet-stream`

`ContentTypes.URL` の場合、ドライバは指定された URL からコンテンツを取り出します。この操作で戻されるコンテンツと MIME タイプが、FAX サーバーに送信されます。

クラス名 `oracle.panama.messaging.transport.driver.fax.RightFAXDriver`

構成

- `server.url`

  RightFax サーバーへの URL。このパラメータは必須です。

- `server.account`

  RightFax サーバーへのアカウント名。このパラメータは必須です。

次のデフォルト属性はすべてオプションです。これらの属性は、カバー・シートのカスタマイズにのみ使用されます。

- `server.coverpage`

  このドライバが使用する表紙。未指定の場合は、FAX ゲートウェイのデフォルトの表紙が使用されます。

- `default.sender.name`

  表紙に表示するデフォルトの送信者名。

- `default.sender.corporation`

  表紙に表示するデフォルトの送信元会社名。

- `default.sender.fax`

  表紙に表示するデフォルトの送信元 FAX 番号。

- `default.sender.phone`

  表紙に表示するデフォルトの送信元電話番号。

- `default.sender.address`

  表紙に表示するデフォルトの送信元住所。

- `default.sender.notes`

  表紙に表示するデフォルトの送信元注記。

- `server.coverpage`

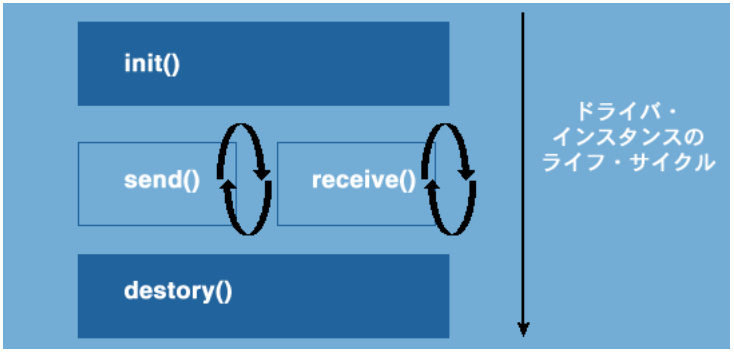
  RightFAX ドライバには、表紙をレンダリングするための表紙設定が必要です。表紙は、RightFAX サーバーの表紙ディレクトリにあります。このパラメータには、希望する表紙のファイル名を設定します。このパラメータが未設定の場合、FAX サーバーのデフォルトの表紙が使用されます。

新規ドライバの開発方法

ドライバ・インタフェースは、特定のプロトコル用のドライバを実装することを意図しています。前述のように、ドライバを簡単にトランスポート・システムにプラグインして、ネットワーク・プロトコルのサポートをベース製品へと拡張できます。ドライバは、きわめて薄いレイヤーであることが予期されており、プロトコル固有の詳細のみを処理します。ドライバの設計は、ライフ・サイクル、ロード・バランシングまたはスケーラビリティの問題を処理することを目的としていません。これらの問題はトランスポート・システムで処理されます。

トランスポート・システムは、ドライバ・インタフェースに指定された `init()` および `destroy()` メソッドをコールして、それぞれドライバ・インスタンスの初期化と破棄を実行します。また、ロード・バランシングと並行性も処理します。ドライバでは特定のプロトコルのセマンティクスの解析にのみ重点を置き、他はすべてトランスポート・システムに任せる必要があります。

図 10-12 ドライバのライフ・サイクル



ドライバは、送信専用、受信専用または送受信用に指定できます。送信のセマンティクスを実装するには、ドライバは単にインタフェースするドライバに指定されている `send()` メソッドを実装します。受信の場合は少し複雑で、受信アクションはトランスポート・システムにより駆動されます。受信機能を実装するために、ドライバはインタフェースするドライバに指定されている `receive()` メソッドに受信論理を挿入します。トランスポート・システムは、ドライバ・インスタンスのライフ・サイクルを通じて `receive()` メソッドを継続的に起動します。

ドライバでは、インスタンスをスレッド・セーフにするように設計する必要があります。または、ドライバ・インスタンスをスレッド化しない適切な構成を設定できるように、システム管理者に使用方法を明確に伝える必要があります。

次の項では、OracleAS Wireless で動作する SMS ドライバ・インタフェースの開発に必要な、主要クラスおよびインタフェースについて説明します。

クラス oracle.panama.messaging.transport.TransportLocator

クラス `TransportLocator` では、メッセージのインタフェースとドライバ・インタフェースへの初期アクセスを提供するインタフェースを定義します。このクラスに定義される 2 つの主要メソッドは、次のとおりです。

- `getDriverController()` は、ドライバ・インタフェース用のインスタンスを戻します。
- `getMessagingController()` は、メッセージ・インタフェース用の Controller インスタンスを戻します。

インタフェース oracle.panama.messaging.transport.Driver

これは、特定のプロトコル用のドライバ開発に使用するメイン・インタフェースです。ドライバを開発するには、ドライバ・インタフェースを実装します。このインタフェースを実装する場合のコンポーネントは、限定付き OracleAS Wireless ドライバです。

init() メソッドと destroy() メソッド この 2 つは、ドライバ・インスタンスのライフ・サイクルを制御するメソッドです。init メソッドに渡される初期化プロパティは、OracleAS Wireless Tools の構成フレームワークを介して指定します。

init() メソッドは、次のいずれかの初期化ステータスを戻します。

```
Driver.FAILED, Driver.SEND, Driver.RECEIVE
Driver.SEND_RECEIVE.
```

戻されるステータスに、OracleAS Wireless Tools を介して構成したステータスとの整合性があるかどうかを確認してください。両者が異なる場合は、このメソッドで戻され、Webtool を介して構成されるステータスの値が使用されます。

send() メソッド ドライバは、このメソッドを実装して、特定のプロトコルによるメッセージの送信に適した処理を実行します。配信するコンテンツは send() メソッドに渡される Message オブジェクトに格納され、アドレス・パラメータではメッセージの配信先となる 1 人以上の受信者を指定します。

また、ドライバは、各メッセージの一意 ID または各受信者の ID を戻すことが予期されます。これらの ID は、トランスポート・システムで必要に応じて配信ステータスの問合せに使用されます。

> **注意：** 後述されているクラスはすべて、別のパッケージの指定がないかぎり、oracle.panama.messaging.transport パッケージを使用していることを想定しています。

ドライバがトランスポートを再試行するためには、NULL のメッセージ ID を戻す必要があります。send メソッドでスローされた例外が確認されると、送信ステータスとしてログに記録されます。send （）メソッドで例外がスローされた場合、トランスポートは再試行しません。send （）メソッドでタイプ DriverException の例外がスローされると、例外コードが検証されます。例外コードが致命的とマークされると、このドライバ・インスタンスの送信機能は取り消されます。

- 例外が致命的とマークされていなければ、ドライバは引き続き他のメッセージの送信に使用されます。
- 例外のタイプが DriverException でない場合、ドライバは、他のメッセージの送信にも使用されます。

ドライバは、致命的でない DriverException をスローする前にリカバリを試行します。たとえば、TCP/IP 接続が切断されると、ドライバは例外をスローせずに再接続を試みます。ドライバがリカバリを試行せずに致命的でない例外をスローすると、トランスポートは引き続きメッセージを送信することになります。エラーがリカバリされないので、その送信は失敗となります。これは、負荷が大きい場合は特に、システムのパフォーマンスが低下します。

receive() メソッド ドライバは、このメソッドを実装して、特定のプロトコルによるメッセージまたはステータス・レポート（あるいはその両方）の受信に適した処理を実行します。

前述のように、トランスポート・システムによって操作が駆動されます。通常、ドライバは、メッセージの受信後にこのメソッドから戻ることが予期されます。このような制御によってトランスポート・システムが通常の状態に戻るため、次の最適な手順を決定できます

`receive()` メソッドは、トランスポート・システムによって継続的にコールされます。そのため、メッセージを受信しない場合は `receive()` メソッドでブロックすることをお薦めします。ただし、無限にブロックしないでください。そうでない場合は、リソース集中型の操作とみなされ、receive をコールしたスレッドが終了します。リソース集中型のスレッドの経過時間は、「ランタイム構成」の「リクエストに対する最大実行時間（秒）」を設定することで構成できます（デフォルトは 60 秒です）。メッセージを受信した場合は、メッセージ・リスナーの `onMessage` メソッド（または、メッセージがステータス・レポートの場合は、ステータス・リスナーの `onStatus` コールバック）をコールして、メッセージを送信する必要があります。メソッドがタイプ DriverException の例外をスローし、トランスポートに対して `receive ()` メソッドのコール停止を要求するために致命的としてマークする場合があります。このように設計されているのは、ドライバの論理とスレッドの制御を簡素化するためです。

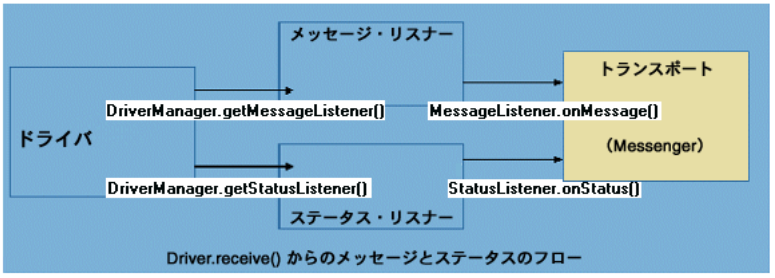
getStatus() メソッド トランスポートは、特定のメッセージの配信ステータスを取り出すときに、このメソッドをコールします。

queryTracking() メソッドと queryNotifying() メソッド 一部のプロトコルでは、外部サービスへのアクティブなポーリングを実行して、以前に送信したメッセージのステータスを調べる必要があります。これらのメソッドはトランスポートによってコールされ、ステータスを取り出すために `getStatus()` を発行する必要があるか、またはこれらのコールなしでドライバからトランスポートにステータスが渡されるかが判断されます（後者の場合、通常はドライバが `receive()` 内で `onStatus()` をコールします）。

インタフェース oracle.panama.messaging.transport.DriverController

このインタフェースは、他のドライバ関連ユーティリティとメッセージ・リスナーなどのインタフェースへのエントリ・ポイントを提供します。Locator の `getDriverController()` メソッドをコールして、DriverController のインスタンスを取得できます。

図 10-13 メッセージとステータスのフロー



getMessageListener() メソッドと getStatusListener() メソッド この 2 つのメソッドは、トランスポートに受信メッセージまたはステータスのコールバック・インスタンスを戻します。通常は、ドライバ・インタフェースの `receive()` メソッドの実装内で `onMessage()` または `onStatus()` メソッドをコールして、トランスポート・システムにそれぞれメッセージとステータスを渡します。

インタフェース oracle.panama.messaging.transport.GSMSmartMSGEncoder

OTA WAP プロビジョニング、着信音、グラフィックスなど、GSM スマート・メッセージ配信を実装で処理する必要がある場合は、このインタフェースが役立ちます。OracleAS Wireless には、このインタフェースのデフォルト実装が用意されており、プロパティ `wireless.messaging.gsmsms.encoder.class` の値を取得することによって検索できます。このデフォルト実装では、Nokia および Ericsson のハンドセットについて OTA WAP プロビジョニング、着信音およびグラフィックスが処理されます。

この基本的な機能を拡張する場合は、このインタフェースを拡張して独自の実装を開発します。その後は、独自に実装したクラスの値を使用してトランスポートのプロパティ wireless.messaging.gsmsms.encoder.class を構成する必要があります。

encode() メソッド これはインタフェースに必須の唯一のメソッドです。パラメータでは、タイプ（着信音、グラフィックス）、機種（Nokia、Ericsson）および要求されたタイプに関連するすべての属性を指定します。

情報を処理してエンコードされたメッセージを GSMSmartMsg 形式で戻します。このメッセージは、メッセージと特定のスマート・メッセージ情報のフラグメントです。

タイプ、機種または他の属性が実装でサポートされていない可能性があります。この場合は、次の 2 つの選択肢があります。

- 処理方法がわかっている場合、およびユーザー・データが破損したり適切でないことが確実な場合は、トランスポート例外をスローします。この場合は、（物理的な Java プロセスではなく）スマート・メッセージ・プロセスが終了します。
- 処理方法が不明な場合は、例外をスローせずに単に NULL を戻します。

この場合、（UCP ドライバ、SMPP ドライバなどの）ビルトイン SMS ドライバは、GSMSmartMsgEncoder のデフォルト実装に戻り、この状況を処理できるかどうかを調べます。これは、ドライバが正しいセマンティクスで開発されているかどうかによって異なります。GSMSmartMsgEncoder の機能拡張に重点を置いている場合は、開発内容を最大限に利用できるように、この規則に従ってください。

インタフェース oracle.panama.messaging.transport.MessageListener および StatusListener

これらのインタフェースのインスタンスを取得するには、DriverController インタフェース内で適切なメソッドをコールします。

通常、これらのインタフェースはドライバ・インタフェースの `receive()` メソッドの実装内で使用され、トランスポート・システムにメッセージまたはステータスの可用性を通知します。

クラス oracle.panama.messaging.common.Message

このメッセージ・クラスは、配信または受信されるコンテンツを取得するために使用されます。このクラスは包括的であり、電子メールと同様の高機能を備えています。複数の部分からなるメッセージがサポートされており、コンテンツに MIME タイプを関連付けることができます。

ただし、特定の部分または MIME タイプの処理方法は、ドライバの実装によって決定されます。

情報がドライバ・インタフェースを介してドライバに渡されない場合は、ユーザーのアプリケーションからドライバに情報を渡すには、メッセージのヘッダーが適していると考えられます。

クラス oracle.panama.messaging.common.ContentTypes

これはドライバ専用クラスではありません。このクラスでは、標準的な MIME タイプの他に少数のコンテンツ・タイプ（MIME タイプ）を指定します。ドライバ実装者は、これらの MIME タイプを見つける可能性があります。これらの MIME タイプの処理方法は個々のドライバに応じて異なりますが、見落とさないようにする必要があります。

ドライバのプロパティ

OracleAS Wireless Tools を使用して OracleAS Wireless にドライバを追加する場合は、次の表に示す一連のプロパティを指定する必要があります。

表 10-7 ドライバのプロパティ

名前	説明
名前（必須プロパティ）	ドライバの任意名。
クラス（必須プロパティ）	ドライバを実装するクラス。
デリバリのカテゴリ（必須プロパティ）	SMS、電子メールなど、ドライバでサポートされているトランスポートの、サポートされているカテゴリ。
プロトコル	SMPP や UCP など、ドライバで送信に使用される特定のプロトコル。この値はオプションです。デフォルトは "*" で、可能なプロトコルすべてを意味します。
電話会社	Cingular、Telia など、ドライバでサポートできる電話会社。これは、特に SMS 領域に意味を持つオプション・プロパティです。デフォルトは "*" で、可能な電話会社すべてを意味します。
スピード	ドライバのスピード。このプロパティを使用するとロード・バランシングを改善できます。このプロパティはオプションで、可能な値は 0 ～ 10、デフォルト値は 0（最も低速）です。
コスト	このドライバを使用するコスト。このプロパティを使用するとロード・バランシングを改善できます。このプロパティはオプションで、可能な値は 0 ～ 10、デフォルト値は 0（最も低コスト）です。
機能	ドライバが送信用か、受信用か、または送受信用か。このプロパティはオプションで、デフォルトは送信専用です。
エンコードロケール	使用しません。

ドライバ用のカスタム・プロパティ

ドライバのインストール時に、ドライバのカスタム・プロパティを指定して機能させることができます。たとえば、電子メール・ドライバを機能させるには、imap ホスト名のプロパティが必要になる場合があります。ドライバは、名前 `imap.hostname` などのプロパティを予期するようにコーディングできます。OracleAS Wireless Tools を介してドライバを追加する場合は、必要な数のプロパティ名を指定できます。ドライバ・インスタンスの作成時には、指定したプロパティの特定の値を設定できます。たとえば、同じドライバ・コードから 2 つの電子メール・ドライバ・インスタンスをインストールし、それぞれのインスタンスで 2 つの異なる IMAP サーバーに imap ホスト名を提供できます。

これらのカスタム・プロパティは、`init()` のコール時にドライバ・インスタンスに渡されます。カスタム・プロパティ・セットの他に、次のような OracleAS Wireless のサイト・レベルのプロパティも暗黙的に渡されます。

- wireless.log.directory
- wireless.firewall.http.use.proxy
- wireless.firewall.http.proxy.host
- wireless.firewall.http.proxy.port
- wireless.firewall.http.non.proxy.hosts
- wireless.firewall.ftp.use.proxy
- wireless.firewall.ftp.proxy.host
- wireless.firewall.ftp.proxy.port
- wireless.firewall.authentication.set
- wireless.firewall.authentication.username
- wireless.firewall.authentication.password

例 : サンプル・ドライバ

```
// Copyright (c) 2001 Oracle Corporation. All rights reserved.
package oracle.panama.messaging.transport.driver.sample;
/**
* A SimpleDriver class.
* <P>
* @author Oracle Corporation
*/
import Java.util.Properties;
import Java.net.ServerSocket;
import Java.net.Socket;
import Java.io.BufferedReader;
import Java.io.PrintStream;
import Java.io.InputStreamReader;
import Java.io.IOException;
import Java.io.PrintWriter;
import Java.io.FileOutputStream;
import Java.text.SimpleDateFormat;
import oracle.panama.messaging.transport.*;
import oracle.panama.messaging.common.*;
import oracle.panama.model.DeliveryType;
import oracle.panama.util.MessageCatalog;
import oracle.panama.core.admin.L;

/**
* A Simple driver
*
* @author ashah, jxiang
```

```
*/
public class SimpleDriver implements Driver {
private String mCompanyName;
private String mDeliveryType;
private String mVersion;
private PrintWriter log = null;

/**
* Initialize the driver.
*
* @param properties the driver's properties.
* @return the initialization status.
*/
public int init(Properties properties) {
// get the locator instance and various listeners
TransportLocator locator = TransportLocator.getInstance();
DriverController manager = locator.getDriverController();
mMessageListener = manager.getMessageListener();
mStatusListener = manager.getStatusListener();
// read pr operties
mCompanyName = properties.getProperty("company-name");
// delivery type is needed. Use SMS
mDeliveryType = DeliveryType.SMS.getName();
mVersion = "1.0";
int status = Driver.FAILED;
try {
String logName = properties.getProperty("logfile");
if (logName == null)
logName = new String("SimpleDriver.log");
log = new PrintWriter(new FileOutputStream(logName, true ) );
} catch(Exception e) {
e.printStackTrace();
return status;
}
log ("initialized: " + new Java.util.Date());
mPort = Integer.parseInt(properties.getProperty("port"));
mDelay = 20000; // 20 seconds
mMessage = new Message();
mSemaphore = new Object();
status = Driver.SEND_RECEIVE; // TODO - verify the return code
mStatus = new Status();
log ("init complete");
return status;
}

/**
* Destroy the driver.
```

```
*/
public void destroy() {
try {
log ("destroy");
mServerSocket.close();
mReader.close();
mWriter.close();
log ("destroy comp lete");
}
catch (Exception e) {
}
}

/**
* Get the accepted attributes of the driver.
*
* @return the accepted attributes of the driver.
*/
public Parameter[] getParameters() {
Parameter[] parameters = new Parameter[3];
parameters[0] = new Parameter("company-name", "a company name", true, null);
parameters[1] = new Parameter("logfile", "the log file name", false,
"SimpleDriver.log");
parameters[2] = new Parameter("port", "the listening port of the driver", true,
null);
return parameters;
}

/**
* Get the version of the driver.
*
* @return the version of the driver.
*/
public String getVersion() {
return mVersion;
}

/**
* Get additional information of the listener.
*
* @return the information of the listener.
*/
public String getInfo() {
return "Simple Driver";
}

/**
```

```
* Send a message to a single address with this driver.
*
* @param address the address to send to.
* @param encoding the encoding of the device.
* @param tracking the tracking level.
* @param expiration the expiration time.
* @param reliability the reliability level.
* @param priority the priority level.
* @param fromAddr the from-address.
* @param replyToAddr the reply-to-address.
* @param mid a unique message id can be used to generate return message id.
* @param message the message to send.
* @return a unique message id, null if failed.
*/
public String send(String dtype, String address, int mode, String encoding,
int tracking, int expiration, int reliability, int priority, String fromAddr,
String replyToAddr, String mid, Message message) {
log ("send: " + address + " => " + message.getContent());
String id = null;
try {
id = mid + ':' + address;
mWriter.println(id);
mWriter.println(message.getContent());
mWriter.flush();
}
catch (Exception e) {
// no t synchronized, it works for this toy.
mWriter = null;
mReader = null;
id = null;
}
log ("sent id: " + id );
return id;
}
/**
* Send a message to a list of addresses with this driver.
*
* @param addresses the addresses to send to.
* @param encoding the encoding of the device.
* @param tracking the tracking level.
* @param expiration the expiration time.
* @param reliability the reliability level.
* @param priority the priority level.
* @param fromAddr the from-address.
* @param replyToAddr the reply-to-address.
* @param mid a unique message id can be used to generate return message id.
* @param message the message to send.
```

```
* @return a list of unique message ids, null if failed.
*/
public String[] send(String dtype, String[] addresses, int[] modes,
String encoding, int tracking, int expiration, int reliability,
int priority, String fromAddr, String replyToAddr,
String mid, Message message) {
String[] ids = null;
log ("send: mult iple => " + message.getContent());
try {
int count = addresses.length;
ids = new String[count];
String id = mid + ':' + addresses[0];
ids[0] = id;
mWriter.print(id);
for (int i=1; i<count; i++) {
id = mid + ':' + addresses[i];
ids[i] = id;
mWriter.print(',' + id);
}
mWriter.println();
mWriter.println(message.getContent());
mWriter.flush();
}
catch (Exception e) {
// no t synchronized, it works for this toy.
mWriter = null;
mReader = null;
ids = null;
}
log ("sent multiple");
return ids;
}

/**
* Send a message to a list of addresses with this driver.
*
* @param dtypes the delivery types for all destinations
* @param addresses the addresses to send to.
* @param modes the delivery modes
* @param encoding the encoding of the device.
* @param tracking the tracking level.
* @param expiration the expiration time.
* @param reliability the reliability level.
* @param priority the priority level.
* @param fromAddr the from-address.
* @param replyToAddr the reply-to-address.
* @param mid a unique message id can be used to generate return message id.
```

```
* @param message the message to send.
* @return a list of unique message ids, null if failed.
*/
public String[] send(String[] dtypes, String[] addresses, int[] modes,
String encoding, int tracking, int expiration, int reliability,
int priority, String fromAddr,String replyToAddr, String mid,
Message message) throws DriverException {
String[] ids = null;
int count = addresses.length;
log (" send: " + count + " recipients : " + message.getContent());
ids = new String[count];
for (int i=0; i<count; i++) {
ids[i] = send(dtypes[i], addresses[i], modes[i], encoding, tracking,
expiration, reliability, priority, fromAddr,
replyToAddr, mid, message);
}
return ids;
}

/**
* Get the sending status of a message. The
* status got by this call should be reported
* the transport system via the driver listener
* onStatus callback.
*
* @param mid the id of the message.
*/
public void getStatus(String mid) {
}

/**
* Get the sending statuses of a list of messages.
* The statuses got by this call should be reported
* the transport system via the driver listener
* onStatus callback.
*
* @param mids the ids of these messages.
*/
public void getStatus(String[] mids) {
}

/**
* Check if query is required to get the notification.
*
* @return true if required, false otherwise.
*/
public boolean queryNotifying() {
```

```
return false;
}

/**
* Check if query is required to track the
* sending status.
*
* @return true if required, false otherwise.
*/
public boolean queryTracking() {
return false;
}

/**
* Receive a message/status. If any message/status
* is received, the driver should use the onMessage/
* onStatus callbacks of the driver listener (got
* via the controller) to report it to the transport
* system. This method should do something if the
* initization status has the RECEIVE ability.
*/
public void receive() {
log ("receive started");
synchronized (mSemaphore) {
try {
if (mServerSocket == null) {
try {
mServerSocket = new ServerSocket(mPort);
mServerSocket.setSoTimeout(mDelay);
}
catch (IOException ioe) {
mServerSocket = null;
mSocket = null;
throw ioe;
}
}
if (mSocket == null) {
try {
mSocket = mServerSocket.accept();
mSocket.setSoTimeout(mDelay);
}
catch (IOException ioe) {
mSocket = null;
throw ioe;
}
}
if (mReader == null) {
```

```
mReader = new BufferedReader(
new InputStreamReader(mSocket.getInputStream()));
mWriter = new PrintStream(mSocket.getOutputStream());
}
String buf = mReader.readLine();
log ("receive read: " + buf);
if (buf.charAt(0) == '*') {
String address = buf.substring(1);
mMessage.setContent(mReader.readLine());
DeviceInfo info = new DeviceInfo();
info.setDeliveryType(mDeliveryType);
info.setEncoding("7b");
String from = "FROM-ME-TODO";
mMessageListener.onMessage(from, info, address, mMessage);
log ("message sent to message listener");
}
else {
mStatus.setContent("received");
mStatusListener.onStatus(buf.substring(1), mStatus);
log ("status sent to status listener");
}
}
catch (IOException ioe) {
mReader = null;
mWriter = null;
}
}
}

/**
* write to message log
*
* @param message string
*/
void log(String message) {
if( log != null ) {
synchronized( log ) {
String currentTime = new SimpleDateFormat(
"yyyy-MM-dd HH:mm:ss").format( new Java.util.Date() );
log.println(currentTime + " " + message);
log.flush();
}
}
}

private Socket mSocket;
private Object mSemaphore;
```

```
private ServerSocket mServerSocket;
private MessageListener mMessageListener;
private StatusListener mStatusListener;
private BufferedReader mReader;
private PrintStream mWriter;
private Message mMessage;
private Status mStatus;
private int mDelay;
private int mPort;
}
```

OracleAS Wireless 9.0.2x ドライバのアップグレード

OracleAS Wireless 9.0.2x のドライバのインタフェースは、9.0.4 のインタフェースと互換性がないため、9.0.4 トランスポート・サーバーで使用するには 9.0.2x ドライバをアップグレードする必要があります（反対に、9.0.2x トランスポート・サーバーで使用するには 9.0.4 ドライバをダウングレードする必要があります）。ビルトイン・ドライバがアップグレードされており、現行のリリースで動作します。カスタマ開発による 9.0.2x ドライバのみに影響します。この項では、9.0.2x ドライバのアップグレード方法を示します。

9.0.2x と 9.0.4 のドライバ・インタフェースの主な相違点は次のとおりです。

- メソッド `getParameter()` が追加されました。
- 3 種類の `send()` メソッドすべてが変更されました。
- パラメータ・リストに複数の新しい属性が追加されました。

新規メソッドと変更されたメソッド

- `getParameter()` メソッド

  このメソッドは、ドライバが受け入れる属性をリストします。このメソッドをコールして、使用できる属性情報を動的に取得できます。

- `send()` メソッド

  これらのメソッドは、トランスポートからドライバにさらに多くのパラメータを渡すように拡張されています。当初のパラメータはいずれも削除されていないため、9.0.2x ドライバを簡単にアップグレードできます。新しく追加されたパラメータは有用ですが、必須ではありません。新しく追加されたパラメータには、優先順位、メッセージ ID（mid）および有効期限が含まれています。優先順位は、メッセージの優先レベルです。可能な値は `MessageInfo` に定数で定義されています。メッセージ ID は、一意のリターン・メッセージ ID を生成するために、ドライバがシードとして使用できる一意のメッセージ ID です。一般に、ドライバはこのメッセージ ID（宛先アドレスに追加される）をリターン・メッセージ ID として使用できます。有効期限は、メッセージが失効する時間（秒単位）です。値が 0（ゼロ）以下の場合、メッセージは失効しません。新し

く追加されたパラメータの詳細は、新しいドライバ・インタフェースの Javadoc を参照してください。

9.0.2x ドライバをアップグレードするには、次の手順を実行する必要があります。

1. 新しい `getParameter()` メソッドを追加します。
2. 最新のシグネチャを使用するために、3 種類の `send()` メソッドすべてを変更します。

トランスポート・サーバーの拡張 : フック

アプリケーションでは、フックのタイプに応じて、メッセージの送受信中に起動するフックをインストールできます。フックはすべてオプションです。通常、フックにはアプリケーションで指定する情報がすべて渡され、適切な操作を実行できます。フックはルーティング情報の提供に役立ち、他のカスタム論理を実行する場合もあります。

フックは、主に次の 2 つのカテゴリに分かれています。

- 名前付きフック : 各種のフックを最大 1 つのみ追加できます。追加するには、OracleAS Wireless Tools の「構成」を使用する必要があります。
- 一般的なフック : 一般的なフックには、プリセンド、ポストセンド、プリレシーブ、ポストレシーブの 4 種類があります。各種のフックが存在しない場合と複数が存在する場合があり、OracleAS Wireless Tools を介して、または Messenger インタフェースで使用可能なメソッドを介してプログラムによって追加または削除できます。

この製品には、デフォルト・フックは用意されていません。

名前付きフック

- DriverFinder（インタフェース `oracle.panama.messaging.transport.DriverFinder`）。このフックに予期されるセマンティクスは、配信リクエスト用のドライバ名を入力することです。
- CarrierFinder（インタフェース `oracle.panama.messaging.transport.CarrierFinder`）。これは、OracleAS Wireless Tools を介して構成できる名前付きフックです。このフックに予期されるセマンティクスは、特定のデバイス・アドレス用の電話会社を検索することです。電話会社情報は、DriverFinder またはトランスポート・システムで使用され、ルーティングが実行されます。このフックは、通常はメッセージごとに一度コールされます。この種のフックは 1 つしか使用できません。
- SmartMsgEncoder（インタフェース `oracle.panama.messaging.transport.SmartMsgEncoder`）。このフックは、GSM または CDMA（あるいはその両方）のスマート・メッセージのエンコードに使用されます。リリース 9.0.2.x では、このフックは `GSMSmartMsgEncoder` と呼ばれています。このフックは、GSM スマート・メッセージのエンコードのみをサポートします。このインタフェースは、GSM と CDMA 両方のスマート・メッセージをサポートするように、このリリースで拡張されました。ただし、セマンティクスの変更はありません。

フックがメッセージをエンコードできない場合は、適切なエンコーダが見つかるまで他のエンコーダが起動されるように、NULL を戻してその旨を示す必要があります。従来のエンコーダは、新しいインタフェースにアップグレードされます。9.0.2.x からこのリリースに実装をアップグレードするには、クラス定義を `implements oracle.panama.messaging.transport.GSMSmartMsgEncoder` から `extends oracle.panama.messaging.transport.SmartMsgEncoder` に変更し、メソッド名を `encodeSmartMsg` から `encodeGSMSmartMsg` に変更します。

OracleAS Wireless メッセージ・システム

- FailOverHook（インタフェース `oracle.panama.messaging.transport.FailOverHook`）。このフックは将来のために予約されています。

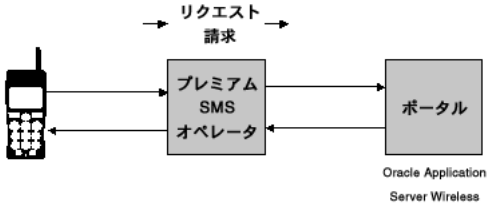
一般的なフック

- PreSendingHook
（インタフェース `oracle.panama.messaging.transport.GeneralHook`）。このフックは、メッセージの送信前にコールされます。
- PostSendingHook
（インタフェース `oracle.panama.messaging.transport.GeneralHook`）。このフックは、メッセージの送信後にコールされます。
- PreReceivingHook
（インタフェース `oracle.panama.messaging.transport.GeneralHook`）。このフックは、受信されたメッセージがリスナーに渡される前にコールされます。
- PostReceivingHook
（インタフェース `oracle.panama.messaging.transport.GeneralHook`）。このフックは、受信されたメッセージがリスナーに渡された後にコールされます。

プレミアム SMS と逆課金 SMS のサポート

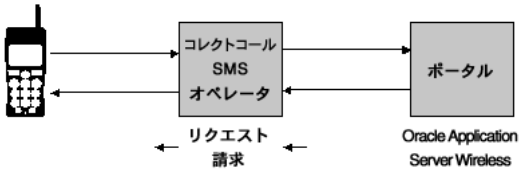
SMS 電話は、Async によって処理される主要なデバイスの 1 つです。SMS の分野で普及している課金モデルはプレミアム SMS です。これは、SMS に変換された音声電話の 900 モデルに似ています。SMS 電話はメッセージ（サービスに必要な情報すべて）を短縮番号（ラージ・アカウントと呼ばれます）に送信します。メッセージはコンテンツ・プロバイダにルーティングされます。コンテンツ・プロバイダは番号に送信された SMS メッセージをリスニングするようにシステム設定されています。ユーザー・リクエストが処理され、結果がデバイスに送信されます。エンド・ユーザーは、モバイル・デバイスから発行されたリクエスト・メッセージの典型的な SMS 転送料金に加えて、サービス利用料がチャージされます。利用料は、起動されるサービスのタイプによって異なります。リクエスト・チャージは、加入者の既存の SMS 電話請求書に反映されます。請求サイクルの終わりに、利用料は電話会社（プレミアム SMS オペレータ）とコンテンツ・プロバイダ間で分割されます。図 10-14「プレミアム SMS のプロセス」に、プロセスを示します。

図 10-14 プレミアム SMS のプロセス



プレミアム SMS サービスにアクセスするには、サービスを識別するために、モバイル・ユーザーがキーワード（OracleAS Wireless の用語では非同期短縮名）を使用してラージ・アカウント（OracleAS Wireless の用語では非同期アドレス）にメッセージを送信します。デバイスからプレミアム SMS オペレータのネットワークにリクエスト・メッセージが送信されると、事前定義済のサービス利用料（ラージ・アカウントに関連付けられている）がモバイル加入者にチャージされます。コンテンツ・プロバイダは、メッセージを受信すると、サービス・キーワードによって識別された対応するサービスを起動します。サービス結果は、別の SMS メッセージを介してモバイル加入者に返信されます。

図 10-15 逆課金 SMS



逆課金 SMS は、結果 SMS メッセージに対してモバイル加入者にサービス利用料をチャージする課金モデルです。リクエストのサービス利用料は、モバイル・ユーザーがアクセスするサービスによって異なります。各サービスには、関連付けられている料金クラスがあります。また、サービス・プロバイダがサービス結果を生成すると、オペレータが請求トランザクションを正確に記録できるように、逆課金 SMS オペレータに必要な情報を提供します。

プレミアム SMS と逆課金の新機能

OracleAS Wireless でプレミアム SMS と逆課金モデルを適切にサポートするために、次の新機能が追加されました。

- サービスを分類するために、サービス・カテゴリと呼ばれるグループ化オブジェクトが作成されました。このグループは、論理的ないくつかの類似点を共有する一連のサービスを表します。たとえば、同じ利用料レベルを持つプレミアム SMS サービスは、1 つのサービス・カテゴリに入れることができます。
- 各アクセス・ポイント（たとえば、非同期アドレス）は、1 つ以上のサービス・カテゴリに任意に関連付けることができるようになりました。これによって、OracleAS Wireless のアクセス・ポイントが、アクセス可能な一連のサービスにバインドされます。一連のサービスに関連付けられているアクセス・ポイント外部のサービスを起動しようとした場合、サービスの起動は失敗となります。プレミアム SMS の場合、ラージ・アカウントを表す非同期アドレスは、ラージ・アカウントの利用料レベルを反映する一連のサービス・カテゴリにマップできます。この関連付けは、サービスとアクセス・ポイント間の利用料レベルに基づいて、リクエストの認可モデルを提供します。利用料レベルが 10 セントのラージ・アカウントに送信されるリクエストは、それを上回る利用料のサービスにアクセスすることはできません。サービス・カテゴリの作成や、サービスとアクセス・ポイントの関連付けは、OracleAS Wireless Tools を介して実行できます。
- 一連のルーティング情報パラメータが、非同期アプリケーションの属性の一部として追加されました。結果メッセージとともに値がメッセージ・ヘッダーとして返信されるように、請求情報（ラージ・アカウントなど）をサービスに関連付けることができます。メッセージが正しいアカウントを介してチャージされるように、情報は、メッセージ・ドライバに、最終的にはプレミアム SMS または逆課金 SMS オペレータに引き継がれます。
- ルーティング情報がプリセットとして実装されるようになりました。`_MESSAGE_ROUTE_` と呼ばれる事前シード済プリセット定義が、標準インストールに組み込まれています。管理者は、独自の要件に応じてフィールドを追加、変更または削除して、プリセット定義を編集できます。定義の編集によって、結果メッセージの属性の定義に柔軟性がもたらされます。

プレミアム SMS サービスの有効化

OracleAS Wireless サービスをプレミアム SMS 対応にするには、次の手順に従います。個々の詳細な手順は、『OracleAS Wireless 管理者ガイド』を参照してください。

1. Enterprise Manager を使用して、「すべてのアプリケーションへのアクセスを許可」フラグの選択を解除してアクセス・ポイントを作成します。アクセス・ポイントのアドレス値には、プレミアム SMS オペレータから提供されるラージ・アカウントを指定します。
2. OracleAS Wireless Tools を使用して、サービス・カテゴリを作成し、前述の手順で作成したアクセス・ポイントにこのサービス・カテゴリを関連付けます。このオブジェクトは、アクセス・ポイントからアクセス可能なすべてのサービスをグループ化します。
3. 新しく作成されたサービス・カテゴリに（アクセス・ポイントからアクセス可能な）非同期アプリケーションすべてを割り当てます。
4. （オプション）事前シード済の _MESSAGE_ROUTE_ プリセット定義を編集して、結果メッセージの請求情報として SMS ドライバに送信されるメッセージ・ヘッダーを各ポータルがカスタマイズできるようにします。たとえば、ROUTE_COST_LEVEL の記述は、コスト・レベルから料金クラスに変更できます。そのメタ・フィールドも追加または削除できます。

   デフォルトでは、ROUTE_CHANNEL および ROUTE_REV_CHANNEL の 2 つのフィールドの値は、結果メッセージの「送信者」および「返信先」フィールドに設定されます。そのため、カスタム組込みのドライバなしでプレミアム SMS オペレータに情報を渡すことができます。管理者がマッピングを変更する必要がある場合は、2 つの属性 wireless.async.routeinfo.to と wireless.async.routeinfo.replyto を修正します。
5. コンテンツ・マネージャを使用して、プレミアム SMS 対応のすべてのアプリケーションに SMS ルーティング情報を追加します。典型的な例は、返信メッセージをチャージするラージ・アカウントの値を「チャネル」フィールドに割り当てることです。

11

通知エンジン

項ごとに様々なトピックを記載しています。各項の内容は、次のとおりです。

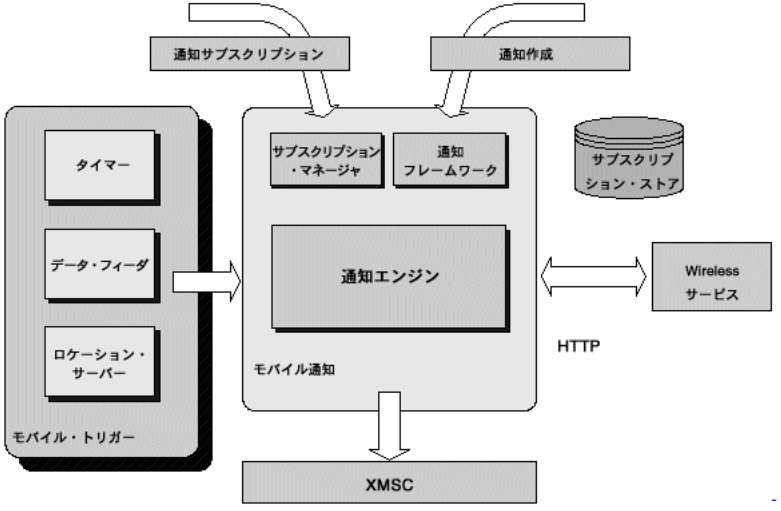
- 概要とアーキテクチャ
- 通知の作成
- データ・フィーダ
- 統合された通知ソリューション
- 通知システムの移行

概要とアーキテクチャ

OracleAS Wireless の通知システムは、モバイル通知メッセージを事前定義の条件に基づいて配信するための柔軟でスケーラブルなメカニズムを提供します。メッセージのコンテンツは、通常の OracleAS Wireless アプリケーションを起動すると、ユーザー定義の条件を確認した上で生成されます。エンド・ユーザー・メッセージは、OracleAS Wireless のメッセージ・アプリケーションを使用して配信されます。詳細は、第 10 章「メッセージ・アプリケーションの作成」を参照してください。

通知エンジンは、時間、データまたはロケーションのプロバイダからイベントを受信するたびに、条件を評価します。このエンジンは、基礎となるフィードが変更されるたびにトリガーされます。

図 11-1 通知フロー



通知トリガーには、主として次の 3 つのタイプがあります。

- データ・トリガー : データ・フィーダを使用してデータ・イベントを生成します。このイベントを受信すると、通知システムは、ユーザー定義の条件を評価し、条件と一致するユーザーにメッセージを送信します。データ・イベントには、複数の条件を定義できます。各条件は AND または OR で結合できます。ただし、各条件間に AND/OR が混合した演算を定義することはできません。このタイプの通知アプリケーションの一般的な例には、リクエストされた株が特定の値に達した場合は、ユーザーに株価情報を送信する例があります。
- 時間トリガー: 時間ベースのトリガーを単独またはデータ条件と併用して使用できます。たとえば、ユーザーは、毎日午後 3 時に株式指数の通知メッセージをリクエストできます。時間ベースの通知の場合、ユーザーは、時間の条件に加えて、メッセージの受信に

対して、指定した時間に株式指数が特定の値に達した場合というようなデータ条件も指定できます。

- ロケーション・トリガー : ロケーション・イベントは別のコンポーネント（ロケーション・イベント・サーバー）によって生成されます。ロケーション・ベースの条件は、次のような使用方法があります。
  - 別のユーザーのロケーションの評価 : 特定のトラックが顧客サイトに到着した場合は知らせてください。
  - ユーザー・グループのロケーションの評価 : チーム・メンバー全員がオフィスにいるときに、メッセージを送信してください（グループ・ミーティングを設定できるように）。
  - 別のユーザー・ロケーションへの問合せを指定の時間に実行（ロケーションと時間）: 子供が午前 9 時から午後 3 時までの間に学校にいない場合は知らせてください。
  - ロケーション条件の確認がそれ以降の処理に対する入力となるプロモーションおよびトラフィック・レポート・タイプのアプリケーション : 特定の店舗の前にいるときにプロモーションに関する通知メッセージを送信してください。または仕事をしていないか在宅の場合は、午前 9 時にトラフィック・レポートを送信してください。

    関連項目： ロケーション機能の詳細は、第 14 章「ロケーション・サービスの使用」を参照してください。

通知は、必ずしも前述の条件 1 つのみに基づいて送信する必要はありません。同じ通知に複数のエントリ・ポイントを指定でき、これらの組合せでトリガーできます（つまり、通知は時間とデータをベースにし、データ・フィーダとタイマーの両方でトリガーできます）。

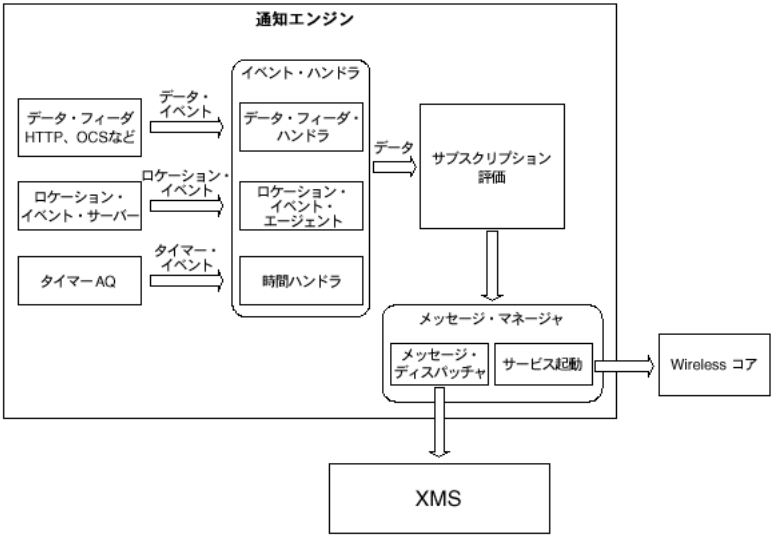
アーキテクチャ

通知エンジンは、次のタスクを実行するイベント・サーバーです。

- 着信イベントを評価して、特定のイベントに関心のある一連のユーザーを検索します。
- イベントの関連コンテンツを生成するアプリケーションを起動します。
- 生成されたコンテンツを含むメッセージをユーザーに配信します。

イベントは、基礎となるデータ・ソースが変更された場合にトリガーされます。データ・ソースにはこのイベントの評価に必要な特定のデータが含まれています。

図 11-2 通知アーキテクチャ



通知メカニズムは、次の 4 つのレイヤーで構成されています。

- イベント・ジェネレータ・レイヤー。OracleAS Wireless には、次の 3 タイプがあります。
  - データ・フィーダ : コンテンツ・ソースから通知エンジンにデータのストリームを指定の間隔で提供することで、値イベントを生成します。このデータ・フィーダは入力パラメータ（ユーザーが指定）を取得し、指定された入力パラメータのデータ行を戻します。この行の各列は、通知エンジンでは出力パラメータとみなされます。たとえば、株価フィードでは、特定銘柄に関する株価、株価収益率および株価変動を取り出すことができます。

  - タイマー : ユーザー指定の間隔で時間イベントを生成します。
  - ロケーション・イベント・エージェント : ユーザー指定の条件が確認されたときにロケーション・イベントを生成します。
- イベント・ハンドラ・レイヤー : このコンポーネントは、対応するイベント・ジェネレータからイベントを受信するたびに、この特定イベントに関心のある一連のユーザーを検索します。つまり、着信データに一致する条件を持つ一連のユーザーを導出します。たとえば、データ・フィーダが ORCL 株の株価をプッシュしてイベントを生成した場合、値イベント・ハンドラは、ORCL に関心があり、現在の株価と一致する値条件を持つユーザーを検索します。
- サービス起動レイヤー : 条件が一致するユーザーに対して、適切なアプリケーションが OracleAS Wireless Tools によって起動され、通知メッセージとしてエンド・ユーザーにプッシュされるコンテンツが取り出されます。このプロセスには、使用可能なデータ（データ・フィーダが提供）のアプリケーションへの引渡し、ユーザー・セッションと認証の作成およびコンテンツの取出しが含まれます。
- メッセージ・ディスパッチャは、生成されたコンテンツを XMS レイヤーを介してエンド・ユーザーに配信します。

主な機能

通知システムの主な機能は、次のとおりです。

- アクション可能通知 : OracleAS Wireless では、ユーザー相互作用を有効化することによって、通知機能を拡張します。ユーザーは、通知メッセージを受信できるだけでなく、メッセージに返信して、以降の処理を実行することもできます。たとえば、株価通知に対して、株の売却注文で返信できます。アクション可能通知の詳細は、第 10 章「メッセージ・アプリケーションの作成」を参照してください。
- 柔軟なメッセージ・コンテンツ : 通知メッセージ・コンテンツはアプリケーションの起動によって生成されるため、通知エンジンンによって、すべてのタイプの通知を柔軟に生成できます。この柔軟性によって、コンテンツの生成時に特定のアプリケーション・ロジックも実行できます。アクション可能な通知を使用すると、このアプリケーション・ロジックに対して以降の処理を指定できます（つまり、Oracle 株が特定の値に達したときにその株を売るという確認リクエストを含むメッセージを送信できます）。
- ロケーション・サポート : ロケーション条件は、ロケーション基準、有効期限および評価モードのセットを使用して指定できます。ロケーション基準は、リージョン（システム・リージョン、カスタム・リージョンまたはユーザー定義リージョン）、ターゲット（OracleAS Wireless ユーザー、コミュニティまたはモバイル・デバイス）およびタイプ（IN または OUT）別に定義されます。指定した基準間の関係は、ロケーション・サーバーによって AND にデフォルト設定されています。したがって、この条件が満たされるのは、すべての基準が一致した場合のみです。ロケーション条件の評価モードは、1 回のみ評価または有効期限まで評価のいずれかです。1 回のみ評価の場合は、ロケーション条件が最初に満たされた後は、評価されません。

- 時間ベースの通知 : 値ベースの条件と併用します。この場合、通知メッセージは、時間ベースとデータ・ベースの条件が両方とも一致するか、その 1 つ以上が一致した場合に送信されます。
- パーソナライズ・コンテンツと一般的なコンテンツ : マスター・アプリケーションを起動すると、通知エンジンは、詳細にカスタマイズできるコンテンツを提供できます。ただし、このアプリケーションの起動には時間がかかるため（メッセージ・テンプレートの処理に比較して）、この機能によって、通知処理の速度が遅くなり、エンジンのパフォーマンスが低下する可能性があります。この問題を解決するために、通知エンジンでは、ユーザー固有でないコンテンツの場合に、特定のイベントに対してターゲット・アプリケーションを 1 回起動し、そのコンテンツをこのイベントとコンテンツに関心のあるすべてのユーザーの間で共有できます。
- メッセージ・テンプレート : コンテンツを生成するためにマスター・アプリケーションを作成せずに、変数付きのメッセージ・テンプレートを指定します。この変数は実行時に既存のデータで置換されます。通知エンジンは、アプリケーションを起動することでメッセージを生成するため、デフォルトのマスター・アプリケーションは通知エンジンで生成でき、パラメータのマッピングも通知エンジンによって処理されます。
- 包括的な時間条件 :
  - アクティブ化の日付と有効期限 : 特定の通知をアクティブにする時期と期限切れの日付を指定できます。
  - 開始日と終了日の一時停止 : 特定の通知を指定の期間（休暇中など）一時停止できます。
  - 特定の日付の通知 : 特定の日付を選択して、1 回の通知（出発日 / 時刻に関するフライト情報の送信）を配信できます。
  - 反復通知 : 通知を特定期間にわたって繰り返してリクエストできます。たとえば、株価を午後 1 時から 4 時まで 30 分毎に受信する株価通知をサブスクライブできます。
  - 通知時間は、毎日、平日および週末などの頻度とあわせ、<Hour:Minute> で指定できます。
  - ユーザーのタイムゾーン（ユーザー・プロファイルで指定）は、時間条件で重要な役割を果たします。ユーザー指定の通知時間はすべて、そのユーザーのタイムゾーン内とみなされます。たとえば、2 人のユーザーが午前 9 時に株価を受信する通知をサブスクライブするとし、それぞれのプロファイルに、タイムゾーンとして 1 人は GMT を、もう 1 人は PST を指定した場合、GMT を指定したユーザーは、2 番目のユーザーより 8 時間前に通知を受信することになります。
- デバイス・サポート : ユーザーは通知メッセージを配信するデバイスを選択できます。デバイスに送信できる通知の最大数は日次ベースで設定できるため、この最大数に達した場合の処理方法も指定できます。処理方法の選択肢は、「残りの通知は代替デバイスに送信する」か「残りの通知は無視する」のいずれかです。

- 存在認識デバイスの選択 : ユーザーによる通知配信用デバイスの指定がない場合、OracleAS Wireless は、ユーザーが定義した連絡ルールと適切なチャネルを使用して、デバイスを選択します。連絡ルールの詳細は、第 7 章「Wireless Customization Portal」を参照してください。
- データ・ベースの条件リレーション : 複数の条件を含めた通知を定義できます。たとえば、「価格が 20 を超えた場合または変動が 10%未満の場合（あるいはその両方の場合）」のように定義できます。複数の条件を定義する場合は、条件間のリレーション・タイプを指定できます。リレーションは、AND または OR のいずれかです。リレーションの混合はサポートされていません。

下位互換性

前回のリリース以降、通知エンジンは大幅に変更されています。アーキテクチャ上の最も重要な相違は、コンテンツの生成方法です。コンテンツは、メッセージ・テンプレートを処理するのではなく、アプリケーションから取り出されるようになりました。このパラダイムの変更によって、新規エンジンでは、古いタイプの通知（Wireless の以前のリリースでアラート、アラート・サービスおよびマスター・アラート・サービスと呼ばれていた通知）を処理できません。

この下位互換性の問題を解決するために、管理者は、個別の通知プロセスを作成できます。2 つの異なるバージョンで作成された通知はそのままです。新しいエンジンが処理するのは、新しいタイプの通知のみです。新機能は旧バージョンで作成された通知には使用できません。

OracleAS Wireless 通知エンジンでは、メッセージ・テンプレートの指定をサポートしているため（11-5 ページの「主な機能」で説明）、旧バージョンで作成されたアラート・サービス（つまり、通知アプリケーション）を新バージョンに簡単に移行できます。このプロセスを自動化するツールが用意されています。移行後、移行されたアラートは、新規エンジンによって処理されるため、このバージョンに含まれる新機能を使用できます。

通知の作成

OracleAS Wireless では、通知を簡単に作成できます。単にアプリケーションを構築し、それを通知に対応させます。アプリケーションに通知を対応させるときに指定した条件に基づいて、エンド・ユーザーは、独自の基準でそのアプリケーションをサブスクライブします。これで、このプロセスは完了します。残りのプロセスは通知エンジンが実行します。実行時には、条件の照合、サブスクリプションの収集、メッセージの生成とディスパッチがすべて自動化され、通知エンジンによって管理されます。ユーザー操作は不要です。次に作成プロセスのサマリーを示します。

- OracleAS Wireless Tools の使用 :
  - データ・フィーダの作成（通知がデータ・ベースの場合）
  - 通知マスター・アプリケーションの作成
    - データ・フィーダの割当て（データ・ベースの場合）
    - トリガー条件の作成（データ・ベースの場合）
    - メッセージ・テンプレートの定義（オプション）
  - アプリケーション・リンクの作成
    - アプリケーション・リンクの通知への対応化
  - アプリケーションの公開
- OracleAS Wireless Customization Portal またはカスタム組込みのエンド・ユーザー・ポータルの使用 :
  - アプリケーション・ツリーからの通知対応アプリケーションの検索
  - サブスクリプション基準の指定
    - 時間ベースの場合は、アクティブ化の開始日と終了日、頻度および間隔を指定
    - データ・ベースの場合は、入力パラメータとトリガー条件パラメータを指定
    - ロケーション・ベースの場合は、ターゲット、リージョンおよび移動タイプを指定

この全プロセスは、OracleAS Wireless Tools と Customization Portal を使用して実行できます。これらのツールには、手順ごとに指示を表示するウィザードが用意されています。詳細は、『Oracle Application Server Wireless 管理者ガイド』を参照してください。また、開発者は、付属の公開 API を使用して、これらのタスクを実行できます。

マスター通知アプリケーションの定義

条件

マスター・アラート・アプリケーションを作成する場合、最も重要な情報は、使用可能な条件のタイプを定義することです。条件タイプには、データ・ベース、時間ベースまたはロケーション・ベースがあります。さらに、マスター通知アプリケーションをこれら3タイプの条件を組み合せて構成できます。次に、使用可能な条件タイプを示します。

- 純時間ベース : 通知エンジンは、指定した時刻または期間にアプリケーションを起動します。たとえば、午前9時にその日の予定スケジュールを送信します。この場合、通知エンジンは、ユーザー・カレンダを取得するために指定したアプリケーションを毎朝9時に起動します。
- 純データ・ベース : 通知エンジンは、値条件（複数可）が満たされた場合のみアプリケーションを起動します。たとえば、株価が指定の値を超えたときに株価情報を送信します。値ベース条件を含める場合は、このマスター通知アプリケーションが構築されているデータ・コンテンツを記述する必要があります。OracleAS Wireless では、データ・フィーダを使用してデータ・コンテンツを定義できます。データ・フィーダでは、特定のコンテンツが2つの形式、つまり入力パラメータと出力パラメータで定義されます。たとえば、株価コンテンツは、入力パラメータとしてチッカーを、出力パラメータとして株価、取引高、変動、変動率を使用するように定義できます。データ・フィーダは、入力パラメータ（複数可）を使用して、出力パラメータを一意に識別します。
- 純ロケーション・ベース : 通知エンジンは、ロケーション条件が満たされた場合にアプリケーションを起動します。たとえば、特定のトラックが顧客サイトに到着したときに通知メッセージを送信します。
- 時間およびデータ・ベース : この場合、データ条件が一致すると、指定した時間に通知が配信されます。たとえば、ORCL 株がユーザー指定の値を超えた場合は、午前9時に株価が送信されます。
- 時間およびロケーション・ベース : ロケーション条件が指定の時間に満たされた場合は、その時間に通知が配信されます。たとえば、指定したトラックが顧客サイトに午前9時に到着している場合は、通知メッセージが輸送責任者に送信されます。
- データおよびロケーション・ベース : データ条件とロケーション条件の両方が満たされた場合に通知が配信されます。たとえば、ユーザーが勤務していないときに ORCL 株がユーザー指定の値を超えた場合に、通知メッセージが送信されます。
- 時間 + データ + ロケーション・ベース : 値とロケーションの条件が満たされた場合は、指定の時間に通知が配信されます。たとえば、ORCL の株価が指定値を超え、ユーザーがオフィスにいない場合は、午前9時に通知メッセージが送信されます。

サブスクライバ・フィルタリング・フック

生成されたすべてのイベントについて、通知エンジンは、このイベントに関心があるユーザーのリストを導出します。場合によっては、通知設計者は、このリストを操作するための追加ロジックを適用する必要があります。OracleAS Wireless 通知エンジンを使用すると、開発者は、追加のフィルタ処理ロジック用に、Java インタフェースの MobileAlertSubscriberFilter を実装する Java クラスを指定できます。

トリガー条件

設計者は、データ・ベースのマスター通知に対して一連のトリガー条件を指定できます。これらのトリガー条件は、選択したデータ・フィーダの出力パラメータに基づいている必要があります。設計者は、条件ごとに出力パラメータを指定し、ユーザー入力に対して、条件タイプとデフォルト値をチェックする必要があります。たとえば、株価と変動を提供する株価フィードに対する条件は、「出力パラメータの値が < ユーザー指定の値 > を超えた場合」のように定義できます。

数値タイプの出力パラメータの場合、条件タイプは次のようになります。

- 値の変更
- 等しい
- 絶対値に等しい
- 以上
- 絶対値以上
- より大きい
- 絶対値より大きい
- 以下
- 絶対値以下
- より小さい
- 絶対値より小さい
- 等しくない

文字列タイプの変数が出力パラメータの場合、条件タイプは次のようになります。

- 先頭
- 変更
- 含む
- 最後

- 一致
- 含まない
- 不一致

注意： トリガー条件はオプションです。トリガー条件がない場合、通知エンジンは、着信データ・イベントすべてに対して通知メッセージを生成します。つまり、基礎となるデータ・フィーダがデータを取得するたびに、そのデータは通知エンジンにプッシュされます。

メッセージ・テンプレート

このパラメータはオプションです。複数の設計者によってメッセージ・テンプレートが指定されている場合は、それを使用して通知コンテンツを生成できます。テンプレートの指定されたメッセージは、妥当な Mobile XML 文書です。このマスター通知アプリケーションが、データ・フィーダに基づいている場合、データ・フィーダの入力または出力パラメータは、XML エンティティの表記規則に類似した *&<parameter name>;* の表記規則を使用して、テンプレート内で使用できます。次に、入力パラメータの *sym* と出力パラメータの *price* と *change* を含む通知のサンプル・テンプレートを示します。

```
<?xml version = "1.0" encoding = "UTF-8" standalone="yes" ?>
<SimpleResult>
  <SimpleContainer>
    <SimpleText>
      <SimpleTitle>OracleAS Wireless</SimpleTitle>
      <SimpleTextItem>Sample Notification: price: &price; and change: &change; for
stock: &sym;</SimpleTextItem>
    </SimpleText>
  </SimpleContainer>
</SimpleResult>
```

メッセージ・テンプレートが指定されている場合、通知エンジンは、必要なアプリケーション・パラメータ（URL やパラメータ・マッピングなど）を自動的に生成し、このメッセージ・テンプレートを処理して通知メッセージを生成できます。OracleAS Wireless には、汎用 JSP が含まれているため、アプリケーション設計者は、基本アプリケーションを作成し、それを通知に対応させる必要があります。このためには、このアプリケーションがメッセージ・テンプレートに基づいていることを指定します。

API サンプル : マスター通知アプリケーションの作成

次のコード部分に、時間 + データ + ロケーションをベースにした StockNotification というマスター通知アプリケーションの作成方法を示します。この通知アプリケーションには、データ・フィードとして StockFeed が指定されており、これには、*price* > *user_input*（デフォルト値は 23）および *change* > *user_input*（デフォルト値は 10）という 2 つのトリガー条件が設定されています。これら 2 つの条件間のリレーション・タイプは、AND です。さらに、メッセージ・テンプレートも提供されています。

```
MetaLocator m = MetaLocator.getInstance();
ModelFactory f = m.getModelFactory();
ModelServices s = m.getModelServices();

//Locate the data feed that will be used by this notification
DataFeeder df = s.lookupDataFeeder("StockFeed")

//Create a master notification with timebase enabled.
//Note that, new master notification interface is
oracle.panama.mobilealert.MasterAlertService
MasterAlertService mAS = null;
mAS = f.createMobileMasterAlertService("StockNotification", true, "Stock Master
Notification", df);

//Set the condition relation type to "AND"
mAS.setConditionRelationType(MasterAlertService.RELATION_AND);

//Set time based typed to strictly time based
mAS.setTimeBasedType(true);

//Enable Location based support
mAS.enableLocationBaseAlert(true);

StringBuffer msgTemplate = new StringBuffer("<?xml version = ¥"1.0¥" encoding =
¥"UTF-8¥" standalone=¥"yes¥" ?>");
msgTemplate.append("<SimpleResult><SimpleContainer>");
msgTemplate.append("<SimpleText>Stock alert for [&symbol;] price is [&price;]");
msgTemplate.append("</SimpleText>”);
msgTemplate.append("</SimpleContainer></SimpleResult>");

//Provide the default message template
//that can be used with default notification master service
mAS.setFormattedXMLTemplate(msgTemplate.toString());

//add conditions with default valuesto this notifications, i.e. if "price > 10 "

AlertConditionType cType1 = s.getMobileAlertConditionTypeByName("GT");
FeedMetaData fmdPrice = df.getOutputParameter("price");
mAS.addConditionDefinition("PriceMax", fmdPrice, cType1, "23");
```

```
AlertConditionType cType2 = s.getMobileAlertConditionTypeByName("GT");
FeedMetaData fmdChange = df.getOutputParameter("Change");
mAS.addConditionDefinition("ChangeMax", fmdChange, cType2, "10");

//Notification object does not use the persistent store/wireless caching
//Any create/update operation has to be committed with the save
mAS.save();
```

マスター通知アプリケーションのマスター・アプリケーションへのマッピング

通知コンテンツの生成には、アプリケーションの起動をお薦めします。設計者は、マスター通知アプリケーションを通知コンテンツの生成に使用するマスター・アプリケーションにマップする必要があります。場合によっては、以降の処理を実行するために、アプリケーションでマスター通知アプリケーションの入力または出力パラメータの一部またはすべてにアクセスする必要があります。たとえば、株価の監視通知は、株価が大幅に下落した場合、株を売却するための株売却アプリケーションをトリガーできます。この場合、株売却アプリケーションは、売却対象を認識するために、チッカーにアクセスする必要があります。この情報を提供するために、アプリケーション開発者は、マスター通知アプリケーションの入力および出力パラメータをマスター・アプリケーションの入力パラメータにマップしておきます。

通知エンジンは、様々なイベント・ジェネレータを処理して複数のアプリケーションを起動できる汎用イベント・サーバーであるため、マスター通知アプリケーションとマスター・アプリケーション間で多対多のマッピングを作成できます。つまり、特定のマスター通知アプリケーションは、株取引や株式ニュースのマスター・アプリケーションなど、多数のマスター・アプリケーションにマップすることが可能です。たとえば、ユーザー A の株を売却するマスター・アプリケーションを起動できる株価通知エンジンは、ユーザー B の特定の株に関する最新のニュースを送信するマスター・アプリケーションの起動にも使用できます。同じエンジンを使用して、簡単なメッセージ・テンプレートをユーザー C に配信することもできます。このように、株取引マスター・アプリケーションは、株価が特定の時間に特定の値に達したかどうかをチェックするマスター通知アプリケーション（時間および値ベース）、または株価が大幅に変動（10% の下落など）した場合に起動する別の通知アプリケーション（値ベース）、あるいはその両方によって起動できます。

OracleAS Wireless Tools が管理できるのは、1 対 *n* のマッピングのみです。これは、ユーザー・インタフェース制約（同じマスター通知アプリケーションを複数のマスター・アプリケーションにマッピングできる）があるためです。ただし、通知 API を使用すると、多数対 *n* のマッピングを実行できます。

11-5 ページの「主な機能」で説明したように、通知エンジンでは、ユーザーごとにターゲット・アプリケーションを起動してパーソナライズ・コンテンツを生成したり、ターゲット・アプリケーションをイベント当たり 1 回起動できるため、そのイベントに関心がある全ユー

ザーがコンテンツを共有できます。マッピングの作成時に、アプリケーション設計者は、適切なコンテンツの生成タイプをユーザー要件に応じて選択する必要があります。

サンプル・コード : 通知マッピング

次のコード部分に、既存のマスター通知アプリケーション（11-12 ページの「API サンプル : マスター通知アプリケーションの作成」で作成した内容）とマスター・アプリケーションとの間のマッピングの作成方法を示します。この例では、マッピングによってパラメータの symbol と price がマップされます。また、コンテンツ生成タイプは personalized content に指定されています。

通知マッピング

```
//Locate the existing master notification service
MasterAlertService mAS = s.lookupMobileMasterAlertService("StockNotification");

//Locate the existing stock trade master service
MasterService masterService =
s.lookupMasterService("/master/examples/StockTradeMasterService");

//Create the mapping definition between the notification service and master service
MAlertServiceMapping map = mAS.createMapping(masterService,
"StockNotificationMapping");

//Retrieve input arguments for the master service
Arguments args = masterService.getInputArguments();
InputArgument inpArgTicker = args.getInput("symbol");
InputArgument inpArgPrice  = args.getInput("price");

//Retrieve notification parameters (including input and output)
AlertParameterMeta[] alertParams = mAS.getParameters(); //returns symbol, price

//Map notification master service parameter "symbol" to stock trade input argument
"ticker"
map.addChainParameter(alertParams[0], inpArgTicker);

//Map notification master service parameter "price" to stock trade input argument
"price"
map.addChainParameter(alertParams[1], inpArgPrice);

//This invocation has to be performed in "personalized content mode", for each user
separately
map.setInvocationType(false);

//Commit changes
mAS.save();
```

サンプル・コード : テンプレート・ベースの通知マッピング

次のコード部分に、StockNotification マスター通知アプリケーションに既存のメッセージ・テンプレートを使用してマッピングを作成する方法を示します。この例では、最小量の情報を使用して、テンプレート・ベースのマッピング用のマスター・アプリケーションを作成する必要があります。パラメータ・マッピングとターゲット URL の設定は、`createTemplateMapping` メソッドで実行します。

テンプレート・ベースの通知マッピング

```
//Locate the existing master notification service
MasterAlertService mAS = s.lookupMobileMasterAlertService("StockNotification");

//Create a simple master service
MasterService templateMasterService = f.createMasterService("StockInfo",
s.lookupUser("orcladmin"), s.lookupAdapter("HttpAdapter"),
s.lookupFolder("/master/examples"));

//Parameter mappings will be handled by the createTemplateMapping method
//this method will also modify the provided master service to use default template
processor
mAS.createTemplateMapping(templateMasterService);

//Commit changes
mAS.save();
```

サブスクリプション

注意 : サブスクリプションは、エンド・ユーザーに表示可能な唯一の手順です。

通知を作成してアプリケーションにマッピングし、1 つのアプリケーションとして公開した後、ユーザーは、この通知対応アプリケーションのサブスクリプションを OracleAS Wireless Portal を使用して開始できます。通知ではなくアプリケーションをサブスクライブする理由は、エンド・ユーザーの関心が、コンテンツの生成方法ではなく、生成されたコンテンツにあるためです。このコンテンツは、任意の通知で起動できるアプリケーションによって生成されているため、エンド・ユーザーは、基礎となるインフラストラクチャを気にすることはありません（またその必要もありません）。

通知の作成プロセスでは、条件情報を指定して構造とメタデータを定義するため、ユーザーは、サブスクリプションの手順で各条件に必要なパラメータ値を指定する必要があります。各条件タイプに、次の情報を設定する必要があります。

- データ・ベース : データ・フィーダに必要な入力パラメータおよびトリガー条件パラメータ。
- 時間ベース :
  - 通知頻度 : 使用可能な値は、daily、weekday、weekend および once です。
  - 通知の受信時 : 1 日の特定の時間（時間と分の情報を含む）または期間。期間には、開始時間（時間と分）、終了時間（時間 / 分）および間隔が必要です。たとえば、午後 3 時から午後 5 時まで 15 分ごと、などと指定します。
  - ブラックアウト期間 : 指定したブラックアウト期間中（2 つの日付の間の期間）、ユーザーは通知を受信しません。これは、ユーザーが休暇中の場合に役に立ちます。
  - 有効期限 : 指定した日付以後、ユーザーは通知を受信しません。UI の複雑性の問題のため、OracleAS Wireless Portal では、有効期限をサポートしていません。
- ロケーション・ベース : ロケーション条件は、ロケーション条件オブジェクトに基づいています。ロケーション条件は、1 つ以上のロケーション基準で構成されています。各基準は、ターゲット、リージョンおよび基準（移動）タイプ別に定義されます。ターゲットは、ロケーション・サーバーが追跡する対象（ユーザー、ユーザー・グループ、携帯電話の番号など）です。基準タイプは、ロケーション・サーバーで監視する必要がある移動のタイプです。たとえば、ユーザーがリージョンに移動する場合やリージョンから移動する場合が対象となります。UI の複雑性の問題のため、OracleAS Wireless Customization Portal では、1 つのロケーション条件に対して複数の基準の作成はサポートしていません。特定のサブスクリプションに対してこのツールで作成できる基準は 1 つのみです。ただし、付属している Java API を使用すると、複数の基準を含めることができます。

この情報を使用して、ユーザーは、通知の受信に使用するデバイスと代替デバイスを選択できます。この代替デバイスは、プライマリ・デバイスが通知の最大数に達した場合に通知を送信します。デバイスの選択はオプションの手順です。

通知エンジンは、ユーザーのサブスクリプション条件を確認し、メッセージ・コンテンツを生成すると、配信のためにそのメッセージを XMS レイヤーに渡します。ユーザーが通知検索用のデバイスを選択している場合、XMS では、その選択されたデバイス（その日の最大数に達している場合は、指定した代替デバイス）を使用して、エンド・ユーザー・メッセージを適切なプロトコルで配信します。ただし、ユーザーがデバイスを指定しない場合、XMS では、連絡ルール、ユーザー・プロファイルおよびメッセージ・コンテンツ・タイプに従って、最適なデバイスを選択します。デバイス選択の詳細は、11-28 ページの「統合された通知ソリューション」を参照してください。

サンプル・コード : サブスクリプションの作成

次のコード部分に、通知対応アプリケーションをサブスクライブする方法を示します。この例のリンクは、11-14 ページの「サンプル・コード : 通知マッピング」で使用した StockTradeMasterService マスター・アプリケーションに基づいています。前述の例では、このマスター・アプリケーションは、StockFeed マスター通知アプリケーションにマップされたことに注意してください。

サブスクリプションの作成

```
//Locate the master notification service
MasterAlertService mAS = s.lookupMobileMasterAlertService("StockNotification");
//Locate the stock trade master service
MasterService masterService =
          s.lookupMasterService("/master/examples/StockTradeMasterService");
//Locate the stock trade link
Link link = s.lookupLink("/Examples/StockTradeLink");
//Locate the user
User user = s.lookupUser("orcladmin");

//Locate primary and alternative device addresses
DeviceAddress addr1 = s.lookupDeviceAddress(DeliveryType.SMS, "1234567890");
DeviceAddress addr2 = s.lookupDeviceAddress(DeliveryType.EMAIL,
          "Okan.Alper@oracle.com");

//Create a subscription for orcladmin on the stock trade link
ServiceAlertSubscription sub = mAS.addUserAlertSubscription(user, link);

//Set the data feed input parameter (ticker) to ORCL
AlertInputParamValue[] paramVals = sub.getInputParameters();
paramVals[0].setValue("ORCL");

//Set triggering condition values: price:30 and change: 12
AlertConditionValue[] conVals = sub.getConditions();
conVals[0].setValue("30"); //price
conVals[1].setValue("12"); //change

//Set frequency to daily, receive notifications on weekdays
AlertTimeFrequency freq = AlertTimeFrequencyImpl.getAlertTimeFrequencys()[0];

//Activate the notification tomorrow
Calendar startDate = Calendar.getInstance();
startDate.add(Calendar.DATE, 1);

//User will be subscribed for 365 days
Calendar expirationDate = Calendar.getInstance();
expirationDate.add(Calendar.DATE, 366);

//User will be going on vacation 30 days from now,
//so deactivate them temporarily in that period for 10 days
Calendar blackoutStartDate = Calendar.getInstance();
blackoutStartDate.add(Calendar.DATE, 30);
Calendar blackoutEndDate = Calendar.getInstance();
blackoutEndDate.add(Calendar.DATE, 40);
```

```
    //Create location condition for monitoring myself for getting into a region with
id 18191.
    LocationPrivacyDomain lbDomain = new LocationPrivacyDomain(masterService);
    LBCondition lbCondition = f.createLBCondition(LBCondition.MODE_REPEAT,
                expirationDate, user, lbDomain);
    lbCondition.addCriteria("orcladmin", "user", "IN", 18191);

    //Set data and time predicates for this subscription.
    //This notification will evaluate the conditions starting at 8:00 a.m. till
1:30pm
    //every 45 minutes, on every weekday (Monday-Friday).
    sub.setCondition(paramVals, conVals, 8, 0, 13, 30, 45, freq,
                 expirationDate, startDate);
    //Set the location condition
    sub.setLocationCondition(lbCondition);

    //Set the primary device that notifications will be send to
    sub.setSubscriptionDevice(addr1);
    //When the max. number of notifications is reached,
    //send the notifications to the alternative device
    sub.setAlternativeType(ServiceAlertSubscription.AFTERMAX_DEVICE);
    //Set the secondary device as alternative device
    sub.setAlternativeDevice(addr2);

    //Set blackout periods, activation/deactivation information
    sub.setSuspendStartDate(blackoutStartDate);
    sub.setSuspendEndDate(blackoutStartDate);
    sub.setStartDate(startDate);
    sub.setExpirationDate(expirationDate);

    //save the subscription
    sub.save();
```

通知の管理

通知エンジンは、独自のリソース（スレッドなど）を管理して、連続モードで実行する必要があります。それによって、着信イベントを処理し、通知の配信に適格なサブスクリプションを決定できます。通知の作成が完了した後、アプリケーション開発者は、通知エンジンに対して個別プロセスを作成し、前もって設計されている適切な通知を添付する必要があります。

通知エンジンは、スケーラブルなシステムとして設計されているため、Oracle Enterprise Manager の一部として、OracleAS Wireless Tools では、1 つの通知を管理するために複数のプロセスを作成できます。この場合、ロード（着信イベント）は、これらのプロセス間に分散されます。各プロセスは、着信イベントを個別に処理します。つまり、プロセスごとに受信した特定のイベントのフィルタ処理が実行されます。

リソースをあまり消費しない通知の場合は、1 つのプロセス内の様々な通知間でリソースを共有することも可能です。この場合、アプリケーション設計者は、プロセスを作成して複数の通知を追加する必要があります。

前述のとおり、データ・ベースの通知は、データ・フィーダが提供する着信データ・イベントに依存しています。したがって、通知エンジンにデータ・イベントを提供するデータ・フィーダ・インスタンスに対しても、個別のデータ・フィーダ・プロセスを作成し、起動する必要があります。

通知の移行

9.0.2.*x* の通知（以後アラートと呼びます）は、付属している `migrateNotifications.sh`*[.bat]* スクリプトを実行して、現行のリリースに移行できます。移行の手順は、次のとおりです。

1. $ORACLE_HOME/wireless/bin/ にナビゲートします。
2. 次のいずれかを実行します。
   a. `migrateNotifications.sh`*[.bat]* `name` <*deprecated master alert name(s)*> `-owner` <*owner username*> を実行します。

      name パラメータを使用して、アラート（移行対象の）を名前で検索します。

      <deprecated master alert name(s)> には、% などのワイルド・カードを使用できます。9.0.4.*x* の通知関連オブジェクト（マスター通知アプリケーション、マスター・アプリケーション、アプリケーション・リンクなど）はすべて、指定したユーザー名で所有されます。

   b. 次のように、同じスクリプトを `-oid` オプション付きで実行することもできます。

      ```
      migrateNotifications.sh[.bat] -oid <deprecated master alert oid> -owner
      <owner use name>
      ```

      OID オプションを使用し、オブジェクト ID で特定の通知を検索します。

前述の（a）または（b）を実行すると、次の操作が実行されます。

- 新規のマスター通知アプリケーションが作成されます。名前は、*<old master alert name>_New* となります。このプロセスでは、必要に応じてメッセージ・テンプレートが有効な Mobile XML に変換されます。
- このフォルダが存在しない場合は、新規フォルダがマスター・アプリケーションの /master/notifications として作成されます。
- 新規のマスターアプリケーションが作成されます。名前は、*<old master alert name>_MS* となります。
- 古いマスター・アラートのメッセージ・テンプレートに基づいて、新規のマスター通知とマスター・アプリケーションのマッピングが作成されます。
- このフォルダが存在しない場合は、リンク用の新規フォルダが */Users Home/<username>/notifications* として作成されます。
- 関連する 9.0.2.X AlertService（通知アプリケーション）オブジェクトがすべて検出され、リンク・オブジェクトに変換されます。このプロセスの実行時に、トピック・レベルの認可がリンク・レベルの認可にフラット化されます。
- 前項で変換されたアラート・アプリケーション用に、すべてのサブスクリプションが変換されます。

サンプル使用方法

次に、通知移行のサンプル使用方法を示します。

UNIX:

```
migrateNotifications.sh  -name StockAlert% -owner orcladmin
```

名前が StockAlert（StockAlertNews や StockAlertWarning など）で始まる 9.0.2.X マスター・アラート・アプリケーションをすべて移行します。新規オブジェクトはすべて、orcladmin ユーザーによって所有されます。

```
migrateNotification.sh -name StockAlert -owner systemadmin
```

StockAlert の名前を持つ 9.0.2.X マスター・アラート・アプリケーションを移行します。新規オブジェクトをすべて systemadmin ユーザーに割り当てます。

WINDOWS:

```
migrateNotification.sh -oid 1973 -owner systemadmin
```

ID が 1973 の 9.0.2.X マスター・アラート・アプリケーションを移行します。新規オブジェクトをすべて systemadmin ユーザーに割り当てます。

データ・フィーダ

データ・フィーダは、コンテンツをダウンロードするエージェントです。データ・フィーダは、アプリケーションの起動に関係なく定期的に動作します。フィード・フレームワークは、OracleAS Wireless プロセス用にコンテンツをダウンロードするように設計されています。ダウンロードされたコンテンツは、非同期通知と、同期アプリケーション用にキャッシュされたデータの両方に使用できます。

データ・フィーダのダウンロード・スケジュールは、そのデータ・フィーダの更新ポリシーでメンテナンスされます。更新ポリシーにより、更新間隔、つまりデータ・フィーダの実行頻度が決定されます。更新ポリシーでは、データ・フィーダを実行する時刻と曜日が追跡されます。

各データ・フィーダには、コンテンツのソースとなるコンテンツ・プロバイダがあります。コンテンツ・プロバイダでは、コンテンツの URI、コンテンツのダウンロードに使用するプロトコルおよびダウンロードするデータのフォーマットに関する情報がメンテナンスされます。

フィードの指定時に、ユーザーはフィード・パラメータを使用して、ダウンロードするコンテンツのメタデータ定義をセットアップします。これらのパラメータは、データ型 FeedMetaData のインスタンスです。フィード・パラメータには基礎となる SQL データ型があり、`oracle.panama.feed.FeedUtil` に定義されている定義済の型セットから選択されます。

フィードの入力パラメータは、コンテンツ・プロバイダ固有です。この入力パラメータでは、コンテンツ・プロバイダからのデータをリクエストするときに使用するデータを指定します。たとえば、HTTP を使用してコンテンツ・プロバイダからデータをダウンロードする場合、入力パラメータは `GET URL` の作成に使用されるか、HTTP リクエストで `POST` パラメータとして使用されます。

フィードの出力パラメータでは、コンテンツ・プロバイダからの出力のデータ型を定義します。

データ・フィーダのランタイム動作は、`FeedDownloadHook` および `FeedDataFilterHook` を使用してカスタマイズできます。

FeedDownloadHook は、コンテンツのダウンロード用 URI のカスタマイズに使用します。たとえば、HTTP によるダウンロードで、入力パラメータは `GET URL` の作成に、`GET HTTP` パラメータとして使用される入力パラメータとともにデフォルトで使用されます。ただし、ベース URL が入力パラメータに依存する場合があります。その場合の URL は、http://www.ahost.com/input_param_1/input_param2/index.html となります。URL の作成動作をカスタムの FeedDownloadHook でオーバーライドして、必要な結果を得ることができます。

FeedDataFilterHook は、ダウンロードしたコンテンツに対する追加の処理に使用されます。各データ行がダウンロードされるたびに、各行でデータ・フィルタ・フックが起動されます。これにより、フィード実装者は、1 つの出力パラメータを複数に分割するなどの特殊な処理を実行できます。

パススルー・データ・フィーダは、ユーザー定義の Java コードを介してローカル・コンテンツにアクセスするデータ・フィーダです。そのため、パススルー・データ・フィーダには、コンテンツ・プロバイダも更新ポリシーもありません。また、FeedDownloadHook と FeedDataFilterhook も関連しません。パススルー・データ・フィーダの場合も、フィードのメタデータはセットアップする必要があります。

注意： 従来、データ・フィーダはリクエスト / 返信（データ・プル・フィード）を実行するために設計されました。アーキテクチャはプッシュ・データ・フィードに対応するように設計されていますが、この機能はこのバージョンには組み込まれていません。

データ・フィーダの作成

データ・フィーダは、OracleAS Wireless Tools を使用するか、またはプログラムによって作成できます。OracleAS Wireless Tools にはウィザードが用意されており、各ステップに表示される指示に従ってデータ・フィーダを作成できます。詳細は、『Oracle Application Server Wireless 管理者ガイド』を参照してください。

データ・フィーダの作成作業には、次の手順が含まれます。

1. 名前付きデータ・フィーダの作成 : すべてのデータ・フィーダに名前を付ける必要があります。この名前は変更できます。また、データ・フィーダには、永続的な一意のオブジェクト ID もあります。これは作成時にシステムによって生成されます。
2. コンテンツ・プロバイダのパラメータの設定 : 現行のコンテンツ・プロバイダ用のプロトコルとフォーマットを設定します。組込みのプロトコルとフォーマット用の定数があります。
3. データ・フィーダの入力パラメータの作成 : データ・フィーダには、1 つ以上の入力パラメータが必要です。指定する入力パラメータごとに、内部名とデータ型を指定する必要があります。パラメータには、選択したフォーマットに応じて異なるオプションを使用できます。選択したフォーマットが区切り記号付きテキストの場合は、入力パラメータが表示される列の番号を指定するオプションがあります。これは、入力パラメータがコンテンツ・プロバイダからの出力にも含まれる場合に役立ちます。列の索引は、SQL と同様に 1 から始まります。0（ゼロ）を指定すると、入力パラメータは出力に含まれないものとみなされます。
4. データ・フィーダの出力パラメータの作成 : データ・フィーダには、1 つ以上の出力パラメータが必要です。出力パラメータは、入力パラメータと同じ方法でカスタマイズできます。
5. フィードのファイナライズ : 最後に、DataFeeder のメソッド `createFeedDefinition` をコールする必要があります。このメソッドは、フィードとフィード・キャッシュ表を使用するために必要なフィード・メタデータ定義を、リポジトリに作成します。フィード定義の作成後は、フィード・パラメータを削除できません。できるのは名前の変更のみです。

パススルー・データ・フィーダの作成

パススルー・データ・フィーダの場合は、使用するフィーダのクラス名を指定する必要があります。通常のデータ・フィーダに必要な情報の一部は不要です。特にプロトコルと使用するフォーマットは関連しません。

次のコードでは、パススルー・データ・フィーダが作成されます。

```
ModelFactory mf = MetaLocator.getInstance().getModelFactory();
// Create a named datafeeder
PassthroughDataFeeder df = mf. createPassThroughDataFeeder ("stock_passthrough")
// Set the class name to use for implementation
df.setClassName("fully.qualified.package.and.Class");
// Create input parameters
FeedMetaData fmi = df.createMetaData("sym", "TEXT_30");
df.addInputParameter(fmi):
// Create output parameters
FeedMetaData fmo1 = df.createMetaData("price", "NUMBER");
df.addOutputParameter(fmo1);

FeedMetaData fmo2 = df.createMetaData("change", "NUMBER");
df.addOutputParameter(fmo2);

// Finalize the feed -- create feed definition
// in repository
df.createFeedDefinition();
```

サンプル・アプリケーション

サンプル・アプリケーション: XML による株価のダウンロード

OracleAS Wireless には、`sample200.xml` が含まれています。このサンプル・ファイルには、HTTP を介して株価を取り出すためのデータ・フィーダが含まれています。株価は XML フォーマットで、サンプル・データ・フィーダには、XML 入力フィードから関連する値を抽出するためのスタイルシートが含まれています。

このデータ・フィーダをプログラムによって作成するには、次のコードを使用します。

```
ModelFactory mf = MetaLocator.getInstance().getModelFactory();
// Create a named datafeeder
DataFeeder df = mf.createDataFeeder("stock_screamingmedia");
// Set content provider parameters
ContentProviderInfo cpi = df.getContentProviderInfo();
cpi.setProtoolType(ContentProviderInfo.PROTOCOL_HTTP);
cpi.setPrimarySource("http://www.screamingmedia.com/");
cpi.setFormatType(ContentProviderInfo.FORMAT_XML);
// Create input parameters
```

```
FeedMetaData fmi = df.createMetaData("sym", "TEXT_30");
df.addInputParameter(fmi):
// Set the parameters for this parameter and content provider
Map paramOptions = new Hashtable();
paramOptions.put(ContentProviderInfo.COLUMN_NUMBER, 1);
cpi.setParamArguments(fmi, paramOptions);


// Create output parameters
FeedMetaData fmo1 = df.createMetaData("price", "NUMBER");
df.addOutputParameter(fmo1);
paramOptions.put(ContentProviderInfo.COLUMN_NUMBER, 2);
cpi.setParamArguments(fmo1, paramOptions);

FeedMetaData fmo2 = df.createMetaData("change", "NUMBER");
df.addOutputParameter(fmo2);
paramOptions.put(ContentProviderInfo.COLUMN_NUMBER, 3);
cpi.setParamArguments(fmo2, paramOptions);
// Finalize the feed -- create feed definition in repository
// create cache table as needed
df.createFeedDefinition();
```

サンプル・アプリケーション : CSV フォーマットによる株価のダウンロード

sample200.xml には、HTTP を介して株価を取り出すためのデータ・フィーダも含まれています。このデータ・フィーダでは、株価が comma-separated variable（CSV）フォーマットでダウンロードされます。

次のコードに、このデータ・フィーダをプログラムによって作成する方法を示します。

```
ModelFactory mf = MetaLocator.getInstance().getModelFactory();
// Create a named datafeeder
DataFeeder df = mf.createDataFeeder("stock_yahoo")
// Set content provider parameters
ContentProviderInfo cpi = df.getContentProviderInfo();
cpi.setProtocolType(ContentProviderInfo.PROTOCOL_HTTP);
cpi.setPrimarySource("http://quotes.yahoo.com/quote");
cpi.setFormatType(ContentProviderInfo.FORMAT_DELIMITED);
// Create input parameters
FeedMetaData fmi = df.createMetaData("sym", "TEXT_30");
df.addInputParameter(fmi):
// Set the parameters for this parameter and
// content provider
Map paramOptions = new Hashtable();
paramOptions.put(ContentProviderInfo.COLUMN_NUMBER, 1);
cpi.setParamArguments(fmi, paramOptions);
```

```
// Create output parameters
FeedMetaData fmo1 = df.createMetaData("price", "NUMBER");
df.addOutputParameter(fmo1);
paramOptions.put(ContentProviderInfo.COLUMN_NUMBER, 2);
cpi.setParamArguments(fmo1, paramOptions);

FeedMetaData fmo2 = df.createMetaData("change", "NUMBER");
df.addOutputParameter(fmo2);
paramOptions.put(ContentProviderInfo.COLUMN_NUMBER, 3);
cpi.setParamArguments(fmo2, paramOptions);

// Finalize the feed -- create feed definition
// in repository, create cache table as needed
df.createFeedDefinition();
```

フィードへの入力パラメータ値の追加

データ・フィーダでは、入力パラメータが指定されているコンテンツのみがダウンロードされます。入力パラメータの値は、暗黙的に、またはプログラムによって設定できます。通知トピックのサブスクリプションを追加すると、入力値を暗黙的に追加できます。次のコードに、通知をプログラムによって追加する方法を示します。

```
// Look up existing data feeder
DataFeeder df = ModelServices.getInstance().lookupDataFeeder("stock_yahoo");
// Want to add input params for ORCL
Map params = new Hashtable();
params.put("sym", "ORCL");
df.setData(params);
```

ダウンロードされた値の取出し

データ・フィーダの主な用途の 1 つは、キャッシュされたデータを通常の同期アプリケーションに使用できるようにダウンロードすることです。ダウンロードされたデータには、データ・フィーダ・メソッド `getData()` を使用してアクセスできます。このメソッドは、引数として Map を取ります。この引数は、値を取得するパラメータの名前 / 値のマッピングです。次のサンプル・コードに、現在の株価を取り出して銘柄コードを変更する方法を示します。

```
ModelServices ms = MetaLocator.getInstance().getModelServices();
DataFeeder df = ms.lookupDataFeeder("stock_yahoo");
Map params = new Hashtable();
params.put("sym", "ORCL");
Map values = df.getData(params);
Iterator i = values.keys();
while(i.hasMore()) {
String key = (String)i.next();
String val = (String)values.get(key);
```

```
System.out.println(key + " = " + val);
}
Running this code we while get the following output:
sym = ORCL
price = 18.75
change = 0.5
```

データ・フィーダ・プロセスの起動

システム管理者が、データ・フィーダ・プロセスを起動します。他のプロセスと同様に、データ・フィーダを実行するには、システム管理者がそのプロセスをセットアップする必要があります。詳細は、『Oracle Application Server Wireless 管理者ガイド』を参照してください。

注意： データ・フィーダでは、入力パラメータの値が指定済で、通知アプリケーションが作成され、その通知のサブスクリプションがある場合のみ、コンテンツがダウンロードされます。

フィード・パラメータの外部名

外部名とは、コンテンツ・プロバイダからコンテンツを取り出すときに使用される名前です。このメカニズムは、別のコンテンツ・プロバイダに変更するなど、フィードの作成後にパラメータ名の外部表現に変更があった場合に備えることを意図しています。外部名はオプションであり、指定しなければ内部名が使用されます。

入力パラメータに使用するキャプションを指定します。これは、あくまでもドキュメント用です。

入力パラメータが定義されていても、コンテンツを取り出すときには関係しない場合があります。外部パラメータ名に特殊な定数 `__NONE__` を使用すると、`download URL` または `POST` リクエストの作成時に、入力パラメータが無視されます。

フィードのスケジューリング

デフォルトでは、フィードは起動後は継続的に実行されます。各フィードには更新ポリシーが関連付けられています。このポリシーを使用して、フィードの開始時刻と終了時刻、実行する曜日および実行間隔など、フィードの実行を微調整できます。

次のコードでは、サンプル・データ・フィーダを平日の午前 9 時から午後 5 時まで実行するように、更新ポリシーを設定します。

```
ModelServices ms = MetaLocator.getInstance().getModelServices();
DataFeeder    df = ms.lookupDataFeeder("stock_yahoo");
UpdatePolicy  up = df.getUpdatePolicy();
up.setStartTime(9,0,0);
up.setEndTime(17,0,0);
up.setUpdateDays(UPDATE_WORKDAYS);
```

```
// Set update interval to 300 seconds, i.e. update every
// 5 minutes
up.setUpdateInterval(300);
```

XML データのフィード

XML コンテンツを伴うデータ・フィードにアクセスする場合は、入力 XML を共通の XML フォーマットに変換する XSLT スタイルシートを指定する必要があります。

共通の XML フォーマットは、フィード結果（<omfeed_result>）で構成されており、フィード結果にはデータ行数（0 以上）（<omfeed_datarow>）があり、各行は 1 つ以上の名前付きデータ列（<omfeed_datacolumn>）で構成されています。データ列の名前は、フィードに定義されたパラメータと一致します。各出力パラメータには、対応するデータ列が必要です。次のサンプル・コードに、株価フィードの出力を示します。

```
<?xml version="1.0"?>
<market-data>
<quote-set>
<quote symbol="ORCL" name="ORACLE CORPORATION" type="stock" exchange-code="NASDAQ"
last="32.000000" close="28.562500" close-flag="closed" change="3.4375"
percent-change="12.04%" volume="56362800" open="30.0" high="32.4375" low="29.9375"
bid="32.0" ask="32.0625" bid-size="36" ask-size="90" high-52-week="46.468998"
low-52-week="15.438" shares-outstanding="5629833" pe-ratio="29.299999"
volatlity="16.150000" yield="0.000000" earnings-per-share="1.092000" status="ok"/>
</quote-set>
</market-data>
The stylesheet for transforming this result would then look like this:
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="quote-set">
    <omfeed_result>
      <xsl:for-each select="quote">
        <omfeed_datarow>
            <omfeed_datacolumn>
              <xsl:attribute name="name">sym</xsl:attribute>
              <xsl:value-of select="@symbol"/>
            </omfeed_datacolumn>
            <omfeed_datacolumn>
              <xsl:attribute name="name">price</xsl:attribute>
               <xsl:value-of select="@last"/>
            </omfeed_datacolumn>
            <omfeed_datacolumn>
              <xsl:attribute name="name">change</xsl:attribute>
              <xsl:value-of select="@change"/>
            </omfeed_datacolumn>
        </omfeed_datarow>
      </xsl:for-each>
    </omfeed_result>
  </xsl:template>
```

統合された通知ソリューション

ソフトウェア・アプリケーションの多くは、アプリケーション内で発生している特定のイベントをユーザーに通知する機能を提供しています。たとえば、カレンダ・システムによって、会議に召集されていることをユーザーに通知したり、予定されていた会議の場所が変更されたことを通知できます。多くのアプリケーションでは、ユーザーにとって関心のあるイベント・タイプを提示できます。これによって、ユーザーは、緊急と思われるイベントのみに通知を制限できます。これらのイベント通知は通常電子メールで送信されます。

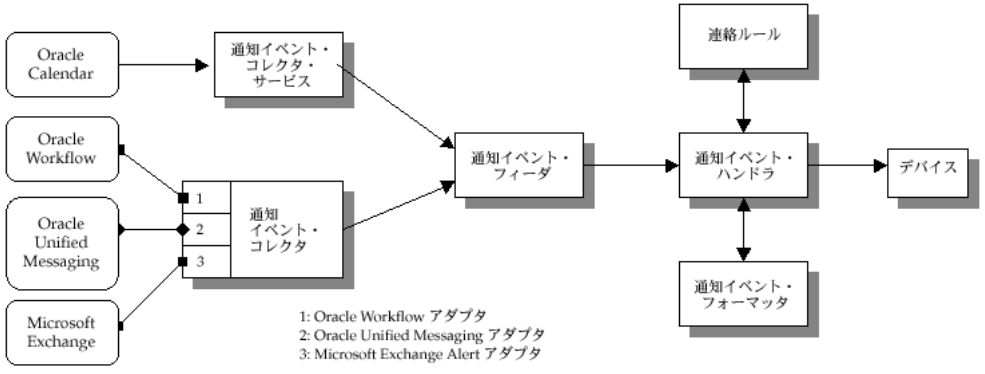
通知エンジンに基づいて、OracleAS Wireless には、Oracle Calendar、Oracle Unified Messaging、Oracle Workflow、Microsoft Exchange Email など、多数のアプリケーションに対応するデフォルトのマルチチャネル通知ソリューションが組み込まれています。また、アプリケーションの通知を、特定のデバイス・タイプ用にカスタマイズされたコンテンツとともに、ユーザーの任意のワイヤレス・デバイスに配信するインフラストラクチャも用意されています。使用しているデバイス・タイプに応じて、同一イベントに対して異なる通知を受信できます。音声ベースの通知では、適切な文とダイアログが使用されます。SMS ベースの通知では、SMS メッセージのサイズ制限が考慮されます。

次の項では、アプリケーション通知と通知エンジンの統合アーキテクチャについて説明します。また、Oracle Workflow と Microsoft Exchange の 2 つの統合についても説明します。

通知エンジンの統合

アプリケーションのイベント通知プロセスでは、OracleAS Wireless の通知エンジンを使用してワイヤレス・デバイスに通知を配信します。アプリケーション・イベントの収集、ユーザーの連絡ルールの処理および通知コンテンツのフォーマットに必要なコンポーネントも追加されます。次に、通知プロセスの多様なコンポーネントに関するアーキテクチャ上の概要を示します。

図 11-3 統合された通知ソリューション



OracleAS Wireless 以外のアプリケーションは、2 つの異なるメカニズムを使用して通知エンジンとインタフェースできます。

最初のメカニズムは、プッシュ・インタフェースです。アプリケーションは HTTP を介して、サーブレットに基づく通知イベント・コレクタに通知イベントを送信します。通知イベント・コレクタは、通知イベント・データを通知イベント・フィーダに渡します。このフィーダは、通知エンジン用にカスタマイズされたデータ・フィーダです。

2 番目のメカニズムは、プル・インタフェースです。通知イベント・コレクタ・プロセスは、アプリケーションに接続して、通知イベントを取得します。通知イベント・データは、次に通知イベント・フィーダに渡されます。通知イベント・コレクタ・プロセスは、多数の異なるアダプタで構成されています。各アダプタは、特定のアプリケーションに固有です。通知コレクタ・プロセスを構成することで、アダプタを有効化および無効化できます。通知コレクタ・プロセスの作成、開始、停止または構成には、Enterprise Manager コンソールを使用します。

通知イベント・ハンドラはカスタマイズされたシステム・レベルの通知アプリケーションで、通知イベント・フィーダからデータを読み込みます。このデータは、この通知のターゲット・ユーザー、通知タイプおよび他の通知固有のデータを示します。

次に、通知イベント・ハンドラは、ターゲット・ユーザーのアクティブな連絡ルールを参照し、ユーザーの優先通知デバイスのタイプとアドレスを判別します。通知イベント・フォーマッタが起動されます。これによって、ユーザーのデバイス・タイプ用にカスタマイズされた通知のコンテンツが生成されます。生成された通知コンテンツは、通知エンジンによってユーザーのデバイスに配信されます。

通知イベント・ハンドラは、システム・レベルの通知アプリケーションです。このプロセスでは、通知を受信するために、ユーザーが通知サブスクリプションを明示的に作成する必要はありません。かわりに、ユーザー ORCLADMIN のみがこのプロセスをサブスクライブします。アプリケーションに応じて、ユーザーは通知を受信するイベントを（Customization Portal または実際のアプリケーション自体に）指定できます。処理済の各通知について、システムでは、ターゲット・ユーザーの連絡ルールを参照し、正しいユーザーが通知を受信することを確認します。通知イベント・プロセスの開始、停止または構成には、Enterprise Manager コンソールを使用します。Oracle Enterprise Manager を使用してプロセスを開始または停止する方法は、『Oracle Application Server Wireless 管理者ガイド』を参照してください。

注意： 通知イベント・ハンドラの通知アプリケーションをサブスクライブする必要があるのは、ORCLADMIN ユーザーのみです。複数のサブスクリプションが存在する場合、ユーザーは各通知について複数（サブスクリプションと同じ数）のコピーを受信します。

通知イベント・コレクタと通知イベント・ハンドラは、2 つの個別のプロセスです。システムがアプリケーション・イベント通知を処理するためには、両プロセスが同時に稼働している必要があります。

ワークフローの統合

Oracle Workflow の統合には 2 つのコンポーネントが含まれます。1 つは、Oracle Workflow の通知キューから通知を受信し、それをユーザーのモバイル・デバイスに送信する通知アプリケーションです。もう 1 つのコンポーネントは、Oracle Workflow の通知ワークリスト・アプリケーションです。このアプリケーションには、OracleAS Wireless Portal を介してアクセスできます。

Oracle Workflow および OracleAS Wireless は両方とも Oracle Application Server のコンポーネントです。そのため OracleAS Wireless には、OID を介して Oracle Workflow に接続する機能があります。OracleAS Wireless は、OID を介して Oracle Workflow に接続するため、同じユーザー・リポジトリが共有されます。

通知アプリケーション

Oracle Workflow には、特定のインスタンスに関する送信通知すべてを格納するキューがあります。キューの各メッセージには、通知とその送信先ユーザーに関する必要情報がすべて格納されています。OracleAS Wireless は、これらのメッセージをデキューし、エンド・ユーザーに、XMS を使用して送信するメッセージを作成します。ユーザーはこの通知にレスポンスできます。レスポンスは、OracleAS Wireless アプリケーションに送られます。アプリケーションでは、ユーザーのレスポンスに応じて Oracle Workflow を更新します。

注意： Wireless と Oracle Workflow との統合のテスト時に、エンド・ユーザーが通知を受信できなかった場合は、ログ・ファイルで ORA-4031 エラーをチェックする必要があります。このエラーは、データベースのメモリー・プール・サイズが不足しているため、通知サービスに失敗したことを示します。共有メモリー・プールのサイズを増加する手順は、次のとおりです。

1. init.ora ファイルにある *shared_pool_size* パラメータの値を大きくします（init.ora ファイルは、通常インフラストラクチャ・マシンの $ORACLE_HOME/dbs ディレクトリにあります）。
2. データベースを再起動して、変更内容を有効にします。

それでもエンド・ユーザーが通知を受信できない場合は、共有メモリー・プールのサイズをさらに増加する必要があります。

ワークリスト・アプリケーション

これは、OracleAS Wireless Portal を介した Oracle Workflow の通知ワークリストに相当します。ワークリスト・アプリケーションは、OID を使用して Workflow に接続し、ユーザーのオープン通知すべてのリストを取得します。各通知は、（通知タイプに応じて）閉じることも、レスポンスすることもできます。

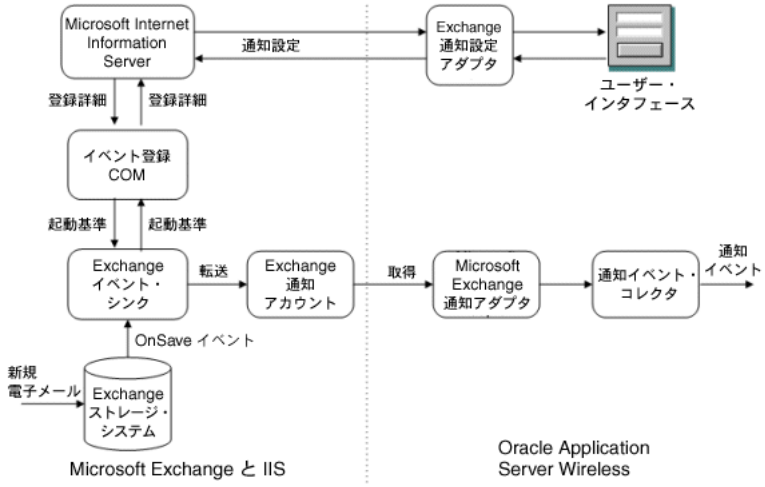
> **関連項目：** Oracle Workflow および OracleAS Wireless の詳細は、『Oracle Workflow 管理者ガイド』および『Oracle Workflow 開発者ガイド』を参照してください。

Microsoft Exchange の通知統合

OracleAS Wireless には、前述のアーキテクチャに基づいて、Microsoft Exchange Email Server 用のマルチチャネル通知機能が用意されています。Microsoft Exchange Email アカウントを持つ OracleAS Wireless のユーザーは、緊急の電子メールまたは指定した人からの電子メール（あるいはその両方）を受信した場合、使用しているワイヤレス・デバイスで通知メッセージを受信できます。

次に、Microsoft Exchange の通知統合で使用する様々なコンポーネントのアーキテクチャの概要を示します。

図 11-4 Microsoft Exchange の通知統合



Microsoft Exchange の通知は、プル・インタフェースを介して統合されます。Exchange Server には、特殊な Microsoft COM オブジェクトがデプロイされています。この COM は、Exchange Server のストア・イベントを傍受し、ユーザーのイベント・サブスクリプション

に基づいて通知イベントを作成します。この通知イベントは、電子メール形式で特殊な Exchange の通知アカウントに配信されます。Microsoft Exchange のアダプタは、通知イベント・コレクタ・プロセスで、これらの通知イベント電子メールを標準の IMAP または POP3 プロトコルを介して取得します。

ユーザーは、OracleAS Wireless Tools を使用して通知サブスクリプションを作成します。サブスクリプション・データは、HTTP/ASP を介して Microsoft Exchange Server のホストに転送され、Microsoft Exchange のストア・イベント・パラメータとして保存されます。

Microsoft Exchange Server と OracleAS Wireless の通知構成の詳細は、『Oracle Applicatioin Server Wireless 管理者ガイド』を参照してください。

通知システムの移行

この項は、9.0.2.x バージョンの通知システムから現行バージョンに移行するユーザーを対象にしています。移行には、両バージョンの通知システムに関する基本的な理解が必要です。この移行を実行するために必要な手順を詳しく説明します。このリリースまたは前のリリースの通知システムの詳細は、該当する開発者ガイドを参照してください。

このリリースでは、9.0.2.x の通知およびその API は削除されています。オラクル社では、9.0.2.x の通知と 9.0.4.x の通知を混合しないことをお薦めします。特定の時間に使用するのは、9.0.2.x の通知のみか、9.0.4.x の通知のみに限定してください。

以前に 9.0.2.x の通知を使用していない場合は、使用している OracleAS Wireless インスタンスを現行のリリースに完全に移行した後で、9.0.4.x を開始してください。

使用しているシステム上に既存の 9.0.2.x 通知が存在する場合、その通知は、OracleAS Wireless 9.0.2.x と 9.0.4.x が混在する環境でのみ、継続して使用してください。使用している OracleAS Wireless インスタンスを 9.0.4.x に移行した後で、通知を 9.0.4.x スタイルにアップグレードし、9.0.4.x 通知のみを使用して起動します。スクリプト（製品の一部として付属）を使用すると、9.0.2.x の通知オブジェクトを 9.0.4.x 準拠の通知にアップグレードできます。このスクリプトについては、この項の後半で説明します。

注意： 9.0.2.x API を使用して通知を操作するアプリケーション（サブスクリプション・ポータルなど）がある場合、これらのアプリケーションは、9.0.4.x の通知 API を使用するためにリライトする必要があります。この項の後半に例を示します。

通知移行の例

次に、一般的な通知移行の例を示します。

1. 純粋な 9.0.2.x 環境で起動します。
2. アップグレードを開始します。この段階では、OracleAS Wireless 9.0.2.x 環境と 9.0.4.x 環境が混在しています。この段階では、9.0.2.x スタイルの通知のみを使用します。
3. OracleAS Wireless 9.0.4.x 環境へのアップグレードを完了します。
4. スクリプト `migrateNotifications` を実行して、9.0.2.x スタイルの通知を 9.0.4.x スタイルにアップグレードします。
5. この時点からは、9.0.4.x スタイルの通知のみを使用してください。このスクリプトについては、この項の後半で詳しく説明します。

構造上の変更

このリリースでは、通知の構造が大幅に変更されています。主な相違点は次のとおりです。

- イベントの生成
- メッセージ・コンテンツの生成
- 認可

イベントの生成

従来は、通知のタイプは、値ベースと時間ベースの 2 つのみでした。

値ベースの通知は、通知エンジンがデータ・フィーダ・コンポーネントからデータ・プッシュ・イベントを受信するたびに評価されます。

時間ベースの通知は、もう少し複雑です。

時間ベースの通知は、特定の時間（通知ユーザーがサブスクリプション時に指定した時間）に評価されました。すべての通知は、時間ベースか値ベースかに関係なく、データ・フィーダに基づいていました（9.0.2.x）。9.0.2.x のタイマー・イベントは、この指定された時間にデータを取得します。さらに、この時間ベースの通知は、通知システムがデータ・フィーダからデータ・プッシュ・イベントを受信するたびに評価されます。たとえば、開発者が、株価フィード・データ・フィーダで作成される、時間ベースの通知を設計するとします。この株価フィード・データ・フィーダには、入力パラメータ *TICKER* が 1 つと出力パラメータ *PRICE* が 1 つあります。ユーザーがこの通知を毎日午前 10 時にサブスクライブすると、通知システムは午前 10 時に通知を生成しますが、さらに、通知エンジンがデータ・フィーダからデータを受信するたびに通知を評価します。したがって、ユーザーが午前 10 時に通知の受信をサブスクライブしている場合でも、そのユーザーは、値トリガー条件が確認されるたびに、通知を受信することになります。

このリリースでは、時間ベースの通知が拡張され、指定時間以外の通知の受信を回避できます。このタイプの時間ベースの通知を完全時間ベースと呼びます。開発者がこのパラメータを指定しない場合、すべての新規通知は、この完全時間ベースにデフォルト設定されます。

このリリースでは、データ・フィーダなしで通知を設計することも可能です。このタイプの通知は、ユーザーが簡単な通知メッセージを特定の時間に送信する場合、またはメッセージ生成メカニズムを使用して関連コンテンツを取得する場合に使用できます。たとえば、開発者がカレンダ通知を設計する場合は、次の 2 つの選択肢があります。

- カレンダ情報を通知システムにプッシュするデータ・フィーダを設計します。この場合、開発者はカレンダ・データ・フィーダに基づいて時間ベースのマスター通知を作成できます。このフィーダは、データに含まれるコンテンツとデータを使用してメッセージを生成します。
- データ・フィーダを使用せずに、時間ベースのマスター通知を設計します。コンテンツの生成時に、通知システムは、開発者が指定した OracleAS Wireless アプリケーションをコールします。この OracleAS Wireless アプリケーションでは、カレンダ・システムに接続して、関連情報を取得し、コンテンツを生成できます。

9.0.4.x では、ロケーション・サーバーを利用して、ロケーション・ベースの通知を設計することもできます。

メッセージ・コンテンツの生成

9.0.2.x では、簡単なメッセージ・テンプレートによって、メッセージ・コンテンツを生成できます。このコンテンツは、次にエンド・ユーザーにプッシュされます。また、9.0.2.x の通知は、フック（Java クラス）を起動して、より複雑なケースのメッセージ・コンテンツも生成できます。

このリリースでは、メッセージ・コンテンツを生成する唯一のメカニズムは、OracleAS Wireless アプリケーションの起動です。つまり、コンテンツを生成するには、通常 OracleAS Wireless アプリケーションを起動します。このため、すべての通知を OracleAS Wireless のマスター・アプリケーションにマップする必要があります。つまり、9.0.4.x では、開発者は、マスター・アプリケーションを通知に対応させる必要があります。この機能によって、コードのレプリケーションが回避されます。これは、同じマスター・アプリケーションを通知と通常のデバイス・アクセスの両方で使用できるためです。さらに、ユーザー・サブスクリプションをモバイル・ポータル・アプリケーション・ツリーを使用して処理できます。

このリリースでも、シード済の OracleAS Wireless アプリケーションを利用することで、メッセージ・テンプレートを使用することは可能です。このアプリケーションでは、このメッセージ・テンプレートにアクセスして処理し、エンド・ユーザー・コンテンツを生成できます。ただし、開発者は、マスター・アプリケーションを作成し、シード済の OracleAS Wireless アプリケーションとマスター通知の間のマッピング情報を定義する必要があります。この場合、マスター・アプリケーションの作成プロセスは自動化されているため、手動による手順（および API コール）はほとんどありません。

認可

従来、通知の認可はトピックとアラート・サービスを使用して実行していました。マスター通知を設計した後、開発者は AlertServices/Topics を作成して、適切なユーザーに割り当て、認可を制御する必要がありました。

このリリースでは、メッセージの生成は通常のマスター・アプリケーションによって実行されるため、通知システムでは、通常のリンク・オブジェクトとフォルダ・オブジェクトを利用して認可を指定します。

移行の制限

9.0.4.x にアップグレードした後は、9.0.4.x の通知と 9.0.2.x の通知を混合しないでください。9.0.4.x の通知に移行する準備ができるまでは、9.0.2.x の通知のみを使用し、移行が完了した後は、9.0.4.X の通知のみを使用してください。

9.0.4.x の OracleAS Wireless Webtool ユーティリティは、9.0.4.x の通知のみを表示、作成するようにデフォルト設定されています。同様に、9.0.4.x の OracleAS Wireless Mobile Portal ユーティリティで処理できるのは、9.0.4.x のサブスクリプションのみです。9.0.4.x の OracleAS Wireless Webtool を使用して、9.0.2.x の通知を処理する必要がある場合は、IASW_HOME/wireless/server/classes/oracle/panama/core/admin ディレクトリにある `System.properties` ファイルを次のように変更する必要があります。

- 標準のテキスト・エディタを使用して、IASW_HOME/wireless/server/classes/oracle/panama/core/admin/System.properties を編集します。
- `DeprecatedAlertSupport` という名前のパラメータを検索し、その値を true に変更します。
- このファイルを保存し、使用している OC4J Portal のインスタンスを再起動します。

9.0.2.x と 9.0.4.x では、サブスクリプション・プロセスと認可が完全に異なるため、OracleAS Wireless Mobile Portal を使用して 9.0.2.x のサブスクリプションを処理することはできません。これを実行するためには、9.0.2.x Customization Portal ユーティリティを有効化するか、独自のサブスクリプション・メカニズムを開発する必要があります。この項の終わりに、既存のコードを移行して、サブスクリプションを処理する方法を示すサンプル・コードを示します。

移行スクリプトの実行

9.0.2.x の通知は、OracleAS Wireless に付属するスクリプトを使用して、9.0.4.x に変換（移行）できます。このスクリプトは次の操作を実行します。

- 9.0.2.x のマスター通知に定義されている情報を使用して、新規テンプレート・ベースの 9.0.4.x のマスター通知を作成します。9.0.2.x の通知が時間ベースの場合、9.0.4.x では時間ベース・タイプが非完全時間ベースに設定されます。この設定は、OracleAS Wireless Webtool を使用していつでも変更できます。9.0.4.x のマスター通知の名前は、*<OLD_MASTER_NOTIFICATION_NAME>_NEW* となります。スクリプトによって、*_NEW* が 9.0.2.x のマスター・アラート名に追加されます。すべての情報（メッセージ・テンプレートのトリガー条件など）がコピーされ、メッセージ・テンプレートの一部が整形式の Mobile XML に必要に応じて変換されます。
- マスター・アプリケーションを作成して、新規マスター通知にマップします。このマスター・アプリケーションの名前は、*<OLD_MASTER_NOTIFICATION_NAME>_MS* となり、マスター・アプリケーションが `/master/notifications` と呼ばれるフォルダに作成されます。
- 9.0.2.x の各アラートに対する新規マスター・アプリケーションに基づいてリンクを作成します。このリンクの名前は、アラート・サービスと同じ名前になり、再び `/notifications` フォルダに作成されます。
- 9.0.2.x の特定のマスター通知アプリケーションの関連オブジェクト（AlertService、Topic など）に関するグループとユーザーのアクセス情報をすべて移行します。
- 既存のサブスクリプションをすべて 9.0.4.x のサブスクリプションに変換します。

移行スクリプトは、整合性とセキュリティを確保するために、古いマスター・アラート・サービスを削除したり、無効にすることはしませんが、ユーザーは、移行プロセスが正常に完了した後は、古いマスター・アラート・サービスを無効にする必要があります。

このスクリプトを実行する手順は、次のとおりです。

1. 移行対象のマスター通知を使用している通知プロセスをすべて停止するか、これらのマスター・アラート・サービスをその各プロセスから削除し、再起動します。
2. ディレクトリを $IASW_HOME/wireless/bin に変更します。
3. パラメータを次のように指定して、`migrateNotifications.[sh|bat]` を実行します。

   a. `migrateNotifications.[sh|bat] -name` *`<9.0.2.X master alert name(s)>`* `-owner` *`<owner user name>`*

   b. `migrateNotifications.[sh|bat] -oid` *`<9.0.2.X master alert oid>`* `-owner` *`<owner user name>`*

c. % は、名前パラメータのワイルドカードとして使用できますが、OID には使用できません。

d. 所有者のユーザー名を使用して、必要なマスター・アプリケーション、アプリケーション・リンクおよびフォルダを作成します。

表 11-1 移行スクリプトの例

スクリプト	機能
migrateNotifications.[sh|bat] -name StockAlert -owner orcladmin	「StockAlert」という名前のマスター通知を移行します。
migrateNotifications.[sh|bat] -name Stock% -owner orcladmin	「StockAlert」や「StockTransaction」などの「Stock」で始まるマスター通知をすべて移行します。
migrateNotifications.[sh|bat] -name % -owner orcladmin	すべてのマスター通知を移行します。
migrateNotifications.[sh|bat] -oid 1089 -owner orcladmin	OID 1089 を持つマスター通知を移行します。

e. OracleAS Wireless Webtool および Mobile Portal をチェックして、マスター通知、マスター・アプリケーション、リンクおよびサブスクリプションの移行プロセスを検証します。

f. 新規通知プロセスを作成して、移行したマスター・アラートを添付します。

移行した 9.0.2.x の通知を含む既存の通知プロセスを停止するか、これらのマスター通知を既存のプロセスから削除して、再起動する必要があることに注意してください。前述のように、移行スクリプトでは 9.0.2.x の通知は削除または無効化されません。したがって、このプロセスが正しくメンテナンスされていないと、ユーザーは、各通知を 2 つ（古いマスター通知から 1 つと新規マスター通知から 1 つ）受信することになります。

両バージョンでのサブスクリプション処理のサンプル・コード

9.0.4.x の通知システムは下位互換性があるため、9.0.2.x の通知またはサブスクリプションを変更するために作成されたカスタムのコードは、移行後もエラーや問題が発生せずに動作します。ただし、9.0.2.x の API 用に記述されたコードは、9.0.2.x の通知とサブスクリプションのみを変更します。つまり、新規マスター通知を古い API を使用して作成すると、作成されたマスター通知は、9.0.4.x の通知システムでは使用できません。したがって、9.0.4.x の通知エンジンでは処理されません。

新機能を利用するためには、開発者は、移行完了後に既存のコードを 9.0.4.x の API 用に変更する必要があります。このリリースでは、`oracle.panama.alert` パッケージに含まれていたすべての通知 API が使用中止になりました。9.0.4.x の機能を提供するために、新規パッケージ（`oracle.panama.mobilealert`）が導入されています。一部のメソッドとインタフェースは、9.0.2.x および 9.0.4.x の両方の API でレプリケートされます。ただし、すべ

ての 9.0.4.x の通知関連メソッド・コールは、この新規パッケージ（oracle.panama.mobilealert）を使用して実行する必要があります。9.0.2.x の API（oracle.panama.alert package）に存在するメソッドに対しても同じです。

次のサンプル・コードに、使用中止になった oracle.panama.alert パッケージを使用して 9.0.2.x でユーザー・サブスクリプションを作成する方法と、oracle.panama.mobilealert パッケージを使用して 9.0.4.x で（同一通知を移行後）ユーザー・サブスクリプションを作成する方法を示します。

9.0.2.x のサブスクリプションを追加するサンプル・コード

次のサンプル・コードに、9.0.2.x のサブスクリプションを追加する方法を示します。

```
    MetaLocator m = MetaLocator.getInstance();
    ModelServices s = m.getModelServices();

    //Assuming we have a user named "DemoUser"
    User myUser = s.lookupUser("DemoUser");

    //Assuming we have a validated E-Mail address, the first e-mail device address
    DeviceAddress[] deviceAddrList =
            myUser.getDeviceAddresses(DeliveryType.EMAIL);
    DeviceAddress subscriptionDeviceAddr = deviceAddrList[0];

    //Retrieve the alert service object that will be used to create a subscription
    oracle.panama.alert.AlertService alert =
s.lookupAlertService("StockNotification");

    alert.setUserAlertDevice(subscriptionDeviceAddr);

    oracle.panama.alert.UserAlertSubscription userSub =
            alert.addUserAlertSubscription(myUser);

    userSub.setDisplayName("DemoSubscription");

    // Set subscription time to 13:30 in daily mode.
    userSub.setHour(13);
    userSub.setMinute(30);
    userSub.setFrequency(
            new oracle.panama.alert.impl.AlertTimeFrequencyImpl(
                    oracle.panama.alert.AlertTimeFrequency.DAILY));

    // Expiration time set to one month ahead
    Calendar expireAt = Calendar.getInstance();
    expireAt.add(Calendar.MONTH, 1); //Expire next month
    userSub.setExpirationDate(expireAt);
```

```
    //Set the input parameter, i.e. stock ticker to ORCL
    oracle.panama.alert.AlertInputParamValue[] pVs = userSub.getInputParameters();
    pVs[0].setValue("ORCL");

    // set the triggering condition, i.e. stock price >= 20
    oracle.panama.alert.AlertConditionValue[] acv = userSub.getConditions();
    acv[0].setValue("20");
    userSub.setCondition(pVs, acv);

    // Save subscription
    userSub.save();

    // Save AlertService, so that user alert device can be persisted
    alert.save();
```

Sample code for adding the 9.0.4.X subscription after migration:

```
    MetaLocator m = MetaLocator.getInstance();
    ModelServices s = m.getModelServices();

    //Retrieve the master alert service object that will be used to create a
subscription
    oracle.panama.mobilealert.MasterAlertService masterAlertService =
            s.lookupMobileMasterAlertService("StockAlert");

    //Retrieve the link that will be used to create a subscription,
    //note that it has the same name as the AlertService object
    Link myLink = s.lookupLink("/notifications/StockNotification");

    //Assuming we have a user named "DemoUser"
    User myUser = s.lookupUser("DemoUser");

    //Assuming we have a validated E-Mail address, the first e-mail device address
    DeviceAddress[] deviceAddrList =
            myUser.getDeviceAddresses(DeliveryType.EMAIL);
    DeviceAddress subscriptionDeviceAddr = deviceAddrList[0];

    //
    oracle.panama.mobilealert.ServiceAlertSubscription userSub =
            masterAlertService.addUserAlertSubscription(myUser, myLink);

    userSub.setSubscriptionDevice(subscriptionDeviceAddr);
    userSub.setAlternativeType(ServiceAlertSubscription.AFTERMAX_DISCARD);

    userSub.setDisplayName("DemoSubscription");
```

```
// Set subscription time to 13:30 in daily mode.
//Since start/end time are same, interval can be any value
userSub.setHour(13);
userSub.setMinute(30);
userSub.setEndHour(13);
userSub.setEndMinute(30);
userSub.setInterval(1);
userSub.setFrequency(
        new oracle.panama.mobilealert.impl.AlertTimeFrequencyImpl(
                AlertTimeFrequency.DAILY));

// Expiration time set to one month ahead
Calendar expireAt = Calendar.getInstance();
expireAt.add(Calendar.MONTH, 1); //Expire next month
userSub.setExpirationDate(expireAt);

//Set the input parameter, i.e. stock ticker to ORCL
oracle.panama.mobilealert.AlertInputParamValue[] pVs =
        userSub.getInputParameters();
pVs[0].setValue("ORCL");

// set the triggering condition, i.e. stock price >= 20
oracle.panama.mobilealert.AlertConditionValue[] acv = userSub.getConditions();
acv[0].setValue("20");

// Save subscription information
userSub.save();
```

混乱を回避するために、すべての通知関連オブジェクトは、フル・パッケージ名で進められます。

12

J2ME の開発とプロビジョニング

項ごとに様々なトピックを記載しています。各項の内容は、次のとおりです。

J2ME の概要

J2ME（Java 2、Micro Edition）は、小型デバイスでの Java アプリケーション用のテクノロジです。 J2ME は、標準ベースのモバイル・アプリケーションの開発プラットフォームで、ブラウザ・ベースのソリューションに匹敵する豊富な UI を提供します。さらに、J2ME ベースのアプリケーションは、ネットワークの中断にもリジリエンスがあります。これは、このアプリケーションでは、多くの複雑な操作が Wireless ネットワークに依存せずに実行されるためです。

J2ME 対応の携帯情報端末が市場に浸透するにつれて、各企業では、自社のモバイル従業員のバックエンド・アプリケーションにアクセスするために、J2ME アプリケーションの開発に力をいれています。経営者もこれらの携帯情報端末に新規顧客のアプリケーションをデプロイして、自社の収益を拡大したいと考えています。

OracleAS Wireless は、完全な J2ME オファリングを提供します。その中には、Web サービス対応 J2ME アプリケーション、J2ME アプリケーション管理および J2ME アプリケーション・プロビジョニングなどの開発に必要な J2ME Developer Kit が、OracleAS Wireless Developer Kit のコンポーネントとして含まれています。この J2ME アプリケーション・プロビジョニングでは、携帯情報端末に対する柔軟で信頼性の高い J2ME アプリケーションのプロビジョニングを実行できます。

OracleAS Wireless J2ME ソリューションが提供する主な機能は、次のとおりです。

- Developer Kit: OracleAS Wireless Developer Kit は、Oracle Application Server Developer Kit のインストール・オプションを使用してインストールします。このキットには、J2ME アプリケーションから標準 Web サービスを起動するための、簡単な J2ME クライアント・ライブラリ API、ユーティリティおよび J2ME Web サービス・プロキシ・サーバーが用意されています。J2ME クライアント・ライブラリ API は、リクエスト・キューイング、レスポンス・キャッシュおよびネットワークの信頼性の欠如を解決するための追加機能など、他の拡張機能も提供します。
- J2ME アプリケーションの管理 : プロビジョニング・システムを使用すると、J2ME アプリケーションを OracleAS Wireless リポジトリにアップロードし、OracleAS Wireless Tools によって管理および分類できます。プロビジョニング・システムによって、開発者は、API スキャン・ポリシー、J2ME アプリケーションをサポートするデバイス（および J2ME アプリケーションをサポートできないデバイス）および J2ME アプリケーションの実行に必要なデバイス機能を指定できます。プロビジョニング・システムは、将来の請求とレポートのために、ユーザーのダウンロード履歴を追跡します。
- デジタル著作権管理（DRM）のサポート : サービス開発者は、デジタル著作権管理ポリシーを各 J2ME アプリケーションに関連付けることができます。OracleAS Wireless には、カウント・ベースと時間ベースのビルトイン DRM ポリシーがデフォルトで用意されています。これらのポリシーは、Open Digital Rights Language（ODRL）1.0 仕様に準拠しています。DRM のフレームワークをさらに拡張すると、任意の外部請求システムにその使用方法を統合できます。これによって、経営者やサービス・プロバイダは、J2ME プロビジョニング・サービス用の有利なビジネス・モデルを作成できます。

- 配信 : OracleAS Wireless J2ME プロビジョニング・システムは、J2ME アプリケーションを J2ME 対応の携帯情報端末に配信するための Sun 社の標準の無線環境 (OTA) プロビジョニング・プロトコルをサポートしています。さらに、このプロビジョニング・システムには、任意の配信プロトコルを介して任意の J2ME アプリケーションをプロビジョニングするためのオープンで拡張可能な配信フレームワークが用意されています。

OracleAS Wireless J2ME ソリューションを使用すると、サービス開発者は、Web サービス・テクノロジを介して企業のバックエンド・システムと統合する J2ME アプリケーションを迅速に開発できるだけでなく、コンテンツ・マネージャによってこれらのアプリケーションを簡単に管理できます。プロビジョニングとデジタル著作権管理のサポートは、顧客に J2ME アプリケーションを販売する経営者やサービス・プロバイダにとって、収益を創出する新しい手段となります。

機能の概要

次の各項では、J2ME クライアント・ライブラリと J2ME プロキシ・サーバーの機能の概要を説明します。

- MIDlet スイートの最小メモリー要件
- Web サービスの簡単な登録と起動
- SOAP Web サービスとエンタープライズ・アプリケーションの両方へアクセス
- 結果のキャッシュとコールのキューイング
- リクエストとレスポンスのパケット化と圧縮
- セッションのサポート
- OracleAS Wireless へのデプロイ

MIDlet スイートの最小メモリー要件

MIDlet はパッケージ化され、MIDlet スイートとしてデプロイされます。各 MIDlet スイートには、MIDlet の JAR ファイルとその記述子（JAD）ファイルが含まれています。J2ME クライアント・ライブラリのクライアント JAR ファイルに必要なメモリーは、MIDlet スイート内の 26KB のみです。現在使用可能な代替は、kSOAP 1.2 です。これは 41KB で、kXML が必要なため、さらに 21KB が追加され合計サイズは 62KB になります。

プロキシ・サーバーは、36KB のメモリー要件で使用可能です。これはサーバーが Web サービス上で操作を実行するため、MIDP デバイスが実行する作業量が削減されるためです。

Web サービスの簡単な登録と起動

Web サービスをコールする J2ME MIDlet の全体のプロセスは、次の操作で構成されています。

1. Web サービスの WSDL（Web Services Definition Language）文書を使用して、Web サービスを J2ME プロキシ・サーバーに登録します。
2. 登録したサービスに対して、J2ME クライアント・スタブ・クラスを生成します。
3. 使用している J2ME MIDlet から生成済スタブ・クラス上の Java メソッドをコールします。生成された J2ME クライアント・スタブの各メソッドは、Web サービスの操作を表します。たとえば、J2ME クライアント・ライブラリの JAR ファイルでコンパイルされた MIDlet は、Java メソッドをコールするだけで Web サービスへのリクエストを行います。

SOAP Web サービスとエンタープライズ・アプリケーションの両方へアクセス

SOAP Web サービスを J2ME プロキシ・サーバーに登録するには、Web サービスの WSDL 文書のファイル位置を指定します。WSDL 登録時に、J2ME プロキシ・サーバーは、SOAP クライアント Java クラスを生成します。実行時に、J2ME プロキシ・サーバーはこのクライアント・クラスをコールして、Web サービス上で操作を起動します。

J2ME プロキシ・サーバーは、クラス登録と呼ばれる第 2 タイプの登録もサポートしています。これによって、任意の Java クラスを J2ME プロキシ・サーバーに登録できます。このため、エンタープライズ・アプリケーションに対する Java クライアントを作成することで、J2ME MIDlet からこれらのアプリケーションにアクセスできます。この Java クライアントを J2ME プロキシ・サーバーに登録すると、Java クラスのすべてのパブリック・メソッドが J2ME MIDlet から起動できるようになります。

結果のキャッシュとコールのキューイング

生成された J2ME クライアント・スタブ・クラスのメソッドには、追加のパラメータが含まれています。このパラメータを使用すると、Web サービス・コールからの結果を MIDP デバイス上のローカル永続記憶域にキャッシュできます。これによって、デバイスがオフになった後やネットワーク・アクセスのない領域に移動した後でも、MIDlet は、追加のネットワーク・ラウンドトリップなしで、Web サービスから戻された結果に繰り返しアクセスできます。

コールのキューイングによって、ネットワーク・エラーで Web サービスを通常どおりコールできない場合でも、Web サービス操作の起動をキューできます。J2ME クライアント・ライブラリは、実行時にキューされたコールを成功するまで自動的に再試行し、そのレスポンスを MIDlet が取り出すまで永続記憶域にキャッシュします。

リクエストとレスポンスのパケット化と圧縮

リクエストのパケット化によって、Wireless ネットワークが特定サイズを超える HTTP リクエストを処理できない場合のために、リクエストの最大サイズを指定できます。J2ME クライアント・ライブラリと J2ME プロキシ・サーバーでは、コールのリクエストとレスポンスが、指定した最大サイズ未満に自動的に分割されます。

J2ME クライアント・ライブラリと J2ME プロキシ・サーバーでは、リクエストとレスポンスが、帯域幅とメモリーの使用量を改善するために圧縮されます。リクエストとレスポンスは、それぞれのサイズを削減するために、ネットワーク送信時と MIDlet でのキャッシュ時にエンコードされます。この操作は自動的に実行されます。

セッションのサポート

セッションのサポートは、デフォルトでアクティブ化されています。そのため、J2ME プロキシ・サーバーに登録された Java クラス（WSDL 登録時に生成されたクラスまたはクラス登録時に提供されたクラスのいずれか）のインスタンスは、HttpSession オブジェクトに格納され、J2ME プロキシ・サーバーが同じ MIDlet から複数のリクエストを受信したときに再利用されます。セッションのサポートは、J2ME クライアント・ライブラリにプロパティを設定して、オフにできます。

OracleAS Wireless へのデプロイ

MIDlet の開発が完了すると、J2ME プロキシ・サーバーに登録した J2ME MIDlet と Web サービスは、完全な OracleAS Wireless インストールに簡単にデプロイできます。WDK と OracleAS Wireless には、このデプロイを容易にする移行スクリプトが含まれています。

Wireless Developer Kit のスタート・ガイド

この項では、Wireless Developer Kit（WDK）の J2ME Web サービス・クライアント・ライブラリ（J2ME クライアント・ライブラリ）と J2ME Web サービス・プロキシ・サーバー（J2ME プロキシ・サーバー）を使用して、Web サービス対応の J2ME MIDlet を開発する方法を説明します。

セットアップ

OracleAS Wireless プロキシ・サーバーは、実際のサーバーとそのサーバーを管理するスクリプトという 2 つのコンポーネント・セットで構成されています。

1. 最初に、次のスクリプトを実行して、J2ME プロキシ・サーバーが組み込まれている WDK サーバーを起動します。

   WINDOWS: *<ORALCE_HOME>¥opmn¥bin¥opmnctl start wdk*

   UNIX: *<ORACLE_HOME>/opmn/bin/opmnctl.sh start wdk*

2. Wireless Developer Kit をファイアウォールの背後で実行する場合は、*<ORACLE_HOME>¥wireless¥bin/* にあるスクリプト j2mesdkmgr.bat（Windows）およびスクリプ

トj2mesdkmgr.sh（UNIX）で、使用する HTTP プロキシ・サーバーの設定も構成する必要があります。これらのスクリプトには、-registerwsdl オプションに対して HTTP プロキシ・サーバーを設定するためのコメント・アウトした例が含まれています。ファイアウォールの外側の URL にある Java クラスをサービスとして登録する場合は、前述のスクリプトの –registerclass オプションに HTTP プロキシ・サーバーの設定も含める必要があります。-registerwsdl オプションの例で示したように、HTTP プロキシ・サーバーを構成するには、Java コマンドラインで次の変数を定義します。

```
Dhttp.useProxy=true -Dhttp.proxyHost=<http proxy server>
Dhttp.proxyPort=<port number> -Dhttp.nonProxyHosts=<hosts inside firewall>
```

WDK の J2ME ディレクトリの構造

次に、J2ME クライアント・ライブラリと J2ME プロキシ・サーバーに関連する WDK のディレクトリを示します。

- wireless/--（OracleAS Wireless と WDK ホーム）
  - bin/（Web サービスの登録と管理用スクリプト）
  - lib/（ライブラリとプロパティのファイル）
  - j2me/--（J2me sdk ホーム）
    - docs/Javadoc（J2me SKD API ドキュメント）
    - lib/（J2me DK JAR ファイルを格納）
    - sample/（MIDlet と WSDL のサンプル・ファイル）
- j2ee/OC4J_Wireless/--（OracleAS Wireless J2EE ベース）
  - applications/wdk/wdk-Web/Webservice --（J2ME プロキシ・サーバー・ホーム）
    - repository/（登録済 Web サービスの説明）
    - stage/（WSDL 登録時に生成されたクラスのソース）
    - classes/（WSDL 登録時に生成されたコンパイル済のクラス）
    - lib/（サンプル MIDlet で使用したテスト・サービス付きの JAR ファイルを格納）
    - src/（ソース・コード）

例 : J2ME MIDlet の開発

この項では、Web サービスをコールする J2ME MIDlet の作成手順を説明します。

- ステップ 1: Web サービスを J2ME プロキシ・サーバーに登録します。
- ステップ 2: 登録した Web サービスに対して、J2ME クライアント・スタブ・クラスを生成します。
- ステップ 3: MIDlet から J2ME スタブ・クラスのメソッドをコールします。

Wireless Tools には、J2ME プロキシ・サーバーへのグラフィカル・インタフェースが用意されています。ただし、Wireless Developer Kit では、J2ME プロキシ・サーバーへのインタフェースは、コマンドライン・スクリプトを介して行います。J2ME プロキシ・サーバーの登録と管理のスクリプトは、次のとおりです。

Windows の場合 : *<ORACLE_HOME>¥wireless¥bin¥j2mesdkmgr.bat*

UNIX の場合 : *<ORACLE_HOME>/wireless/bin/ j2mesdkmgr.sh*

ステップ 1: Web サービスの J2ME プロキシ・サーバーへの登録

Web サービスは、Web サービス登録またはクラス登録のいずれかによって登録できます。

Web サービス登録の場合は、Web サービスを J2ME プロキシ・サーバーに登録して、これらの Web サービスが J2ME MIDlet からアクセスできるようにします。SOAP Web サービスを J2ME プロキシ・サーバーに登録する場合は、Web サービスを記述した WSDL 文書を指定します。登録を完了すると、登録済サービスから生成された J2ME スタブ・クラスのメソッドをコールすることで、Web サービスが J2ME MIDlet で使用可能となります。

SOAP Web サービスの登録に加えて、クラス登録を使用すると任意の Java クラスを J2ME プロキシ・サーバーに登録できます。Java クラスのすべてのパブリック・メソッドが、使用している MIDlet からリモート起動で使用できるようになります。これによって、エンタープライズ・アプリケーションに Java クライアントを作成することで、MIDlet から任意のエンタープライズ・アプリケーションに簡単にアクセスできます。この Java クラスには、引数なしのパブリック・コンストラクタか、またはクラスのインスタンスを戻す引数なしの `getInstance()` という静的パブリック・メソッドを設定する必要があります。

名前空間

Web サービスの登録時に、Web サービスを名前空間にグループ化できます。サービスを名前空間に関連付けることによって、関連する Web サービスをグループ化し、ネーミングの競合を回避できます。名前空間を指定しない場合、Web サービスはデフォルトの名前空間に登録されます。

WSDL 登録スクリプトのオプション（registerwsdl）

j2mesdkmgr –registerwsdl は、Web サービスの WSDL 文書を使用して Web サービスを登録します。

使用方法 :

j2mesdkmgr –registerwsdl <URL of the WSDL> [<namespace>]

たとえば、Oracle Technology Network（OTN）から入手可能な Hello World サービス（hello.wsdl）を登録するには、次のスクリプトを実行します。

Windows の場合 :

```
j2mesdkmgr.bat –registerwsdl
http://otn.oracle.com/tech/Webservices/htdocs/live/hello.wsdl
```

UNIX の場合 :

```
j2mesdkmgr.sh –registerwsdl
http://otn.oracle.com/tech/Webservices/htdocs/live/hello.wsdl
```

サービスを名前空間内部に登録するには、その名前空間をスクリプトの第 3 パラメータとして使用します。たとえば、Hello World アプリケーションを samples という名前空間に登録するには、次のスクリプトを実行します。

Windows の場合 :

```
j2mesdkmgr.bat –registerwsdl
http://otn.oracle.com/tech/Webservices/htdocs/live/hello.wsdl samples
```

UNIX の場合 :

```
j2mesdkmgr.sh –registerwsdl
http://otn.oracle.com/tech/Webservices/htdocs/live/hello.wsdl samples
```

クラス登録スクリプトのオプション（registerclass）

j2mesdkmgr –registerclass は、Java クラスを使用して Web サービスを登録します。

使用方法 :

j2mesdkmgr -registerclass <URL of the Class Library> <Name of the class> [<namespace>]

最初のパラメータは URL にする必要があります。このパラメータによって、このクラスが格納されているディレクトリまたは JAR ファイルのいずれかを指し示すことができます。

2 番目のパラメータは、完全修飾されたクラス名にする必要があります。たとえば、TestWebService（WDK に含まれている）というサービスを登録するには、次のスクリプトを実行します。

Windows の場合 :

```
j2mesdkmgr.bat -registerclass file:C:¥ora9ias¥j2ee¥OC4J_
Wireless¥applications¥wdk¥wdk-Web¥Webservice¥lib¥testservices.jar
oracle.wireless.me.server.TestWebService
```

UNIX の場合 :

```
j2mesdkmgr.sh -registerclass file:/ias/j2ee/OC4J_
Wireless/applications/wdk/wdk-Web/Webservice/lib/testservices.jar
oracle.wireless.me.server.TestWebService
```

注意 : この例の場合、WDK のホーム・ディレクトリは、/iaswv904/wireless（UNIX）と C:¥iaswv904¥wireless（Windows）です。

サービスを名前空間内部に登録するには、その名前空間をスクリプトの第 4 パラメータとして使用します。たとえば、TestWebService クラスを samples という名前空間に登録するには、次のスクリプトを実行します。

Windows の場合 :

```
j2mesdkmgr.bat -registerclass file:C:¥ora9ias¥j2ee¥OC4J_
Wireless¥applications¥wdk¥wdk-Web¥Webservice¥lib¥testservices.jar
oracle.wireless.me.server.TestWebService samples
```

UNIX の場合 :

```
j2mesdkmgr.sh -registerclass file:/ias/j2ee/OC4J_
Wilreless/applications/wdk/wdk-Web/Webservice/lib/testservices.jar
oracle.wireless.me.server.TestWebService samples
```

ステップ 2: 登録済 Web サービスに対する J2ME クライアント・スタブ・クラスの生成

MIDlet から J2ME プロキシ・サーバーに登録した Web サービスをコールする最も単純な方法は、そのサービスに対して J2ME スタブを生成し、生成済スタブのメソッドを MIDlet からコールする方法です。

スタブ生成スクリプトのオプション（-generatestub）

j2mesdkmgr -generatestub は、登録済 Web サービスに対して J2ME クライアント・スタブ・クラスを生成します。

使用方法 :

j2mesdkmgr -generatestub [<namespace>.]<service name> [<Output directory> [<stub name>]]

たとえば、Oracle Technology Network（IOTNHelloWorld）から入手可能な Hello World サービスに J2ME スタブ・クラスを生成するには、次のスクリプトを実行します。

j2mesdkmgr -generatestub IOTNHelloWorld

サービスが名前空間内部にある場合は、サービス名の接頭辞に名前空間とピリオド（.）を使用する必要があります。たとえば、Hello World サービス（IOTNHelloWorld）を名前空間 samples の内部に登録する場合は、次のスクリプトを使用してスタブを生成します。

j2mesdkmgr -generatestub **samples.IOTNHelloWorld**

生成済のスタブを配置するディレクトリを、スクリプトの 3 番目のパラメータとして指定できます。

スタブ名を指定しない場合、次の生成済 J2ME クライアント・スタブがコールされます。

IOTNHelloWorldJ2MEStub.Java

生成済のスタブに別の名前を指定する場合は、第 3 パラメータとして出力ディレクトリ（次の例では `tmp`）を、第 4 パラメータとして必要なスタブ・クラス名（次の例では `HelloWord`）を指定します。次に例を示します。

Windows の場合 :

j2mesdkmgr.bat -generatestub samples.IOTNHelloWorld **c:¥tmp HelloWorld**

UNIX の場合 :

j2mesdkmgr.sh -generatestub samples.IOTNHelloWorld **/tmp HelloWorld**

生成された J2ME クライアント・スタブ・クラスには、Web サービスの各操作ごとに 1 つのメソッドが格納されています。

ステップ 3: MIDlet からの J2ME スタブ・クラス・メソッドのコール

生成された J2ME スタブ・クラス内の各パブリック・メソッドは、登録されたサービスの操作を表します。各メソッドの最初の 2 つのパラメータは、J2ME クライアント・ライブラリにとって、次のような特別の意味を持っています。最初のパラメータはレスポンスをキャッシュするための分数で、2 番目のパラメータはブールです。このブールは、キャッシュ済のレスポンスをすべて無視する場合は true に設定する必要があります（つまり、J2ME クライアント・ライブラリに Web サービスへのネットワーク・コールを強制します）。また、キャッシュされたレスポンス（使用可能で有効な場合）を使用する場合は、ブールを false に設定する必要があります。これらのパラメータの詳細は、12-15 ページの「レスポンスのキャッシュ」を参照してください。

生成済のスタブ内にあるメソッドのその他すべてのパラメータは、メソッドが表すサービス操作のパラメータに対応しています。これらのメソッドを MIDlet からコールすると、Web サービス上のこれらの操作が起動します。

MIDlet を Sun 社の J2ME Wireless Toolkit（http://Java.sun.com/products/j2mewtoolkit/download.html）でテストする手順は、次のとおりです。

1. MIDlet にプロジェクトを作成します。プロジェクトの lib ディレクトリに J2ME クライアント・ライブラリの JAR ファイル j2me_sdk.jar を配置します。生成された J2ME スタブ・クラスをプロジェクトの src ディレクトリに配置します。
2. J2ME MIDlet のソース・コードで、生成された J2ME スタブ・クラスに `import` 文を入力し、スタブ・クラスのインスタンスを作成して、スタブ・クラス内のメソッドをコールします。各スタブ・メソッドの最初のパラメータは、結果をキャッシュするための分数です。キャッシュしない場合は 0（ゼロ）を、永続的にキャッシュする場合は -1 を設定します。各スタブ・メソッドの第 2 パラメータは、キャッシュ結果を無視する場合（常にネットワーク・コールを行う場合）は true、有効なキャッシュ結果（使用可能な場合）を使用する場合は false です。スタブ・メソッドの残りのパラメータは、そのメソッドが表す Web サービス操作のパラメータに対応しています。
3. J2ME MIDlet を作成し、実行します。

Hello World の例 J2ME クライアント・ライブラリには、複数のサンプル MIDlet が含まれています。これらの 1 つが、生成済のスタブを使用して IOTNHelloWorld サービスをコールします。次に、サンプル MIDlet を示します。

<ORACLE_HOME>¥wireless¥j2me¥j2mesdk¥sample¥HelloWorld.Java

注意： ディレクトリには、生成済のスタブまたは J2ME クライアント・ライブラリ API のいずれかを使用し、J2ME プロキシ・サーバーを介して Web サービスをコールする例が含まれた他の MIDlet も格納されています。

その他の管理コマンドライン・ユーティリティ

listservices オプションは、すべての登録済サービスをリストします。removeservice オプションは、登録済サービスを削除します。

リスト・サービス・オプション（-listservices）

オプション j2mesdkmgr –listservices は、すべての登録済サービスをリストします。このスクリプトにはパラメータは不要です。

このスクリプトは、すべての登録済サービスをリストしますが、サービス内の使用可能なメソッドまたはメソッドに必要なパラメータは表示しません。この情報は、次のように WDK 内の J2ME プロキシ・サーバーへの URL を使用して、Web ブラウザで表示できます。

http://<host name>:9010/wdk/proxy

次に例を示します。

http://www.example.com:9010/wdk/proxy

サービス削除のオプション（-removeservice）

オプション j2mesdkmgr –removeservice は、登録済サービスを削除します。

使用方法 :

j2mesdkmgr –removeservice [<namespace>.]<service name>

たとえば、登録済サービス Hello World（IOTNHelloWord）を削除するには、次のスクリプトを実行します。

j2mesdkmgr –removeservice IOTNHelloWorld

サービスが名前空間内部にある場合は、サービス名の接頭辞に名前空間とピリオド（.）を使用する必要があります。たとえば、名前空間 samples 内部に登録されている Hello World サービス （IOTNHelloWorld）を削除するには、次のスクリプトを実行します。

j2mesdkmgr –removeservice **samples.IOTNHelloWorld**

TestStubMidlet を使用した簡単なサービスへのアクセス

J2ME クライアント・ライブラリには、TestStubMidlet というサンプル J2ME MIDlet が用意されています。このサンプルを使用すると、J2ME プロキシ・サーバーに登録する Web サービスについて、生成済のスタブ・ファイルを迅速にテストできます。

サンプル TestStubMidlet を使用する手順は、次のとおりです。

1. Sun 社の J2ME Wireless Toolkit（`http://Java.sun.com/products/j2mewtoolkit/download.html`）をダウンロードして、インストールします。
2. 次の場所にあるサンプル MIDlet ファイル TestStubMidlet.Java を次のように変更します。

   <ORACLE_HOME>/wireless/j2me/j2mesdk/sample/

a. 生成された Java スタブ・クラスに `import` 文を追加します。

b. `callStub()method` を変更してスタブ・クラスをインスタンス化し、操作をコールして、結果の表示に使用するインスタンス変数 `sCallResultString` に結果を割り当てます。

3. Sun 社の J2ME Wireless Toolkit にプロジェクトを作成します（最初に OracleJ2ME Web サービス・クライアント・ライブラリの j2me_sdk.jar を検索する必要があります。これは *<ORACLE_HOME>/wireless/wdk/j2me/lib/* にあります）。

このライブラリをプロジェクトの lib ディレクトリ内に配置します。スタブ・クラスをプロジェクトの src ディレクトリに、変更した `TestStubMidlet` クラスをプロジェクトの src ディレクトリ内にある oracle/wireless/me/sample1/ サブディレクトリにそれぞれ配置します（`TestStubMidlet` はパッケージ `oracle.wireless.me.sample1` 内にあります）。

4. プロジェクトを作成し、実行します。J2ME デバイス・エミュレータが表示されたときに、テスト用の MIDlet を起動し、Call Stub を実行します。テスト結果が表示されます。

次に例を示します。

この例では、`TestStubMidlet` を使用して XMethods Delayed Stock Quote Web サービスをコールし、オラクル社の株価を表示します。

a. XMethods Delayed Stock Quote Web サービス（xmethods-delayed-quotes.wsdl）を次のように登録します。

```
j2mesdkmgr -registerwsdl
http://services.xmethods.net/soap/urn:xmethods-delayed-quotes.wsdl
```

b. スタブを次のように生成します。

```
j2mesdkmgr -generatestub NetXmethodsServicesStockquoteStockQuoteService
```

c. 次の例のように、TestStubMidlet.Java を変更します。

```
...
import NetXmethodsServicesStockquoteStockQuoteServiceJ2MEStub;
...
/**
* Call Stub. Edit this method to test the stub.
* Remember to import the stub class
*/
private void callStub()
{
  try {
    // Add your code to test the stub.
    // For example:
    // Use the stub to call the XMethods delayed stock quote service to get
    // Oracle's stock price and cache the result for 1 minute.
    // Instantiate the Stub class:
```

```
    NetXmethodsServicesStockquoteStockQuoteServiceJ2MEStub stub =
       new NetXmethodsServicesStockquoteStockQuoteServiceJ2MEStub();

    // Call GetQuote Operation to get Oracle's stock price and
    // cache the result for 1 minute
    sCallResultString = new String("Stock price for ORCL :" +
         stub.getQuote(1, false, "ORCL"));
  }
...
```

d. 作成し、実行します。オラクル社の遅延株価が、テスト結果画面に表示されます。2 度目のスタブを 1 分以内に起動した場合、結果は即時に表示されます。これは、ネットワークへのラウンドトリップなしで、ローカル・キャッシュから読み取るためです。

他の Web サービス用 TestStubMIDlet の使用 次の Web サービスを TestStubMidlet に登録してテストできます。

- Oracle Technology Network（http://otn.oracle.com/tech/Webservices/htdocs/live/content.html）が提供する Web サービス：

  Hello World:

  ```
  http://otn.oracle.com/tech/Webservices/htdocs/live/hello.wsdl
  ```

- XMethods 社（http://www.xmethods.com）が提供する Web サービス：

  Delayed Stock Quotes:

  http://services.xmethods.net/soap/urn:xmethods-delayed-quotes.wsdl

  Weather Temperature:

  http://www.xmethods.net/sd/2001/TemperatureService.wsdl

  FedEx Tracker:

  http://www.xmethods.net/sd/2001/FedExTrackerService.wsdl

TestWebService と TestWebService2 のサンプル・サービス OracleAS Wireless の J2ME プロキシ・サーバーには、Java クラスに 2 つの簡単なサンプル Web サービスを持つ JAR ファイルが含まれています。J2ME クライアント・ライブラリとともに配布されるサンプル MIDlet の一部は、これらのサンプル・サービスを使用します。サンプル・サービスを登録するには、次のスクリプトを実行します。

Windows の場合 :

<ORALCE_HOME>¥wireless¥bin¥installj2mesamples.bat

UNIX の場合 :

<ORALCE_HOME>/wireless/bin/installj2mesamples.sh

これらのスクリプトは、2 つの Java クラスを Web サービス（TestWebService と TestWebService2）として登録します。

`TestWebService` クラスには、異なるパラメータ・タイプをテストするための簡単なメソッドがいくつか含まれています。このクラスには、引数なしのパブリック・コンストラクタがあります。

`TestWebService2` クラスには、簡単なテスト・メソッドが 1 つと、TestWebService2 のインスタンスを戻す引数なしの `getInstance()` メソッドが含まれています。このクラスのコンストラクタは、プライベートです。

これらのテスト・サービスのソース・コードは、次の場所にあります。

<ORACLE_HOME>/j2ee/OC4J_Wireless/applications/wdk/wdk-Web/Webservice/src

コンパイル済サービスを含む JAR ファイルは、次の場所にあります。

<ORACLE_HOME>/j2ee/OC4J_
Wireless/applications/wdk/wdk-Web/Webservice/lib/testservices.jar

拡張機能

この項では、J2ME クライアント・ライブラリと J2ME プロキシ・サーバーが提供する主な機能の使用方法について説明します。

レスポンスのキャッシュ

生成済スタブ内のすべてのパブリック・メソッドの最初の 2 つのパラメータは、`int timetokeep` および `boolean refresh` です。これらのパラメータを使用してレスポンスのキャッシュを制御します。J2ME クライアント・ライブラリでは、ワイヤレス・デバイスの永続メモリー内にある Web サービス操作のレスポンスをキャッシュできます。同じ操作を将来起動するために、このレスポンスは、Web サービスではなくローカル・キャッシュから取得されます。

`int timetokeep`

このパラメータは、キャッシュ内にレスポンスを保持する時間（分単位）を示します。指定した分数内で同じ操作を再度起動する場合、レスポンスは、Web サービスへのラウンドトリップなしで、ローカル・キャッシュから取得されます。指定した分数が経過すると、キャッシュ内のレスポンスは無効になり、この操作の次の起動では、再度 Web サービスへのネットワーク・ラウンドトリップが行われます。キャッシュを無効にするには、このパラ

メータを 0（ゼロ）に設定し、永続的にキャッシュする（レスポンスが無効にならない）場合は、-1 に設定します。

`boolean refresh`

このパラメータは、この操作の起動時にキャッシュを無視するかどうかを示します。このパラメータを true に設定すると、有効なレスポンスがローカル・キャッシュ内で使用可能な場合でも、ネットワーク・ラウンドトリップが実行されます。このパラメータを false に設定すると、使用可能なキャッシュ済レスポンスが有効な場合、レスポンスはローカル・キャッシュから取得されます。

HTTP 認証

J2ME のプロキシ・サーバーとクライアント・ライブラリでは、HTTP 基本認証方式を使用する Web サービスへのアクセスをサポートしています。認証用のユーザー名とパスワードを設定するには、インスタンス変数の `userName` および `password` を適切な値に設定して、Web サービスの生成済スタブを更新します。このユーザー名とパスワードは、認証のためにプロキシ・サーバーによって Web サービスの HTTP サーバーに渡されます。

セッションのサポート

セッションのサポートはデフォルトでアクティブになっています。セッションのサポートを無効にするには、インスタンス変数 enableSession を false に設定して、Web サービスの生成済スタブを更新します。J2ME プロキシ・サーバーは、Web サービスの J2ME プロキシ・サーバーへの登録時に WSDL 登録を使用したか、クラス登録を使用したかに関係なく、Java クラスを使用して Web サービス上の操作を起動します。セッションのサポートは、同じ J2ME デバイスによる連続した Web サービスの操作の起動では、この Java クラスの同じインスタンス・オブジェクトが使用されることを意味します。クラスのインスタンスは、J2ME プロキシ・サーバーのサーブレット HttpSession オブジェクトに保存されます。そのため、HTTP セッションが有効であるかぎり使用できます。

リクエストとレスポンスのパケット化

J2ME クライアント・ライブラリは、HTTP を使用してリクエストを J2ME プロキシ・サーバーに送信します。HTTP はすべての MIDP 実装のサポートを保証している唯一のプロトコルです。一部の携帯電話会社では、リクエストが特定サイズを超えていると、HTTP リクエストを正しく送信できません。この問題を回避するために、J2ME クライアント・ライブラリでは、リクエストとレスポンスの最大サイズをバイト単位で指定できます。Web サービス用に生成された J2ME スタブ内で、`maxRequestSize` インスタンス変数を編集し、HTTP リクエストとレスポンスの最大サイズをバイト単位で設定します。

クライアント・ライブラリ API

多くの場合、サーバー生成のクライアント・スタブを使用すると、J2ME プロキシ・サーバーに登録したサービスを起動できます。ただし、コール・キューイングなどの拡張機能を使用する場合は、独自の MIDlet からまたは生成済のスタブを変更して、クライアント・ライブラリ API を直接使用する必要があります。

J2ME クライアント・ライブラリの API ドキュメントは、次の場所にあります。

<ORACLE_HOME>¥wireless¥j2me¥Javadoc¥index.html

Web サービス・コール `ServiceFactory` クラスを使用して、J2ME プロキシ・サーバーに登録したサービスを表す Service オブジェクトを作成します。アクセスを容易にするために、J2ME プロキシ・サーバーの URL を MIDlet スイートのプロパティとして保存します。たとえば、`TestWebService` クラスの `Service` オブジェクトをデフォルトの名前空間に作成するには、次のスクリプトを実行します（この場合、J2ME プロキシ・サーバーの URL が `j2me_proxy_url` という MIDlet スイート・プロパティに格納されていると仮定します）。

```
// Get the URL of the J2ME proxy server.
String servURL = getAppProperty("j2me_proxy_url");
if (servURL == null) {
servURL = "http://localhost:9010/wdk/proxy";
System.out.println("Unable to get j2me_proxy app property, using
default: " + servURL);
}
ServiceFactory serviceFactory = ServiceFactory.getInstance();
Service service = serviceFactory.createService(servURL,Service.DEFAULT_
NAMESPACE,"TestWebService");
```

このコード例では、`createService` の最初のパラメータは J2ME プロキシ・サーバーの URL です。2 番目の servURL はサービス（`Service.DEFAULT_NAMESPACE`）を格納する名前空間で、3 番目は登録済サービス（`"TestWebService"`）の名前です。

クラス `Service` のオブジェクトを作成した後は、`Call` オブジェクトを作成できます。各 `Call` オブジェクトは、リモートの Web サービスの操作を表します。たとえば、hello という操作の `Call` オブジェクトを作成するには、次のスクリプトを実行します。

```
// Set cache timeout to 1 hour.
Call call = service.createCall("hello", 60);
```

`createCall` の最初のパラメータは、Web サービス（`"hello"`）の操作名です。2 番目のパラメータは、ワイヤレス・デバイスでレスポンスをキャッシュする分数（`60`）です。レスポンスのキャッシュを無効にするには 0（ゼロ）を、永続的にキャッシュする場合は -1 を入力します。詳細は、12-15 ページの「レスポンスのキャッシュ」を参照してください。

`Call` オブジェクトを作成した後は、そのオブジェクトが表す Web サービス操作を起動できます。Java ベクターを作成してパラメータを保持します。次に `Call` オブジェクトの `invoke` メソッドをコールします。hello 操作に 1 つの `String` パラメータが必要な場合、`invoke hello` は次のようになります。

```
// Put the parameters in a Vector.
Vector params = new Vector();
params.addElement( new String("World") );
Response response = call.invoke(params, false);
```

Call オブジェクトの起動メソッドの最初のパラメータは、Web サービス操作用パラメータのベクターです。2 番目のパラメータは、ローカル・キャッシュを無視する場合、true に設定できます。つまり、true を設定すると、キャッシュされた有効なレスポンスはすべて無視され、リモート Web サービスへのネットワーク・コールが実行されます。詳細は、12-15 ページの「レスポンスのキャッシュ」を参照してください。

起動メソッドは Response オブジェクトを戻します。Response オブジェクトのリターン・コードは常にチェックしてください。リターン・コードが 0（ゼロ）の場合、コールは成功し、Response オブジェクトのメソッドを使用して、リモート操作から戻された情報にアクセスできます。リターン・コードが 0（ゼロ）でない場合は、Response オブジェクトの getErrorMsg メソッドを使用してエラー・メッセージを取得してください。操作 hello が String を戻した場合は、次のスクリプトを実行します。

```
// Check if the response is valid (0 = OK).
int retCode = response.getReturnCode();
String retVal;
if (retCode == 0) {
  retVal = response.getString();
}
else { // return code not 0
  retVal = "Call unsuccessful: " + response.getErrorMsg();
}
```

プロパティの設定 Service.setProperty method を使用して Service オブジェクトの次のプロパティを設定します。このオブジェクトは、J2ME プロキシ・サーバーに登録された Web サービスを表します。

Service.USERNAME および Service.PASSWORD: HTTP 基本認証が必要な Web サービスにアクセスするには、これらのプロパティを設定します。

Service.SESSION_MAINTAIN: セッション・サポートを有効または無効にするには、このプロパティを設定します。詳細は、12-5 ページの「セッションのサポート」を参照してください。

Service.MAX_REQUEST_SIZE: HTTP リクエストとレスポンスの最大サイズを設定します。詳細は、12-16 ページの「リクエストとレスポンスのパケット化」を参照してください。

コール・キューイング ワイヤレス・デバイスから Web サービスへのコールが、ネットワーク接続の問題で失敗する場合があります。J2ME クライアント・ライブラリを使用すると、MIDlet は Web サービス・コールをキューし、後で起動できます。キューされたコールは、J2ME クライアント・ライブラリによって、コールが成功するまで定期的にかつ自動的に再試行されます。ネットワーク・エラーの後に ServiceManager.queueCall メソッドを使用すると、コールは次のようにキューされます。

```
try {
   Response response = call.invoke(params, false);
...
catch (ServiceException se) {
   if (se.getErrorCode() == ServiceException.CONNECTION_FAILED) {
      int queuedId = ServiceManager.getInstance().queueCall(call);
   }
   ...
}
...
```

ServiceManager クラスの getQueuedCalls メソッドを使用してキューされたコールを取得します。起動済のキューされたコールの ID を取得するには、getQueuedCalls のブール・パラメータを true に、未起動のキューされたコールの ID を取得するには、false に設定します。次に、ServiceManager.getQueuedCall メソッドを使用して、ID から Call オブジェクトを取得し、Call.getResponse を使用してコールのレスポンスを取得します。

起動済の Web サービスのキューされたコールを検索するには、次のスクリプトを実行します。

```
int[] queuedIds = ServiceManager.getInstance().getQueuedCalls( true );
for (int i = 0; i < queuedIds.length; i++){
  //Retrieve Call object
  Call invokedCall =
     ServiceManager.getInstance().getQueuedCall(queuedIds[i]);
  Response response = invokedCall.getResponse();
  ...
}
```

未起動のキューされたコールを検索するには、次のスクリプトを実行します。

```
...
int[] queuedIds = ServiceManager.getInstance().getQueuedCalls( false );
...
```

MIDlet に必要なクリーンアップ MIDlet が存在している場合は、常に ServiceManager クラスの close() method をコールする必要があります。これによって、レスポンス・キャッシュとコール・キューイングに使用する RMS RecordStore がクローズされます。J2SE 環境では、この操作は、ServiceManager の finalize() メソッドで実行されます。ただし、J2ME は、finalize() メソッドをサポートしていません。したがって、このコードが、J2ME クライアント・ライブラリが使用する RMS RecordStore をクローズするために使用されます。

MIDlet のすべてのクリーンアップ・コードを 1 つのメソッドに収集する必要があります。このメソッドを MIDlet の destroyApp メソッドおよび MIDlet の終了の原因となるその他のコード（Exit コマンドなど）からコールします。次の例は、J2ME クライアント・ライブラリにあるサンプル MIDlet の 1 つからの抜粋です。クリーンアップ・コードが exitMIDlet メソッドに収集されます。

```
private void exitMIDlet()
{
  try {
    ServiceManager.getInstance().close();
  }
  catch (Exception e) {
  }

  notifyDestroyed();
}


protected void destroyApp(boolean unconditional)
   throws MIDletStateChangeException
{
   exitMIDlet();
}


public void commandAction(Command c, Displayable d)
{
   if (c == exitCommand) {
exitMIDlet();
}
            . . .
}
```

サポートされているデータ型 現在 J2ME の Web サービス・クライアント・ライブラリとプロキシ・サーバーでは、Web サービス操作パラメータと戻り値について、次のデータ型をサポートしています。

- `String`
- `Integer`
- `Boolean`
- `Date`
- `int`
- `boolean`
- `Vector`
- `Hashtable`
- `String`、`Integer`、`Boolean` および `Date` の配列

登録されたすべての Web サービスについては、これらのデータ型を使用する操作のみが J2ME プロキシ・サーバーに登録でき、J2ME クライアント・ライブラリを使用して起動できます。

注意： J2ME は、データ型 `float` または `double` をサポートしていません。解決策として、J2ME プロキシ・サーバーは、`float` と `double` を `String` に変換します。この方法は、Web サービス操作の戻り値のデータ型のみに適用され、操作パラメータには適用されません。この機能を使用すると、価格、気温およびその他の小数値を戻す Web サービス操作をコールできます。

次に、WSDL の簡単なデータ型（XML Schema のデータ型）を Java データ型にマッピングする方法を示します。

表 12-1 WSDL データ型のマッピング

WSDL データ型	Java データ型
BOOLEAN	Java.lang.Boolean
integer	Java.lang.Integer
string	Java.lang.String
dateTime	Java.util.Date

OracleAS Wireless への MIDlet のデプロイ

Wireless Developer Kit を使用して開発とテストを終了した Web サービス対応 J2ME MIDlet は、実際の OracleAS Wireless インストールにデプロイできます。次の 2 つの方法のいずれかを使用して実行できます。

再登録によるデプロイ Web サービスを OracleAS Wireless J2ME プロキシ・サーバーに再登録することで、作業を移行できます。OracleAS Wireless Tools の 1 つであるサービス・マネージャを使用して、Web サービスを登録し、サービスの J2ME スタブを生成できます。

再登録によってサービスをデプロイする手順は、次のとおりです。

1. MIDlet がアクセスするすべての Web サービスを登録します。
2. MIDlet がコールする Web サービスに対して新しい J2ME スタブを生成します。
3. MIDlet を新しい J2ME スタブで再コンパイルします。

生成済のスタブを手動で変更した場合は、ステップ 2 の代替としてすでに所有している J2ME スタブを手動で変更する必要があります。これを実行するには、`_proxyUrl instance` 変数の値を変更する必要があります。次の URL を OracleAS Wireless サーバーの正しいホスト名、OracleAS Wireless の正しいポート番号（通常は 7777）および正しいパス（通常は /mcs/wsproxy/proxy）に更新します。次に例を示します。

```
private String _proxyUrl= "http://example.com:7777/mcs/wsproxy/proxy";
```

スクリプトによるデプロイ 移行スクリプトを使用することで、MIDlet がアクセスするすべての Web サービスの再登録が不要になります。このプロセスは、登録済 Web サービス上のすべての情報を XML ファイルにダウンロードまたはエクスポートし、次に、この情報を XML ファイルから OracleAS Wireless インストールにアップロードまたはインポートする構成になっています。

1. 次のスクリプトを使用して、WDK J2ME プロキシ・サーバーに登録した Web サービスを XML ファイルにエクスポートします。

   `migrateStandalone`

   このスクリプトを使用して、WDK J2ME プロキシ・サーバーに登録した Web サービスを XML ファイルに抽出します。このスクリプトは、Web サービスの登録と管理に使用する他の WDK J2ME プロキシ・サーバーのスクリプトとともに、*<ORACLE_HOME>/wireless/wdk/bin* にあります。

   使用方法 :

   ```
   j2mesdkmgr -export <xml file name>
   ```

   例 :

   Windows の場合 :

   ```
   j2mesdkmgr -export C:¥temp¥registered_services.xml
   ```

   UNIX の場合 :

   ```
   j2mesdkmgr.sh -export /usr/tmp/registered_services.xml
   ```

2. 次のスクリプトを使用して XML ファイルを OracleAS Wireless インストールにインポートします。

   `uploadJ2MEProxy`

   このスクリプトには、1 つのパラメータ、j2mesdkmgr -export スクリプトによって作成された XML ファイルの名前またはフルパスが必要です。この XML ファイルに記述されているすべての Web サービスは、この OracleAS Wireless のデータベースに挿入され、この OracleAS Wireless インストールの J2ME プロキシ・サーバーに登録されます。このスクリプトは、*<ORACLE_HOME>/wireless/bin/* ディレクトリにあります。

   使用方法 :

   ```
   uploadJ2MEProxy <xml file name>
   ```

   例 :

   Windows の場合 :

   ```
   uploadJ2MEProxy.bat c:¥temp¥registered_services.xml
   ```

   UNIX の場合 :

   ```
   uploadJ2MEProxy.sh /usr/tmp/registered_services.xml
   ```

3. MIDlet がコールする Web サービスに対して新しい J2ME スタブを生成します。

   すでに所有している J2ME スタブを手動で変更する場合は、この手順をスキップできます（生成済のスタブを手動で変更している場合は、スキップしてください）。_proxyUrl インスタンス変数の値を OracleAS Wireless サーバーの正しいホスト名、正しいポート番号（通常は 7777）および正しいパス（/mcs/wsproxy/proxy）に変更する必要があります。次に例を示します。

   ```
   private String _proxyUrl= "http://example.com:7777/mcs/wsproxy/proxy";
   ```

4. MIDlet を新しいまたは修正された J2ME スタブで再コンパイルします。

OracleAS Wireless インストール間の移行

ある時点で、J2ME プロキシ・サーバーに登録した Web サービスに関するすべての情報を、1 つの OracleAS Wireless インストールから別のインストールに移行する場合があります。このプロセスは、ソースの OracleAS Wireless の登録済 Web サービスに関するすべての情報を XML ファイルにダウンロードまたはエクスポートし、次に、この情報を XML ファイルから移行先の OracleAS Wireless にアップロードまたはインポートする構成になっています。これを実行するスクリプトは、次のディレクトリにあります。

<ORACLE_HOME>/wireless/bin/

J2ME プロキシ・サーバーに登録した Web サービスの情報を、1 つの OracleAS Wireless インストールから別のインストールに移行する手順は、次のとおりです。

1. 次のスクリプトを使用して、ソースの OracleAS Wireless J2ME プロキシ・サーバーに登録した Web サービスを XML ファイルにエクスポートします。

   ```
   downloadJ2MEProxy
   ```

   このスクリプトには、1 つのパラメータ、つまり XML ファイルの名前またはフルパスが必要です。このスクリプトを実行すると、OracleAS Wireless にある J2ME プロキシ・サーバーに登録済のすべての Web サービスは、OracleAS Wireless データベースから抽出され、この XML ファイルに配置されます。

   使用方法 :

   ```
   downloadJ2MEProxy <xml file name>
   ```

   例 :

   Windows の場合 :

   ```
   downloadJ2MEProxy.bat c:¥temp¥registered_services.xml
   ```

   UNIX の場合 :

   ```
   downloadJ2MEProxy.sh /usr/tmp/registered_services.xml
   ```

2. 次のスクリプトを使用して、XML ファイルをインポート先の OracleAS Wireless J2ME プロキシ・サーバーにインポートします。

   `uploadJ2MEProxy`

   このスクリプトには、1 つのパラメータ、つまり downloadJ2MEProxy スクリプトによって作成された XML ファイルの名前またはフルパスが必要です。この XML ファイルに記述されているすべての Web サービスは、このスクリプトを実行すると、OracleAS Wireless のデータベースに挿入され、この OracleAS Wireless インストールの J2ME プロキシ・サーバーに登録されます。

   使用方法 :

   ```
   uploadJ2MEProxy <xml file name>
   ```

   例 :

   Windows の場合 :

   ```
   uploadJ2MEProxy.bat c:¥temp¥registered_services.xml
   ```

   UNIX の場合 :

   ```
   uploadJ2MEProxy.sh /usr/tmp/registered_services.xml
   ```

3. MIDlet がコールする Web サービスに対して新しい J2ME スタブを生成します。

   すでに所有している J2ME スタブを手動で変更する場合は、この手順をスキップできます（生成済のスタブを手動で変更している場合は、スキップしてください）。`_proxyUrl` インスタンス変数の値を変更します。次の URL を OracleAS Wireless サーバーの正しいホスト名、正しいポート番号（通常は 7777）および正しいパス（/mcs/wsproxy/proxy ）に更新します。次に例を示します。

   ```
   private String _proxyUrl= "http://example.com:7777/mcs/wsproxy/proxy";
   ```

4. MIDlet を新しい（または修正された）J2ME スタブで再コンパイルします。

デジタル著作権管理のサポート

デジタル著作権管理（DRM）によって、コンテンツ・マネージャでは、コンテンツがデバイスにダウンロードされた後のコンテンツ使用ポリシーを定義できます。この使用ポリシーは、必要に応じて、一時的および金銭的な追加制約を持つエンド・ユーザーに対してコンテンツに関連して許容される権限を定義します。一般的な権限のタイプは、次のとおりです。

- `Execute`: アプリケーションを起動する権利（J2ME ゲームなど）。
- `Display`: コンテンツを表示する権利（イメージなど）。
- `Play`: コンテンツを再生する権利（オーディオ / ビデオ・クリップなど）。
- `Print`: コンテンツのハードコピーを作成する権利（image/jpeg など）。
- コンテンツのきめ細かな使用規制を提供するためのオプション制約（プレビュー権限など）。
- `Count`: 権限の付与回数（実行できる回数など）。
- `Interval`: 権限の付与期間（使用できる時間数など）。
- `Start` 時間と `End` 時間 : 権限が付与される事前定義の期間。

デジタル権利を表す標準は多数あります。モバイル・コンテンツについて Open Mobile Alliance（OMA）が提唱する標準は、Open Digital Rights Language（ODRL）の簡略版で、デジタル権利は ODRL モバイル・プロファイルとして定義された構文に従って XML 文書で表現する必要があるとしています。

OracleAS Wireless の組込み DRM ポリシー

OracleAS Wireless には、J2ME アプリケーションのパッケージ化に使用できる回数ベースの DRM ポリシーと期間ベースの DRM ポリシーの 2 つのタイプの DRM ポリシーが組み込まれています。

- 回数ベースの DRM ポリシー: ダウンロードされた J2ME アプリケーションをデバイス上で実行する回数を最大 *x* 回に制限します。*x* は、ポリシーの作成時に基本管理開発者によって指定された回数です。
- 期間ベースの DRM ポリシー: ダウンロードした J2ME アプリケーションをダウンロードした後デバイス上で実行する期間を、指定した期間に制限します。期間は、年、月、日、時間または分（またはこれらすべて）で指定します。年（365 日）と月（30 日）の処理には、標準の変換が使用されます。

ポリシーの作成とそのポリシーのコンテンツへの関連付けは、両方とも同じです。実際のコンテンツへのポリシーのパッケージ化は、ダウンロード時に行われます。ポリシーは、ユーザーがダウンロードしたアプリケーションをデバイスで起動した後のコンテンツに適用されます。OracleAS Wireless Tools の 1 つである Foundation Manager を使用すると、組込み DRM ポリシーを作成でき、別のツールのコンテンツ・マネージャを使用すると作成したポ

リシーを J2ME アプリケーションに関連付けることができます。コンテンツ・マネージャの詳細は、『Oracle Application Server Wireless 管理者ガイド』を参照してください。

カスタム組込みのデジタル権利ポリシーとコンテンツの拡張機能

OracleAS Wireless は、カスタマイズしたデジタル権利ポリシーとコンテンツの拡張機能を J2ME デバイス（OracleAS Wireless は、MIDP 1.0 準拠のデバイスをサポートしています）で容易に使用できるプラットフォームを提供します。カスタマイズしたデジタル権利ポリシーとコンテンツの拡張機能は、完全に異なる 2 つの機能ですが、OracleAS Wireless の 1 つのフレームワークを使用して実装されます。

デジタル権利ポリシーまたはコンテンツの拡張機能のカスタム実装には、次の 2 つの開発プロセスの手順があります。

1. MIDP プラットフォーム用カスタム・デジタル権利の実装。MIDP プラットフォーム用に `oracle.wireless.me.drm.DRMAgent` クラスを拡張します。この実装は、デジタル権利オブジェクトとも呼ばれます。
2. `oracle.wireless.me.server.tools.drm.DRMPackager` インタフェースの実装。ステップ 1 で開発された権利オブジェクトにコンテンツのパッケージ化論理を実装します。

事例の使用

次の項では、PoweredByPolicy の例を説明します。この例のスプラッシュ画面は、関連するアプリケーションにパッケージ化されています。

ターゲット・モバイル情報デバイス用の PoweredByPolicy.Java

```
package devguide;

import Java.io.IOException;
import Javax.microedition.midlet.*;
import Javax.microedition.lcdui.*;
import oracle.wireless.me.drm.DRMAgent;

/**
* <code>PoweredByPolicy</code> displays a "Powered By" splash
* screen for a packaged content. The product copyright string may
* be set using a property parameter using OracleAS Wireless Server.
*/
public class PoweredByPolicy extends DRMAgent implements CommandListener {

/**
* value of copyright
*/
private String copyright;
```

```
public PoweredByPolicy(MIDlet ctx) {
super(ctx);
copyright = getProperty("copyright");
}

/**
* Displays a splash screen when the application starts up.
*/
public void onStartApp() {
showSplash();
}

/**
* Creates a splash screen and sets it be the current display
*/
private void showSplash() {
Form form = new Form("Powered By:");
try {
form.append(Image.createImage("/devguide/9i.png"));
}
catch(IOException ioe){
form.append("Oracle AS Wireless");
}
form.append(copyright);
form.addCommand( new Command("OK", Command.SCREEN, 1));
form.setCommandListener(this);
Display.getDisplay(context).setCurrent(form);
}

/**
* Resume the normal application logic when user
* attends to the splash screen
*/
public void commandAction(Command c, Displayable d) {
resumeStartApp();
}
} // end of class PoweredByPolicy
```

PoweredByPolicy クラスは、oracle.wireless.me.drm.DRMAgent（以後 DRMAgent と呼びます）の実装を定義します。このクラスは、単一の引数タイプ Javax.microedition.midlet.MIDlet を使用してコンストラクタを定義し、DRMAgent の一般的なコントラクトを満たします。onStartApp() メソッドの実装では、ポリシーのビジネス・ロジックが格納されている showSplash() メソッドをコールすると、スプラッシュ画面が表示されます。最後に、スプラッシュ画面のイベント・ハンドラが、パッケージ化されたアプリケーションが正常に再開するように、resumeStartApp() をコールします。

PoweredByPolicy class のコンパイルと事前検証には、Sun 社の J2ME Wireless Toolkit を使用できます。DRMAgent クラスを含む必須の lib は、*<IASW_HOME>/wireless/lib/j2medrm_demo.jar* から取得できます。コンパイル済で事前検証済のクラス・ファイルまたはクラス・ファイルを含む JAR ファイル（これも事前検証が必要です）は、12-29 ページの「カスタム組込みのデジタル権利ポリシーのパッケージ化」で説明するようにパッケージャを定義することによって、パッケージ化できます。

カスタム組込みのデジタル権利ポリシーのパッケージ化

oracle.wireless.me.server.tools.drm.DRMPackager インタフェース（以後 DRMPackager と呼びます）は、MIDlet スイートのコンテンツを、12-27 ページの「事例の使用」で開発したデジタル権利ポリシーまたはコンテンツの拡張機能にパッケージするための API を定義します。DRMPackager.getInitPropertiesDef() メソッドは、ツールで使用された一連のプロパティを戻すコントラクトを定義します。このツールには、デジタル権利オブジェクトのパラメータ定義を検出するためのユーザー・インタフェースや公開フレームワークなどが含まれます。次に、OracleAS Wireless Runtime は、パラメータの値で init(Properties prop) メソッドをコールします。DRMPackager は、次の API コントラクトを定義します。このコントラクトは、MIDlet スイートのコンテンツをデジタル権利オブジェクトにパッケージ化するために、OracleAS Wireless Runtime によって起動されます。

```
public byte[] packageDRMContent(byte[] content, Properties policyProperties,
Document odrlXml,UserDevice device)
```

DRMPackager の実装を簡略化するために、oracle.wireless.me.server.tools.drm.J2MEDRMPackager（以後 J2MEDRMPackager と呼びます）は、デフォルトの実装を定義してデジタル権利オブジェクトを指定し、setDRMAgentInfo(String agentClassName, String implJarFileName) メソッドを使用して JAR ファイルを含めます。implJarFileName を権利オブジェクトが含まれているファイルの絶対パスに指定できます。それ以外の場合、J2MEDRMPackager は、CLASSPATH で指定した JAR ファイルの検索を実行します。

ODRL を使用して指定した権利オブジェクトは、カスタマイズ不要です。

次のクラスには、権利オブジェクトのパッケージ化論理が記述されています。この実装では、権利オブジェクトを含む JAR ファイルが C:¥temp¥poweredby.jar ファイルにアーカイブされていると想定します。デジタル権利オブジェクト（C:¥temp¥poweredby.jar）を含むファイルを変更した場合、OracleAS Wireless サーバーの再起動は必要ありません。

```
package devguide;

import oracle.wireless.me.server.tools.drm.J2MEDRMPackager;
import oracle.wireless.me.server.tools.drm.DRMPackager;
import Java.util.Properties;

/**
 * Powered By Rights Packager
 */
```

```
public final class PoweredByPolicyPackager extends J2MEDRMPackager {
  private static Properties defaultInitProperties = null;

  static {
    defaultInitProperties = new Properties();
    // initializes the copyright property with a default value
    defaultInitProperties.setProperty("copyright", "Copyright 2003 Oracle
Corporation. All Rights Reserved.");
  }

  private Properties initProperties = null;

  private static PoweredByPolicyPackager instance = new PoweredByPolicyPackager();

  /**
   * Static factory that returns the instance of DRMPackager
   */
  public static DRMPackager getInstance() {
    return instance;
  }

  private PoweredByPolicyPackager() {
    initProperties = defaultInitProperties;
    setDRMAgentInfo("devguide.PoweredByPolicy", "C:¥¥temp¥¥poweredby.jar");
  }

  public Properties getInitPropertiesDef() {
    return defaultInitProperties;
  }
}// end of the class
```

注意： このクラスでは、次のメソッドを実装することによって、DRMPackager の一般的なコントラクトが実装されます。

```
  public static DRMPackager getInstance() {
    return instance;
  }
```

カスタム組込みのデジタル権利ポリシーのデプロイ

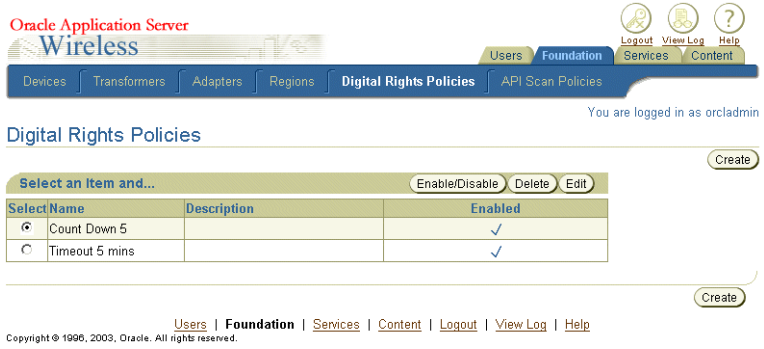
1. カスタム組込みのデジタル権利ポリシーをデプロイする前に、OracleAS Wireless Tools の 1 つである基本管理開発者を使用して新しいデジタル権利ポリシーを作成する必要があります。基本管理開発者の詳細は、『Oracle Application Server Wireless 管理者ガイド』を参照してください。

> **注意：** 基本管理開発者を使用するには、Foundation Manager またはスーパーユーザーのロールが付与されている必要があります。

新しいデジタル権利ポリシーを作成する手順は、次のとおりです。

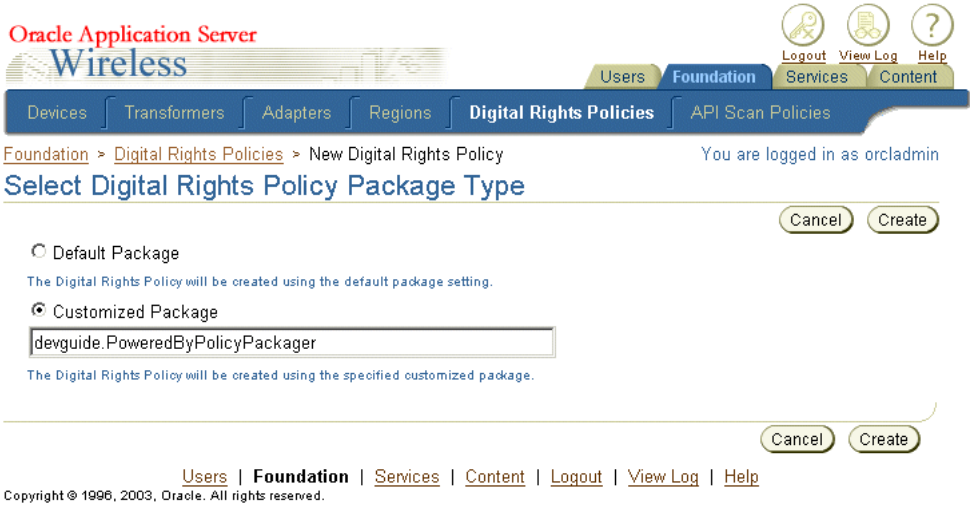
1. 参照ページ（図 12-1）で、**「作成」**をクリックします。「デジタル権利ポリシー・パッケージ・タイプの選択」画面が表示されます（図 12-2）。

図 12-1 DRM ポリシーの参照ページ



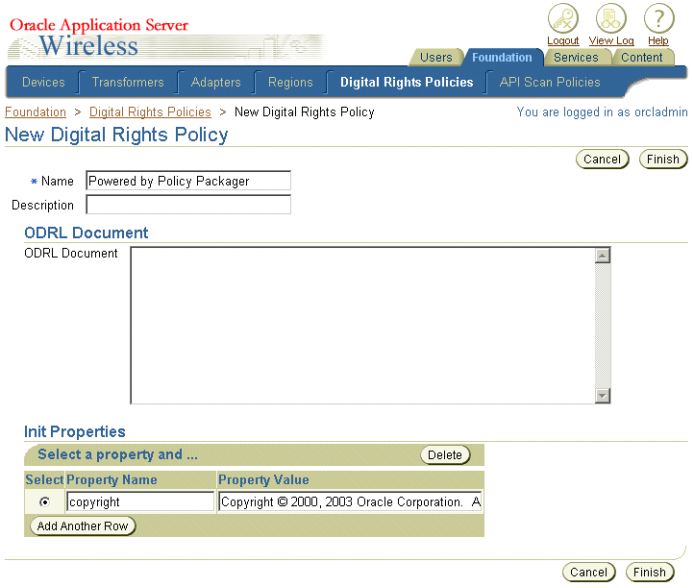
2. **「カスタマイズ済パッケージ」**オプションを選択します。
3. パッケージ名を入力します。ポリシーが `DRMPackager` 実装によってサポートされている場合は、初期化（init）パラメータと ODRL 文書も定義できます。図 12-3「新しいデジタル権利ポリシーの作成」を参照してください。

図 12-2 新しいデジタル権利ポリシーの作成



4. 「**作成**」をクリックします。「新しいデジタル権利ポリシー」画面が表示されます（図 12-3）。
5. ポリシー名を入力します。
6. 「**作成**」をクリックします。

図 12-3 新しいデジタル権利ポリシーの作成



デジタル権利オブジェクトを作成した後は、このオブジェクトを MIDlet アプリケーションのアプリケーション・リンクに関連付けることができます。サービス・マネージャを使用して J2ME MIDlet アプリケーションを作成し、次に、その MIDlet へのアプリケーション・リンクをコンテンツ・マネージャを使用して作成します。このリンクは、アプリケーションをカスタマイズして、ユーザー・グループに公開する手段です。MIDlet アプリケーションの作成方法は、5-23 ページの「J2ME アプリケーションの作成」を参照してください。アプリケーション・リンクの作成方法は、『Oracle Application Server Wireless 管理者ガイド』を参照してください。

図 12-4 デジタル権利ポリシーの J2ME アプリケーションへの関連付け

J2ME プロビジョニング・サーバー

OracleAS Wireless J2ME プロビジョニング・サーバーを使用すると、サービス開発者は、J2ME アプリケーションをアップロード、編成およびダウンロードできます。アップロードした J2ME アプリケーションは、コンテンツ・マネージャを使用するダウンロード用アプリケーションとしてユーザー・グループに公開できます。ユーザーに公開したサービスは、そのユーザーに配布された他のアプリケーションとともに、Wireless ユーザーのアプリケーション・ツリーで使用できます。

アプリケーション・モデル

OracleAS Wireless J2ME プロビジョニング・フレームワークのリポジトリは、コンテンツ・リポジトリとフレームワークで構成されています。このフレームワークは、アップロード済のデジタル著作権管理ポリシーとコンテンツ間の関連付けを作成する手段を提供します。コンテンツと DRM ポリシーの関連付けは、サービスの作成時に実行されます。コンテンツのダウンロード・トランザクションはすべて、監査表に記録されます。

コンテンツ・リポジトリには、すべてのアップロード済コンテンツに関する配信可能なコンテンツ情報が格納されます。この情報には、公開 API を介してアクセスできます。表 12-2 にこの情報を示します。

表 12-2 配信可能なコンテンツ

情報	説明
名前	コンテンツ名。
バージョン	コンテンツのバージョン。
表示名	コンテンツの表示名。

表 12-2 配信可能なコンテンツ（続き）

情報	説明
説明	コンテンツの説明。
検証時間	コンテンツが API スキャン機能を使用して事前検証された時間。
ステータス	コンテンツが有効かどうかを示します。
所有者	コンテンツの所有者。

配信可能なコンテンツには、実際のコンテンツのメタ情報が含まれています。これは、`DeliverableContentItem` オブジェクトとして使用できます。（表 12-3 で説明）。

表 12-3 DeliverableContentItem

情報	説明
MIME タイプ	コンテンツの MIME タイプ （JAR タイプまたは JAD MIME タイプ）。
コンテンツ・バイナリ	バイナリ形式で格納された実際のコンテンツ。
コンテンツ・サイズ	コンテンツのサイズ。
監査情報	作成時間、更新時間およびユーザー情報などの監査情報。

プロビジョニングのトランザクションは監査表、PROVISIONING_TRANSACTION_LOG に記録されます（表 12-4 で説明）。これらのトランザクションは、システム・マネージャでは、ランタイム・メトリックとして表示されます。

表 12-4 PROVISIONING_TRANSACTION_LOG

列名	列タイプ	説明
INSTANCE_NAME	VARCHAR2 (256) (NOT NULL)	このコンテンツを処理したプロビジョニング・サーバーのインスタンス名。
HOST_NAME	VARCHAR2 (256)	サーバーのホスト名。
USER_DOWNLOAD_ID	NUMBER	このダウンロード・トランザクションの一意の内部 ID。
USER_NAME	VARCHAR2(2000)	このコンテンツにプロビジョニングしたユーザー名。
SERVICE_NAME	VARCHAR2 (256)	コンテンツのアクセスに使用したサービス。
APPLICATION_NAME	VARCHAR2 (256) (NOT NULL)	このコンテンツをカプセル化したアプリケーション。
APPLICATION_TYPE	VARCHAR2 (256)	アプリケーション・タイプ。例 : J2ME

表 12-4 PROVISIONING_TRANSACTION_LOG（続き）

列名	列タイプ	説明
CONTENT_NAME	VARCHAR2 (256) (NOT NULL)	コンテンツ名。コンテンツを一意に識別するコンテンツ名とバージョン。
CONTENT_VERSION	VARCHAR2 (100) (NOT NULL)	コンテンツのバージョン。コンテンツを一意に識別するコンテンツ名とバージョン。
MIME_TYPE	VARCHAR2 (256) (NOT NULL)	コンテンツの MIME タイプ。
DRM_POLICY_NAME	VARCHAR2 (256)	コンテンツのダウンロードで使用したこのサービスに関連付けられている DRM ポリシー（ある場合）。
NUMBER_OF_DOWNLOADS	NUMBER	このコンテンツをユーザーがダウンロードした合計回数。
DEVICE_ID	NUMBER	ダウンロード・デバイスの情報。
DOWNLOAD_TIME_STAMP	DATE	ダウンロード操作のタイムスタンプ。
DOWNLOAD_INTERNAL_STATUS	VARCHAR2 (256)	ダウンロードの成否を示す内部ステータス・コード。
DOWNLOAD_DISPLAY_STATUS	VARCHAR2 (256)	ダウンロードの成否を示す説明的なステータス・コード。

フック

フレームワークを使用すると、実装者が様々なステージでフックをプラグインできるため、ダウンロード操作を追跡および制御できます。カスタム実装のフックは、システム・マネージャを使用してプラグインできます。このフックはシングルトン・クラスであることが必要で、そのオブジェクト参照を戻すための `getInstance()` メソッドが含まれている必要があります。

- `Pre download Hook`: 実際のダウンロードが発生する前にアプリケーションが起動されると起動されます。このフック・インタフェースは、ユーザー情報とコンテンツ情報を含むユーザーのダウンロード・ステータス・オブジェクトに対して提供されます。このフックは、true または false ステータスを戻すことで、ユーザーにダウンロードの続行または操作の中断をそれぞれ許可することができます。

  `ProvisioningPreDownloadHook` インタフェースは、パブリック・パッケージ `oracle.panama.rt.hook` に定義します。次に例を示します。

```
public interface ProvisioningPreDownloadHook {
    /** Delegate additional processing of this download
     * @param UserDownloadStatus The download status object encapsulates the
current download transaction i.e. user, application, content, version, mime
type etc.
     * @return boolean to indicate successful hook processing
```

```
     * @see oracle.panama.model.UserDownloadStatus
     */
     public boolean processRequest (
              UserDownloadStatus uds
              );


}
```

- Post download Hook: ポスト・ダウンロード・サイクルの 2 つのステージで起動されます。1 つはダウンロードが成功した後、もう 1 つはダウンロードの成功がデバイスに通知された後です。

  このフックは、ユーザーのダウンロード・ステータス・オブジェクトを提供します。このオブジェクトには、ユーザーとコンテンツの情報が含まれています。フックの実装によって、どちらのステージでダウンロード操作を請求または監査するかを決定できます。

  ProvisioningPostDownloadHook インタフェースは、パブリック・パッケージ oracle.panama.rt.hook に定義します。次に例を示します。

```
public interface ProvisioningPostDownloadHook {
    public static final int POST_DOWNLOAD = 1;
    public static final int POST_NOTIFY = 2;
         public boolean processRequest (
               UserDownloadStatus uds,
               HttpServletRequest request,
               int hookType /* determines if it is a post notify or a
post download hook */
              );
}
```

J2ME アプリケーションのアップロード

J2ME アプリケーションは、サービス・マネージャを使用して作成します。このツールにアクセスするには、サービス・マネージャまたはスーパーユーザーのロールが付与されている必要があります。

J2ME MIDlet アプリケーションを作成する手順は、次のとおりです。

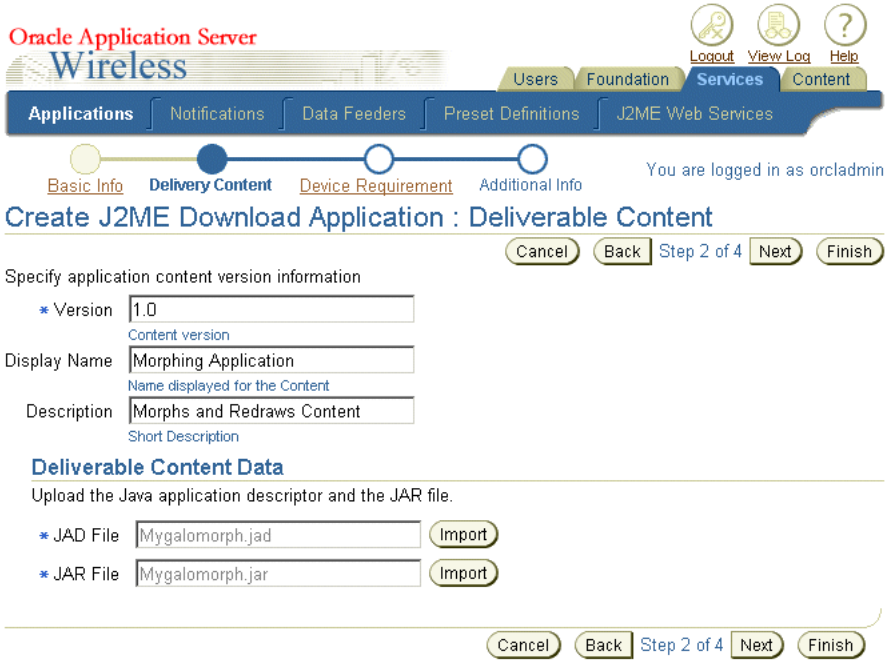
1. 参照画面で**「アプリケーションの作成」**をクリックします。
2. アプリケーション・タイプの選択画面から、アプリケーション・タイプとして「J2ME Midlet」を選択します。
3. MIDlet アプリケーションの名前を入力します。たとえば、「Morphing」（図 12-5 の図のように）と入力します。

   この URL パラメータは、組み込まれているダウンロード・サービス・マネージャの JSP を指し示します。同じ機能を提供するカスタム JSP を入力することもできます。

図 12-5 MIDlet アプリケーションに関する基本情報の入力

4. 「次」をクリックして J2ME コンテンツの詳細を入力します。(図 12-6)。

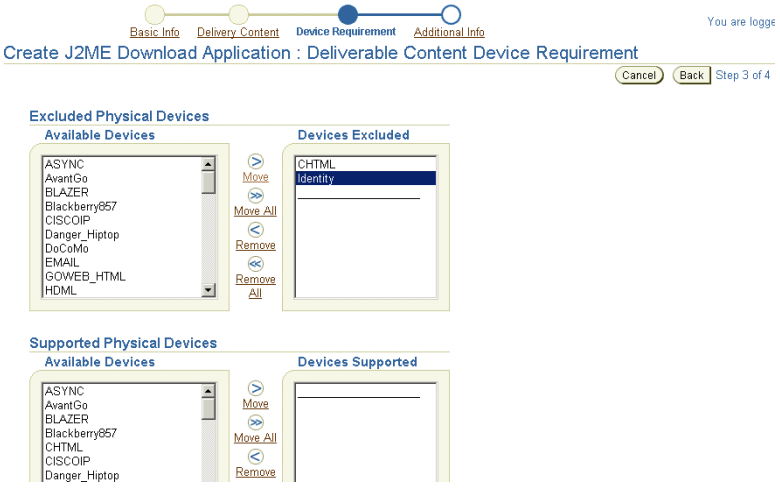
図 12-6 コンテンツ詳細の入力



5. バージョン番号、表示名およびアプリケーションの説明を入力します。
6. J2ME MIDlet アプリケーションの JAD と JAR ファイルをアップロードします。

7. 「**次**」をクリックします。「デバイス要件」ページが表示されます（図 12-7）。

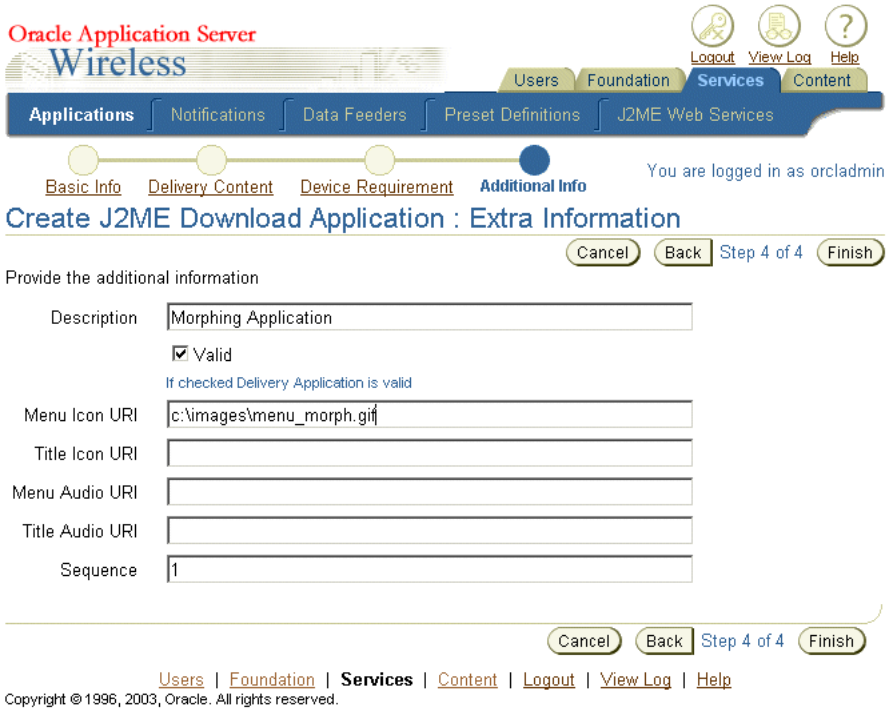
図 12-7 J2ME アプリケーションをサポートするデバイスの選択



8. 必要に応じて、アプリケーションをサポートできないデバイスを除外します。
9. オプションで、デバイスのヒープ・サイズ要件を設定します。

10. **「次」**をクリックします。「追加情報」画面が表示されます（図 12-8）。
11. **「有効」**オプションを選択します。
12. 表示メニュー・アイコンのファイル位置など、アプリケーションに関する追加情報を指定します。
13. **「終了」**をクリックして、J2ME MIDlet アプリケーションを完了します。

図 12-8 J2ME アプリケーションに関する追加情報の入力



J2ME アプリケーションの公開

コンテンツ・マネージャを使用すると、J2ME アプリケーションをユーザー・グループに公開できます。サービス・マネージャを使用して作成されたアプリケーションは、コンテンツ・マネージャを使用して作成したアプリケーション・リンクとして公開されます。サービス設計者が作成したアプリケーション、つまりマスター・アプリケーションは、アプリケーション・リンクのコアとなります。アプリケーション・リンクを使用すると、アプリケーションをカスタマイズできます。たとえば、ユーザー・グループやロケーションに対してアプリケーションをカスタマイズできます。詳細は、『Oracle Application Server Wireless 管理者ガイド』を参照してください。

J2ME アプリケーションを公開する手順は、次のとおりです。

1. コンテンツ・マネージャの参照画面で、**「アプリケーション・リンクの追加」**をクリックします。
2. 公開対象のアプリケーションを選択します。たとえば、「Morphing アプリケーション」を選択します。
3. **「次」**をクリックします。
4. アプリケーション・リンクの名前を入力します。
5. 必要に応じて、DRM ポリシーを選択します。デフォルトでは、ユーザーはダウンロードしたアプリケーションを無制限に使用できます。詳細は、12-27 ページの「カスタム組込みのデジタル権利ポリシーとコンテンツの拡張機能」を参照してください。
6. デフォルトを受け入れて、**「終了」**をクリックしてアプリケーション・リンクを完了できます。「追加情報」画面で、請求用コスト情報の 0（ゼロ）以外の値などの情報を必要に応じて入力します。**「参照可能」**オプションおよび他の情報を必要に応じて選択します。

J2ME アプリケーションのダウンロード

アクセス（使用）対象の J2ME については、そのアプリケーションをユーザー・グループに割り当てる必要があります。アプリケーションを使用できるのは、J2ME アプリケーションを割り当てたグループに属しているユーザーのみです。さらに、グループ・メンバーには 1 つ以上のデバイス・アドレスが必要です。

アプリケーションにアクセスするには、ユーザーは次のような URL を使用して Wireless and Voice Portal にログインする必要があります。

http://yourwirelessserver:7777/ptg/rm

図 12-9 モーフィング・サービスの例

「Morphing アプリケーション」を選択すると（図 12-9 のように）、JSP は、このアプリケーション用にアップロードしたすべての J2ME コンテンツのリストを表示します。**「ダウンロード」**をクリックして、選択したコンテンツをデバイスにダウンロードします。

Customization Portal で、**「アプリケーション」**をクリックし、次に**「ダウンロード履歴の表示」**をクリックすると、ダウンロード履歴ログを表示できます。

13

Web スクレイピング

この章では、トランスコーディングについて説明します。項ごとに様々なトピックを記載しています。各項の内容は、次のとおりです。

- Web スクレイピングの概要
- Web クリッピング
- Wireless アプリケーションの作成
- 既存のトランスコーディング・テクノロジからの移行
- Web クリッピング・サービスのカスタマイズ
- OracleAS Wireless 管理者の管理タスク
- WML Translator

Web スクレイピングの概要

Web 上で使用可能なアプリケーションの大多数は、コンテンツを特定タイプのデバイスに固有のフォーマットでレンダリングします。Web スクレイピングを使用すると、特定のマークアップ言語用に開発されたアプリケーションを他のデバイスで使用するように再フォーマットできます。Web スクレイピングには、PC ブラウザ・アプリケーションに再度目的を設定する Web クリッピングおよび WML アプリケーションを再利用するための WML Translator が組み込まれています。

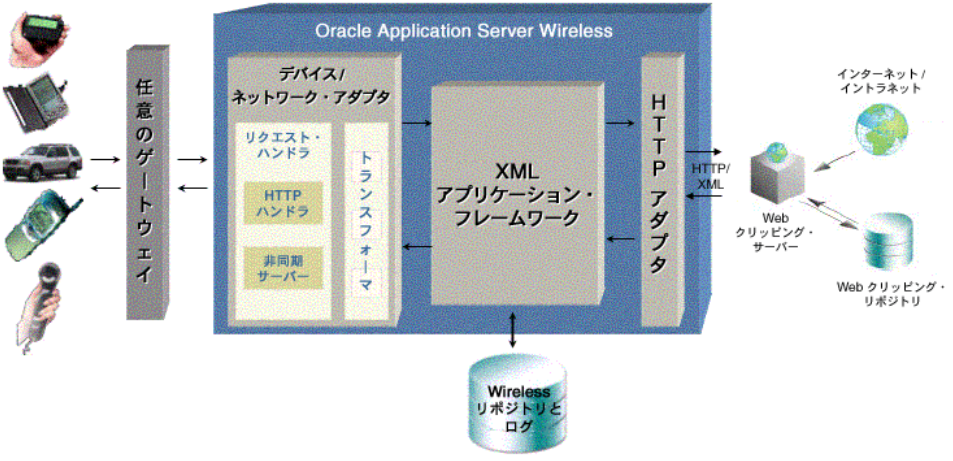
Web クリッピング

この項では、OracleAS Wireless で使用可能な Web クリッピング・アプリケーションの機能について説明します。ワイヤレス・デバイス上で Web クリッピング・アプリケーションを作成、カスタマイズおよび表示するために必要な OracleAS Wireless 管理者とユーザーのロールについても説明します。プロキシ設定とセキュリティの構成方法などの一部の管理タスクに関する情報も含まれます。

概要

OracleAS Wireless に関連する Web クリッピング・サーバーのアーキテクチャを図 13-1「OracleAS Wireless コアに関連する Web クリッピング・サーバーのアーキテクチャ」に示します。

図 13-1 OracleAS Wireless コアに関連する Web クリッピング・サーバーのアーキテクチャ



Web クリッピング・サーバーを使用すると、OracleAS Wireless 管理者は、Web コンテンツをクリップおよびスクレイプして、Web クリッピング・アプリケーションを作成できます。このアプリケーションは Web クリッピング・サーバーのリポジトリに永続的に保存されます。モバイル・ワイヤレス・デバイスを使用しているユーザーが、特定の Web クリッピング・アプリケーションを表示するリクエストを開始すると、図 13-2「Web クリッピング・アプリケーション」に示すように、そのアプリケーションは、HTTP アダプタによって取得され、処理およびモバイル・デバイスへの転送のために OracleAS Wireless コアに配信されます。

図 13-2 Web クリッピング・アプリケーション

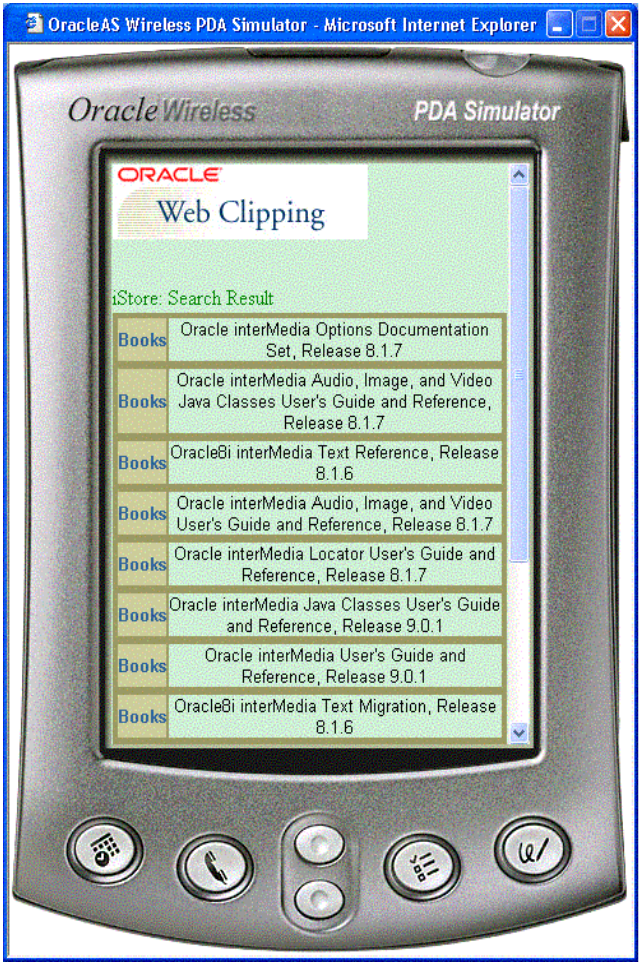
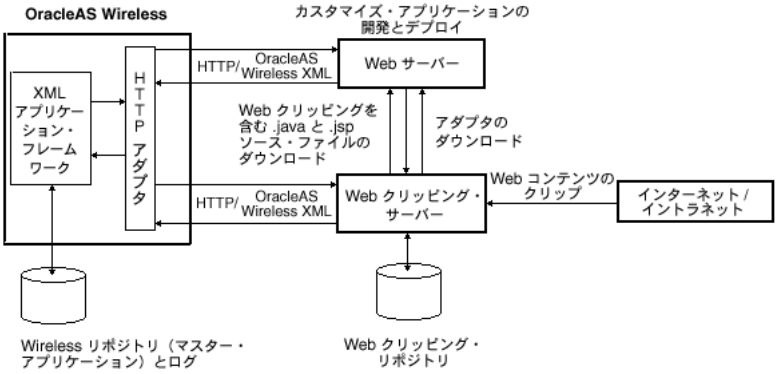


図 13-3「Web クリッピング・サーバー・アーキテクチャの詳細」に Web クリッピング・サーバーの詳細を示します。Web クリッピング・コンテンツは、インターネットまたは大きな組織全体に分散しているイントラネットの Web サイトからクリップしてスクレイプできます。OracleAS Wireless を使用すると、Web クリッピングを含むアプリケーションを作成して、Wireless リポジトリに保存できます。さらに、ユーザーのニーズに最適な開発環境を最初に識別することによって、Web クリッピングを最大限に活用するカスタマイズ・アプリケーションを作成できます。選択した開発環境に応じて、必要なアーカイブ・ファイルをその開発環境にダウンロードしてデプロイする必要があります。次に、必要なコンテンツをクリップします。その後、Java または JSP スタブ・ファイルをダウンロードして開発を開始します。

図 13-3 Web クリッピング・サーバー・アーキテクチャの詳細



各開発環境で使用可能なアーカイブ・ファイルを次に示します。

- J2EE: Web クリッピングの Bean API ではなく、Sun 社の標準 JCA Common Client Interface（CCI）を使用している J2EE（OC4J）環境での開発用およびリリース 9.0.3 以上の OC4J バージョンでの開発用。JCA Resource Adaptor Archive（RAR）ファイル `webclipping.rar` をダウンロードします。
- J2SE: J2EE（OC4J）環境およびスタンドアロン Java 2 SDK 環境での開発用。Java ライブラリ（JAR）ファイル `wcbean.jar` をダウンロードします。

J2EE または J2SE 環境では、アプリケーションをスタンドアロン Java アプリケーションまたはその他の JSP アプリケーションの一部としてカスタマイズしてデプロイできます。あるいは、これらのアプリケーションを Wireless アプリケーションに作成して Wireless リポジトリに格納できます。また、このコンテンツをクリップすることもできます。

Web クリッピング・アプリケーションを作成する場合、OracleAS Wireless 管理者は、Web ブラウザを使用して必要なコンテンツを含む Web ページにナビゲートします。次に、そのページのクリップおよびスクレイプの対象部分を選択して属性を設定します。Web クリッピングでフォーム・ベースの送信を使用する場合は、入力パラメータを表示し、アプリケーションを保存した後、そのアプリケーションをテストします。

Web クリッピング・アプリケーションは、次の操作をサポートしています。

- フォーム・ベースと JavaScript ベースの送信および Cookie ベースのセッション管理付きの HTTP Basic 認証と Digest 認証など、様々な形式のログイン・メカニズムによるナビゲーション。
- クリッピングのファジー・マッチング。Web クリッピングの順序がソース・ページ内で変更された場合やその文字フォント、サイズまたはスタイルが変更された場合、その変更は、Web クリッピング・サーバーによって正しく識別され、Web クリッピング・アプリケーションのコンテンツとして配信されます。
- 広範囲にわたる Web コンテンツの再利用。このコンテンツには、HTML 4.0.1、JavaScript、アプレットで作成されたページのサポート、および HTTP GET および POST（フォーム送信）メソッドによって取得したプラグイン対応コンテンツが含まれます。
- HTML 4.0.1 ページからのページ・コンテンツのクリッピング。次のクリッピングが含まれます。
  - <table>、<td>、<img>、、 の各タグ付きコンテンツのクリッピング。
  - <head> スタイルとフォント、カスケード・スタイルシート（CSS）の保持。
  - UTF-8 準拠のキャラクタ・セット。
  - ハイパーリンク（HTTP GET）、フォーム送信（HTTP POST）、フレーム、URL リダイレクションを介したナビゲーション。
- 適切なサーバー証明書を取得している場合は、HTTPS ベースの外部 Web サイトにナビゲートしてクリップできます。
- 次の方法による既存 Web コンテンツからの各国語セット（NLS）。最初に、HTTP ヘッダーの「Content-Type」で charset 属性をチェックします。この属性が存在する場合、これは HTML ページの文字コードであるとみなします。存在しない場合は、次にそのページの HTML META タグをチェックし、文字コードを判断します。HTML META タグがない場合、デフォルトで ISO-8859-1 文字コードに設定されます。さらに、URL と URL パラメータでの NLS もサポートします。
- Netscape Navigator リリース 6 および 7、Microsoft Internet Explorer リリース 5.0、5.5、6.0 およびそれ以上などのブラウザ。
- 大部分の HTML 以外の要素。これには、アプレット、プラグイン・コンテンツ（埋込み QuickTime ビデオ、Macromedia Flash 表現など）およびクライアント側 JavaScript（v1.2）が含まれます。

次に、Web クリッピングの既知の制限を示します。

- NLS または国際化の制限には、次のものが含まれます。
  - キャラクタ・セットはメタ・タグまたは HTTP ヘッダーで指定する必要があります。指定しない場合、スタジオとプロバイダは、UTF-8 キャラクタ・セットにデフォルト設定されます。
  - UTF-8 でサポートされない言語は正しく表示されません。
  - 2 つ以上のキャラクタ・セットがページ上に指定されている場合、文字は正しく表示されません。
- Web クリッピング・スタジオの**「戻る」**ボタンと**「転送」**ボタンは、個別のフレーム内でなく、ページ全体で動作します。
- プラグイン・コンテンツ（Macromedia Flash など）からリンクされたページ。
- 複数の値を持つ HTTP パラメータのカスタマイズ。
- ログイン・ページ用のユーザー名 / パスワードの JavaScript ベースの暗号化。

次のページ・コンテンツは、Web クリッピングではクリップできません。

- `doc.write()` を含むグローバル JavaScript 文。
- 追加の JavaScript を記述する区分化された Web クリッピングでの JavaScript。

Web クリッピング・サーバーは、クリップした Web コンテンツを Web クリッピング・アプリケーションのコンテンツとして提供およびレンダリングします。Web クリッピング・スタジオを使用すると、OracleAS Wireless 管理者および Wireless アプリケーション・ユーザーは次のことを実行できます。

- Web コンテンツの参照。
- 選択したターゲット・ページの複数セクションへの分割。
- Web コンテンツ内のクリップ対象部分の正確な選択。
- クリップしたコンテンツのプレビュー。
- クリップしたコンテンツのスクレイプ。
- Web クリッピング・アプリケーション属性の設定、フォーム・ベース送信での入力パラメータのパラメータ化、および Web クリッピング・アプリケーションの保存。
- フォーム・ベース送信を使用している場合、Web クリッピング・アプリケーションのテストおよび初期入力値の変更とそのテスト。

Web クリッピング・アプリケーションの定義はすべて、Oracle Application Server Infrastructure データベースに永続的に格納されます。パスワードなどの保護情報は、データ暗号化規格（DES）に従い、Oracle 暗号化テクノロジを使用して暗号化された形式で格納されます。

開始

Web クリッピング・サーバーは、OracleAS Wireless の一部として自動的にインストールされます。OracleAS Wireless 管理者は、Wireless Web ツールの**「アプリケーション : フォルダの参照 :」**ページの下のリポジトリにある Web クリッピング・サーバーにアクセスできます。

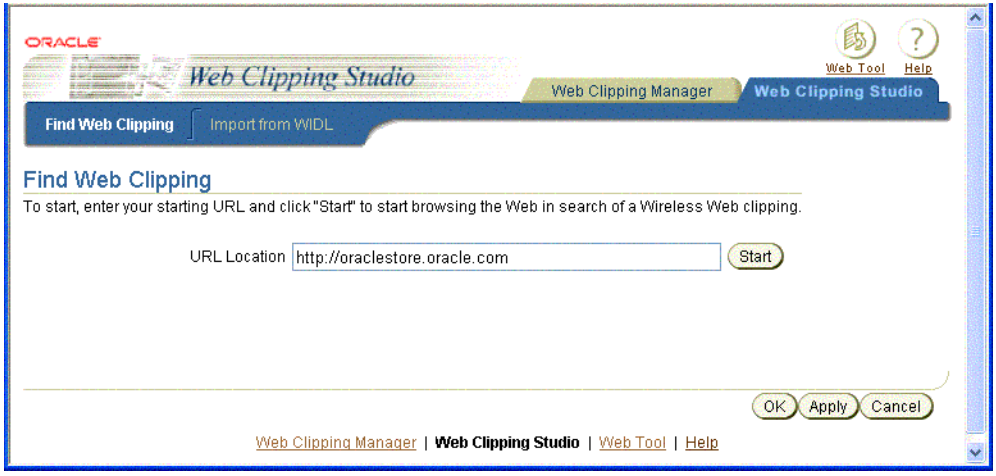
OracleAS Wireless 管理者は、開始前に一部の構成タスク（13-33 ページの「OracleAS Wireless 管理者の管理タスク」を参照）を実行する必要がある場合があります。

Web クリッピング・アプリケーションの作成

Web クリッピングを作成する手順は、次のとおりです。

1. OracleAS Wireless Web ツールに OracleAS Wireless 管理者でログインします。
2. **「サービス」**タブをクリックします。
3. **アプリケーション・フォルダの参照 : マスター・**ページで**「アプリケーションの作成」**をクリックします。
4. **「作成するアプリケーションの選択」**ページの**「Web クリッピング・アプリケーション」**ラジオ・ボタンをクリックしてから、**「作成」**をクリックします。
5. **「Web クリッピングの管理」**ページで**「Web クリッピングの追加」**をクリックします。
6. Web クリッピング・スタジオの**「Web クリッピングの検索」**ページの**「URL ロケーション」**フィールドに、最初の Web ページのロケーションを入力します。これによってクリップする実際の Web ページにアクセスできます。たとえば、図 13-4「Web クリッピングの検索」に示すように、後続の手順では、`http://oraclestore.oracle.com` を開始点とします。

図 13-4 Web クリッピングの検索



注意： WIDL ファイルをインポートして Web クリッピングを作成するには、**「WIDL からインポート」**タブを選択します。**「WIDL からインポート」**ページが表示されます。詳細は、13-29 ページの「既存のトランスコーディング・テクノロジからの移行」を参照してください。

---

7. **「開始」**をクリックします。

   ストアの選択を指示するページが表示されます。

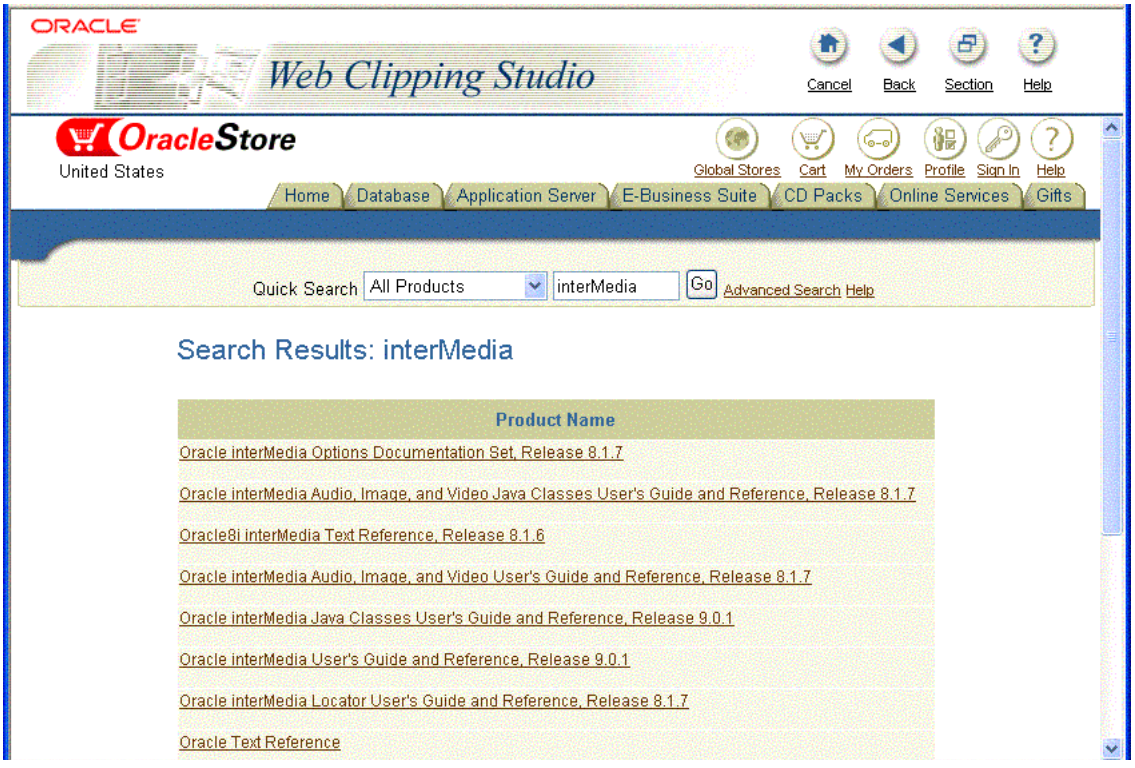
8. **「アメリカ合衆国」**または任意の国を選択してクリックします。

   図 13-5「Web クリッピング・スタジオでの ORACLESTORE.ORACLE.COM の参照」に示すように、選択した国の OracleStore Web ページが、Web クリッピング・スタジオ内に表示されます。

図 13-5 Web クリッピング・スタジオでの ORACLESTORE.ORACLE.COM の参照

クリップするコンテンツを参照するために Web ページのハイパーリンクをクリックするたびに、使用したナビゲーション・リンクが記録されます。たとえば、開始ポイントに URL `oraclestore.oracle.com` を使用し、次に**「アメリカ合衆国」**をクリックして国を選択すると、クイック検索ボックスに名前を入力し、**「実行」**をクリックして、名前で製品を検索できます。たとえば、機能名「interMedia」を入力し、**「実行」**をクリックします。図 13-6「クリップするコンテンツの参照」のように、入力した名前を含むすべての製品が検索結果に表示されます。

図 13-6 クリップするコンテンツの参照



注意： 最終的に Web クリッピングに貢献しない参照操作は、すべて破棄されます。したがって、将来のテスト・モード時の再生用に記録されるのは重要な参照操作のみです。破棄されたリンクにはアクセスしません。図 13-6「クリップするコンテンツの参照」を参照してください。この例に表示されている URL は、開始 URL、1 つの追加 URL およびフォームを含む URL の 3 つのみであることに注意してください。

注意： HTTP Basic 認証または Digest 認証が必要な Web サイトの場合は、ユーザー名とパスワード情報を要求するフォームが表示されます。このエンコードされた認証情報は、参照情報の一部として記録されます。

9. 参照モードの Web クリッピング・スタジオで、クリップするコンテンツを検索した後は、**「セクション」**をクリックします。これによって、図 13-7「Web クリッピング・スタジオでのクリップ可能セクションへのページの分割」のように、ターゲットの Web ページが複数のクリップ可能なセクションに分割されます。

図 13-7 Web クリッピング・スタジオでのクリップ可能セクションへのページの分割

注意：「セクション」をクリックした後は、表示されたページでリンクを参照することはできません。ナビゲーションを続行する場合は、**「セクション解除」**をクリックします。

---

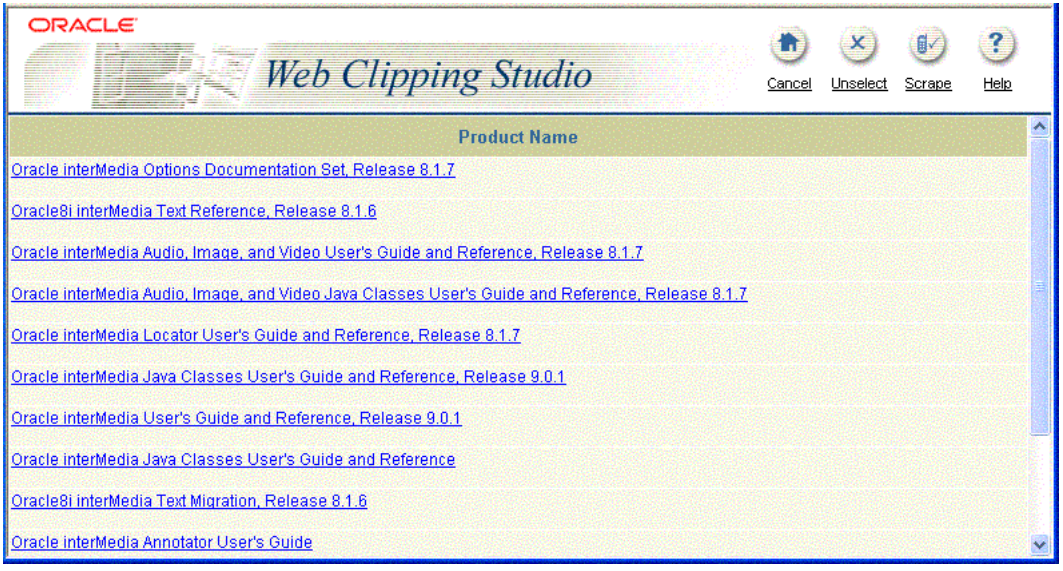
10. セクション・モードの Web クリッピング・スタジオで、クリップする Web コンテンツの「製品名」セクションを検索し、**「選択」**をクリックします。

注意： ページ上の Web コンテンツは、複数の「選択」セクションに区分化されています。クリックできるのは、1 つの**「選択」**セクションのみです。クリッピングとして選択されるのは、選択した**「選択」**セクション・アイコンの直下のコンテンツのみです。プレビュー・モードを続行すると、クリップしたコンテンツを表示できます。選択したセクションが必要なセクションでなかった場合は、**「選択解除」**をクリックすると、再度セクション・モードに戻り、クリップする別のセクションを選択できます。選択可能なセクション数を増やすには、**「ズーム・イン」**をクリックします。たとえば、**「ズーム・イン」**をクリックして、ネストした表の 1 レベル下にドリルダウンします。選択可能なセクション数を減らすには、**「ズーム・アウト」**をクリックします。

---

11. 図 13-8「Web クリッピング・スタジオでの Web クリッピングのプレビュー」のように、プレビュー・モードの Web クリッピング・スタジオで、「製品名」セクションのプレビューが検索結果とともに表示されます。検索結果が必要なセクションの場合は、**「スクレイプ」**をクリックしてスクレイプ・モードに進みます。

図 13-8 Web クリッピング・スタジオでの Web クリッピングのプレビュー



Web クリッピング・アプリケーションでクリップしたセクションを使用しない場合は、**「選択解除」**をクリックして、そのセクションを含むページに戻ります。そのページで別のセクションを選択するか、**「セクション解除」**をクリックして別のページにナビゲートします。

12. スクレイプ・モードで、最初の出力の終わりにあるチェック・ボックスをクリックします。選択した各出力には、下部のフレームの**「値」**と**「名前」**の表の下に値の行が表示されます。ここで出力の名前を指定できます。**「名前」**列で、出力に対して Book などのわかりやすい名前を入力します。

注意： 名前に空白を含めることはできません。

セル付きの表には、特殊なスタック・チェック・ボックスが表示され、セル内の全テキストを出力として選択できます。

出力（最初のチェック・ボックス）を選択すると、選択した最初の出力が同様の出力のコレクション内にある最初の項目を表しているかぎり、全体の繰返しが可能なグループが作成されます。これは、反復性を適用できるのは、同様の出力のコレクションのみであることを意味します。反復レベルを増やして、同様の出力の全コレクションを含めるには、図 13-9「スクレイプ・モード」のように、**「追加」**を 1 回クリックします。**「続行」**をクリックして、Web クリッピング・アプリケーションの属性ページに進みます。

各項目の後のチェック・ボックスは、スクレイプ可能な出力として、その項目を選択できることを示します。1 つ以上の出力項目が、同様の出力のコレクションとなる場合があります。同様の出力の最初の行からスクレイプする出力を選択できます。選択した各出力には、**「値」**と**「名前」**の表の下に値の行が表示されます。ここで、**「名前」**列に出力の名前を指定できます。各出力には、スクレイプ元の列名に基づいた名前などのわかりやすい名前を指定します。**「追加」**をクリックすると、スクレイプ対象の同様の出力のコレクション全体を迅速に選択できます。**「追加」**をクリックすると、選択した出力の連続したグループのチェック・ボックスにチェックマークが付けられ、スクレイプ対象に指定されます。個別のチェック・ボックスをチェックするのではなく、スクレイプする連続したグループの同様の出力を出力として選択することが、最も早い選択方法です。

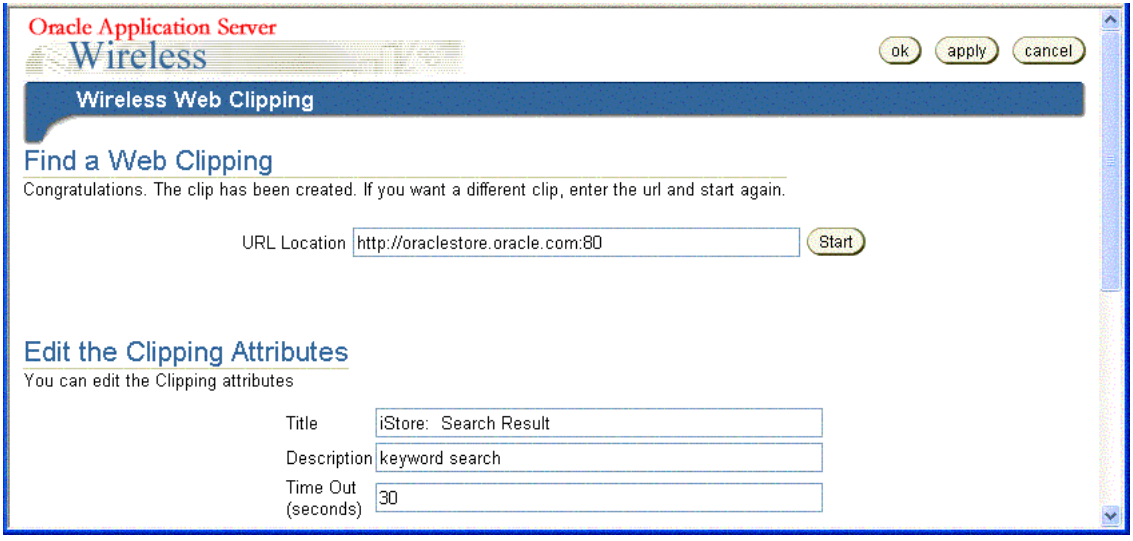
Web クリッピング・アプリケーションとその属性を保存した後は、**アプリケーション・フォルダの参照 : マスター・**ページに戻ります。

図 13-9 スクレイプ・モード

13. 「**Web クリッピングの検索**」ページの「**クリッピング属性の編集**」セクションで、Web クリッピング・アプリケーションの属性を設定できます。

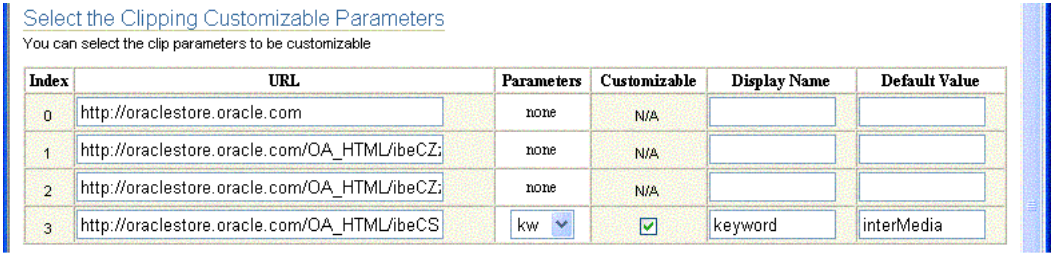
    a. **タイトル、説明、タイムアウト（秒）**の各 Web クリッピング・アプリケーションの属性を指定します。詳細は、ヘルプと図 13-10「Web クリッピング・アプリケーションのオプションの選択とクリッピング属性の編集」を参照してください。この図は、属性ページの上部 3 分の 1 を示しています。「**説明**」フィールドには、「`keyword search`」と入力します。

図 13-10 Web クリッピング・アプリケーションのオプションの選択とクリッピング属性の編集



b. OracleAS Wireless 管理者が、Web クリッピング・アプリケーションのコンテンツのクリッピング中にフォーム・ベース送信を実行した場合は、**「カスタマイズ可能なクリッピング・パラメータの選択」**というタイトルの見出しが、**「Web クリッピング」**ダイアログ・ボックスに表示されます。パラメータをカスタマイズするには、図 13-11「Web クリッピング・アプリケーションのフォーム入力情報のカスタマイズ」のように、表示された表から必要なカスタマイズ可能なパラメータを選択します。この図は、属性ページの中間 3 分の 1 を示しています。ページ・ビューアにカスタマイズ可能として表示する各パラメータを検索します。このためには、**「パラメータ」**列のドロップダウン・ボックスで必要なパラメータを選択してから、**「カスタマイズ可能」**列でそのボックスを選択します。これを実行すると、OracleAS Wireless 管理者は、入力値をカスタマイズして、選択した各パラメータをパーソナライズできます。詳細、特にユーザー名とパスワードのパラメータをカスタマイズしない理由は、OracleAS Wireless ヘルプの Web クリッピング・アプリケーション属性の編集とパラメータのカスタマイズに関するヘルプ・トピックを参照してください。

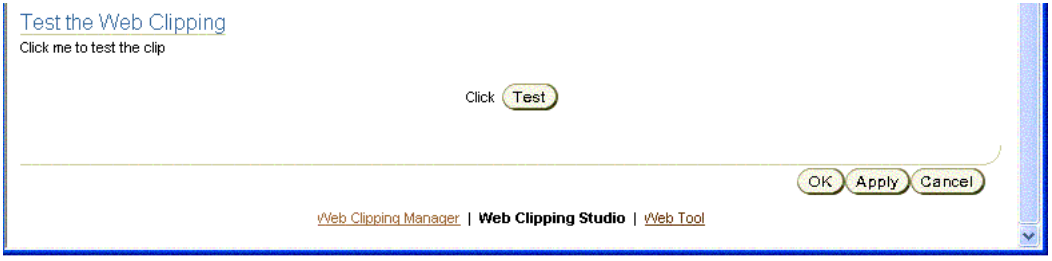
図 13-11 Web クリッピング・アプリケーションのフォーム入力情報のカスタマイズ



c. 索引の値が 3 の行にあるパラメータ kw に対して、その**「表示名」**を kw から keyword に変更し、この名前を Wireless アプリケーション・ユーザーが認識しやすいようにします。次に、**「カスタマイズ可能」**ボックスをクリックして、Wireless アプリケーション・ユーザーがパラメータの入力値をカスタマイズできるようにします。

d. Web クリッピング・アプリケーションをテストするには、図 13-12「Web クリッピング・アプリケーションのテスト」のように、**「Web クリッピングのテスト」**セクションの**「テスト」**をクリックします。この図は属性ページの下部 3 分の 1 を示しています。Web クリッピングのテスト結果は、ブラウザのページに XML 文書として表示されます。この XML 文書のコンテンツを検査し、必要なコンテンツが含まれているかどうかを確認します。ワイヤレス・デバイスでは、この XML 文書はフォーマットされて表示されます。

図 13-12 Web クリッピング・アプリケーションのテスト

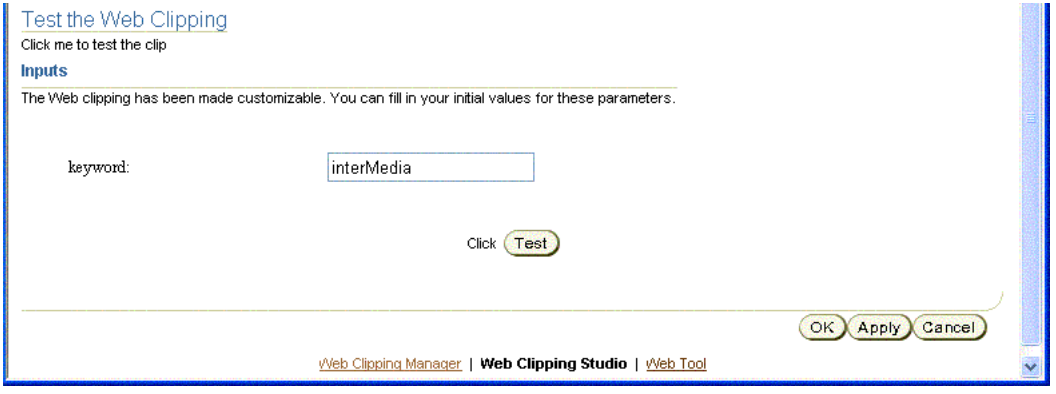


e. オプションの選択を終了した後は、**「適用」**をクリックして、変更内容を保存します。

注意：「**適用**」を選択すると、Web クリッピング・アプリケーションのすべての編集内容とオプションが保存されます。したがって、Wireless アプリケーション・ユーザーは、これらの変更内容に即時にアクセスできるようになります。また OracleAS Wireless 管理者は、Web クリッピング・アプリケーションの属性ページで必要に応じて編集を実行できるようになります。これに対して、「**OK**」を選択すると、Web クリッピング・アプリケーションの属性に対するすべての編集内容が保存され、同時に、Wireless アプリケーション・ユーザーは即時にこれらの変更内容にアクセスできます。ただし、Web クリッピング・アプリケーションの属性ページが終了し、OracleAS Wireless 管理者は、「**アプリケーション**」ページに戻ります。

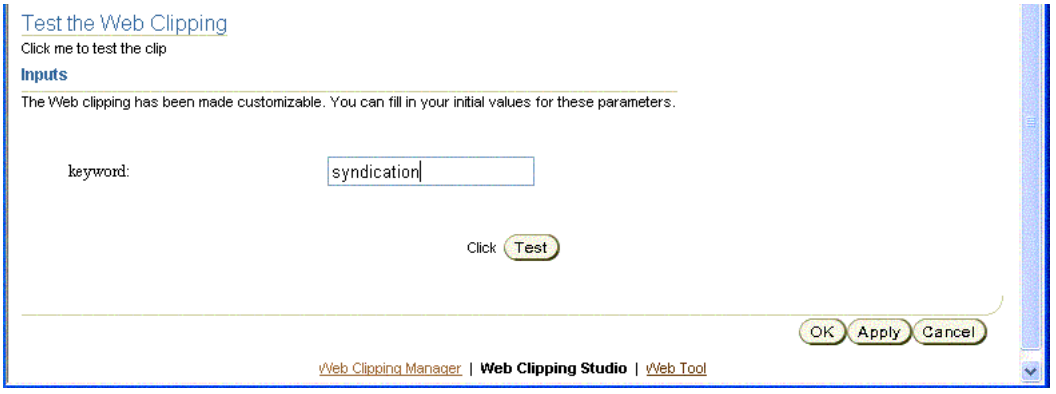
f. **Web クリッピング・アプリケーションの属性**ダイアログ・ボックスの「**Web クリッピングのテスト**」セクションで、「**適用**」をクリックして変更内容を保存すると、図 13-13「Web クリッピング・アプリケーションの入力値のテスト」のように、「**入力値**」見出しが表示され、パラメータ化およびカスタマイズされたすべてのパラメータがその初期入力値とともに一覧表示されます。

図 13-13 Web クリッピング・アプリケーションの入力値のテスト



g. `keyword` という名前のパラメータとその入力値 interMedia が表示されていることに注意してください。別の入力値をテストするには、図 13-14「入力値の変更」のように、初期値の interMedia を置換し、新しい初期値 syndication を入力します。「**テスト**」をクリックして新しい入力値をテストします。

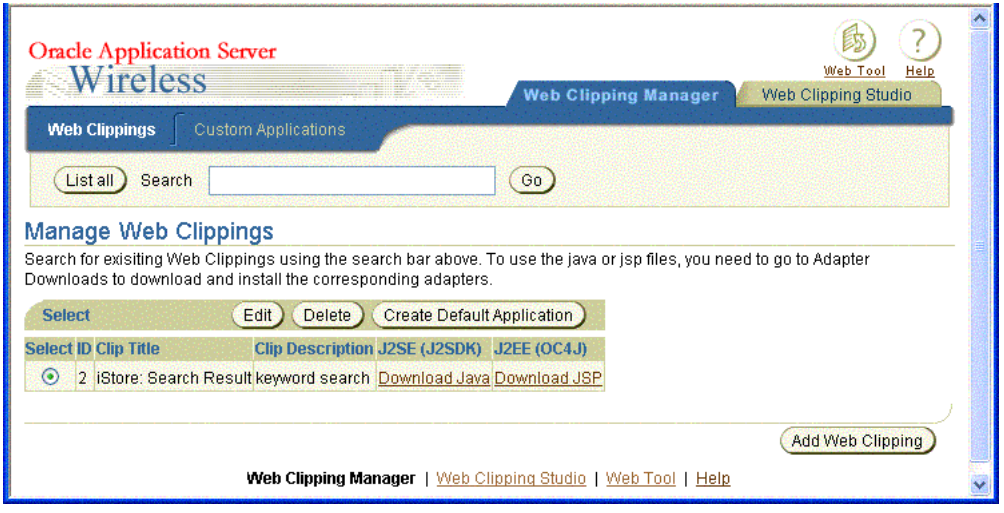
図 13-14 入力値の変更



h. この Web クリッピング・アプリケーションのテスト結果は、ブラウザのページに表示されます。この XML 文書のコンテンツを検査し、必要なコンテンツが含まれているかどうかを確認します。

i. 「**OK**」をクリックして、変更内容を保存し、図 13-15「Web クリッピングの管理」のように、「**Web クリッピングの管理**」ページに戻ります。このページでは、次の方法で Web クリッピングを管理できます。

図 13-15 Web クリッピングの管理



– 新しい Web クリッピングの作成

「Web クリッピングの追加」をクリックします。**「Web クリッピングの検索」**ページが表示されます。ここでは、開始ポイントの URL を入力して Web サイトの参照を開始し、クリップするコンテンツを検索できます。

– Web クリッピングの検索

「検索」フィールドに、検索する Web クリッピングの名前の一部に一致する検索文字列を入力し、次に**「実行」**をクリックして検索を開始します。入力した検索文字列に一致する名前を含む Web クリッピングのリストが表示されます。

– Web クリッピングの編集

編集する Web クリッピングを**「選択」**列にある該当のラジオ・ボタンをクリックして選択し、**「編集」**をクリックします。図 13-10「Web クリッピング・アプリケーションのオプションの選択とクリッピング属性の編集」のように、**Web クリッピングとクリッピング属性の検索**ページが表示されます。このページでは、Web クリッピングの属性を編集できます。

– Web クリッピングの削除

削除する Web クリッピングを**「選択」**列にある該当のラジオ・ボタンをクリックして選択し、**「削除」**をクリックします。Web クリッピングが削除されていることを示す確認メッセージが表示されます。

Wireless アプリケーションの作成

Web クリッピングから Wireless アプリケーションを作成する場合は、次のいずれかの方法を使用します。

- 開発用のデフォルトのアプリケーションを作成し、新しく作成された Web クリッピングのデフォルトのレンダリングを表示します。13-20 ページの「デフォルトのアプリケーションの作成」を参照してください。
- Java API を使用して、クリップしたデータにアクセスし、レンダリングをカスタマイズします。13-26 ページの「カスタム・アプリケーションの作成」を参照してください。

デフォルトのアプリケーションの作成

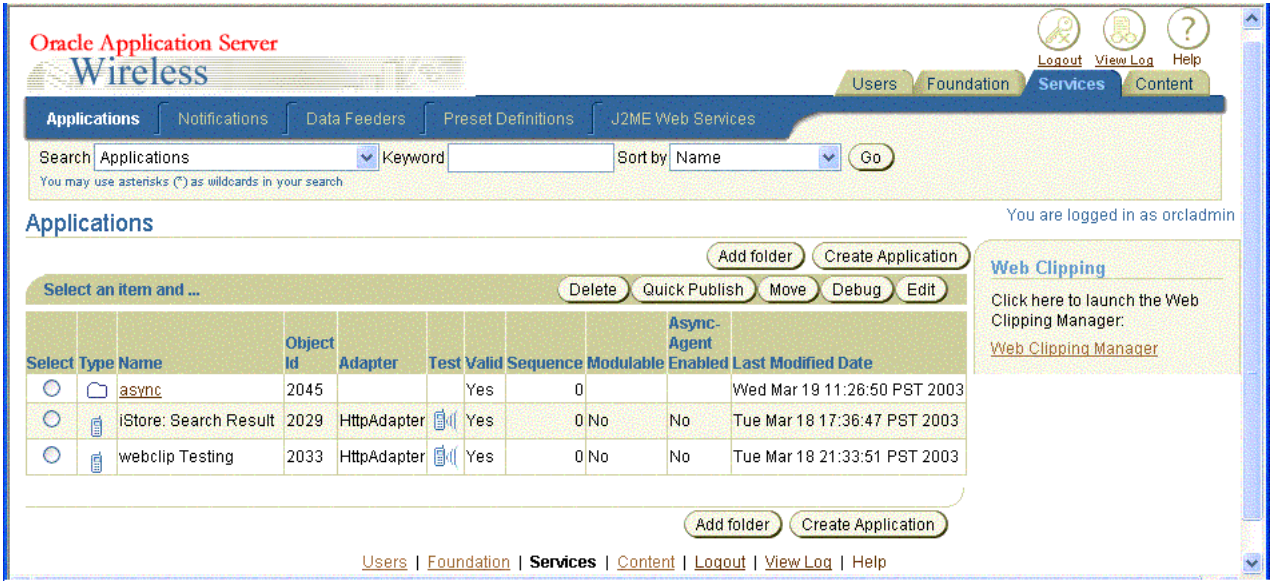
例 13-15「Web クリッピングの管理」に示した**「Web クリッピングの管理」**ページでは、デフォルトの Wireless アプリケーションを作成できます。

デフォルトの Wireless アプリケーションにする Web クリッピングを、その横にあるラジオ・ボタンをクリックして選択し、**「デフォルト・アプリケーションの作成」**をクリックします。デフォルトの Wireless アプリケーションが作成され、直前に実行したスクレイピングのデフォルトのレンダリングが OracleAS Wireless XML に配信されます。

デフォルトのアプリケーションを作成した後は、OracleAS Wireless の**「アプリケーション」**ページに戻り、そこで新しい Web クリッピング・アプリケーションを作成できます。

この例を続行すると、図 13-21「カスタム・アプリケーションの開発ページ」に示すような、iStore: Search Result というアプリケーションがアプリケーション・リストに追加されます。

図 13-16 アプリケーション・リストに追加された新しい Wireless アプリケーション iStore: Search Result



Web クリッピングからデフォルトのアプリケーションを作成すると、次のことも実行可能になります。

- 既存の Web クリッピング・アプリケーションの属性の編集

  編集するアプリケーションを、該当するラジオ・ボタンをクリックして選択し、**「Web クリッピング・スタジオでのクリップの編集」**セクションで**「編集」**をクリックして、Web クリッピング・アプリケーションの属性を編集します。

- Web クリッピング・アプリケーションの削除

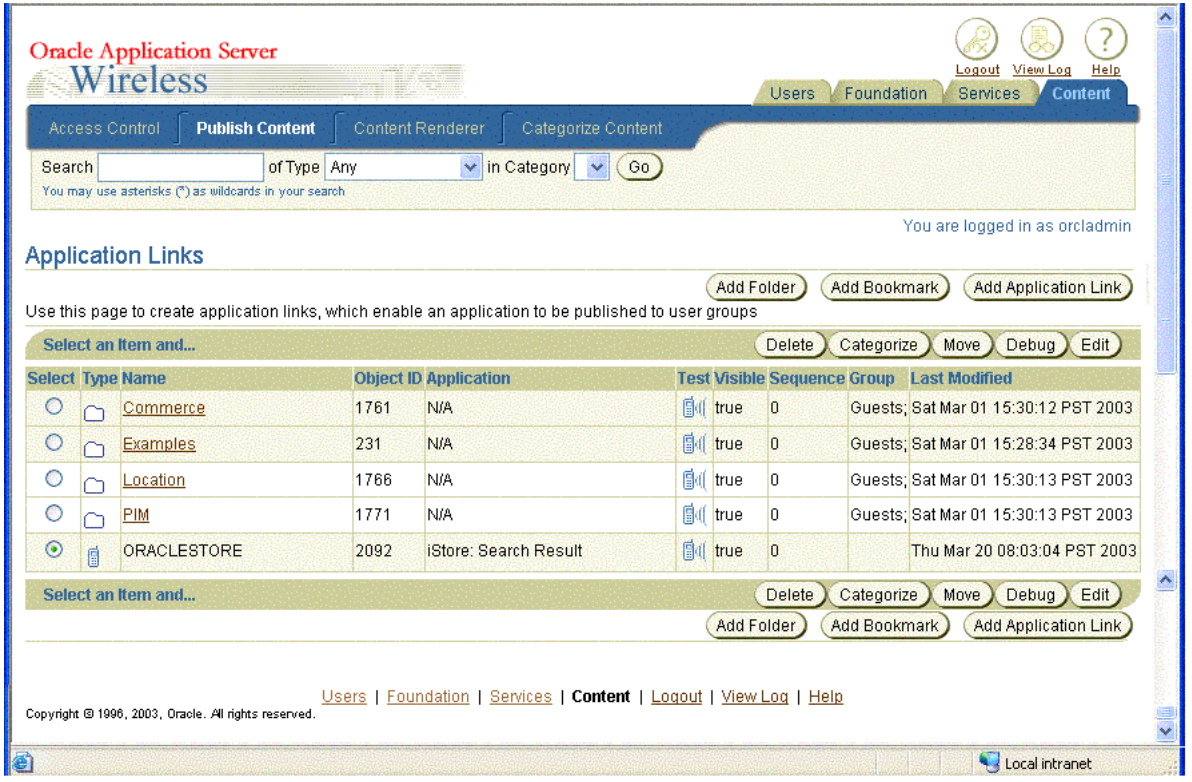
  削除するアプリケーションを**「選択」**列にある該当のラジオ・ボタンをクリックして選択し、**「削除」**をクリックします。選択した Web クリッピングが削除されていることを示す確認メッセージが表示されます。

- 新しいアプリケーションの公開

  新しい Wireless アプリケーションに対して新しいアプリケーション・リンクを作成するには、**「コンテンツ」**タブをクリックして**「アプリケーション・リンクの追加」**をクリックし、この新しいアプリケーションを**「アプリケーション・リンク」**ページに追加します。ウィザードが示す 5 つの手順に従ってください。**「アプリケーション」**手順では、このアプリケーションのベースにするアプリケーションを選択します。次に、アプリケーション iStore: Search Result を選択し、**「次」**をクリックします。**「一般」**手順では、新しいアプリケーションの名前としてアプリケーション名「ORACLESTORE」を入力し、**「次」**をクリックします。**「入力パラメータ」**手順では、**「次」**をクリックして、

新しいアプリケーションのデフォルトの設定を受け入れます。**「非同期アプリケーション」**手順では、**「次」**をクリックします。**「追加」**手順では、**「説明」**フィールドに「Books」と入力し、次に**「終了」**をクリックします。図 13-17「新しいアプリケーション ORACLESTORE のコンテンツ・マネージャへの公開」のように、新しいアプリケーションがアプリケーションの「アプリケーション・リンク」リストに表示されます。これによって、このアプリケーションはユーザー・グループに公開され、ワイヤレス・デバイスのモバイル・ユーザーがアクセスできるようになります。

図 13-17 新しいアプリケーション ORACLESTORE のコンテンツ・マネージャへの公開



- 既存の Web クリッピング・アプリケーションのテスト

  アプリケーションのテストは、**「サービス」**タブまたは**「コンテンツ」**タブ（公開済の場合）から実行できます。

  a. **「コンテンツ」**タブでは、**「テスト」**列の**「テスト」**アイコンをクリックして、テストするアプリケーション（この例では新しいアプリケーション・リンク ORACLESTORE）を選択します。「OracleAS Wireless PDA Simulator」ページが新

しいブラウザのウィンドウに表示され、図 13-2「Web クリッピング・アプリケーション」のように、Web クリッピング・アプリケーションのコンテンツが表示されます。

b. OracleAS Wireless PDA Simulator のテスト表示で、ページの下へスクロールして**「入力の変更」**をクリックし、図 13-18「初期値 interMedia が表示された Wireless PDA Simulator」のように、`keyword` というパラメータの初期入力値の値を変更します。初期入力値の interMedia を削除して、図 13-19「新しい値 Syndication が表示された Wireless PDA Simulator」のように、値「syndication」を入力します。次に、**「実行」**をクリックします。

図 13-18 初期値 interMedia が表示された Wireless PDA Simulator

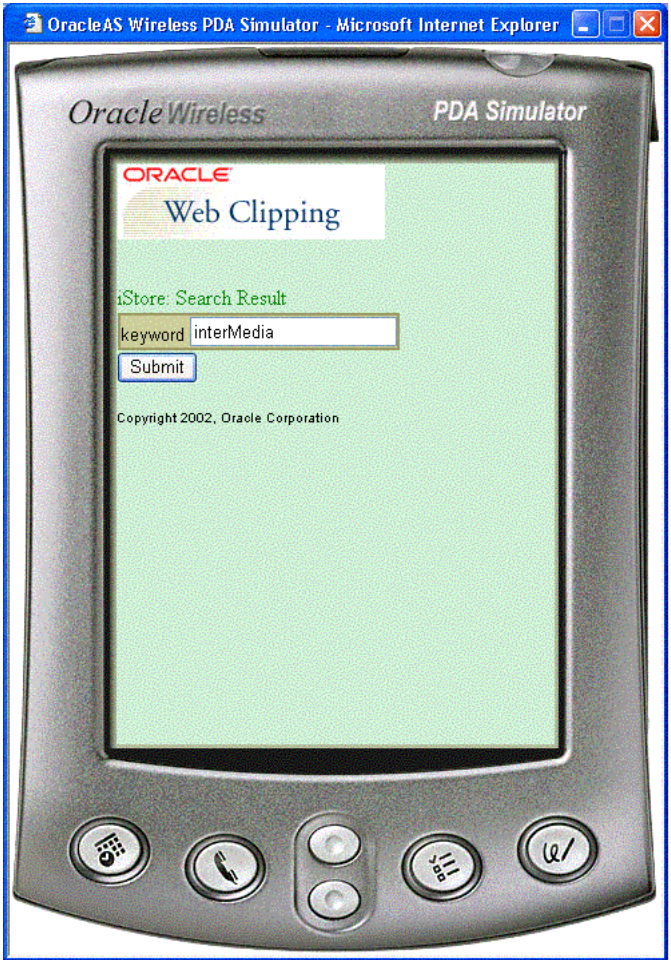
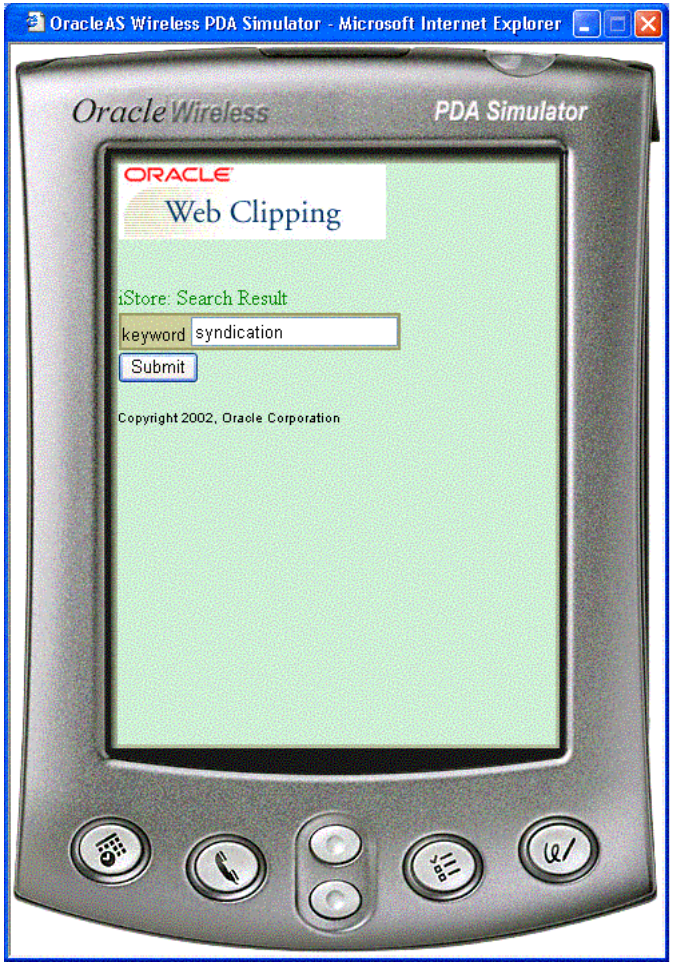
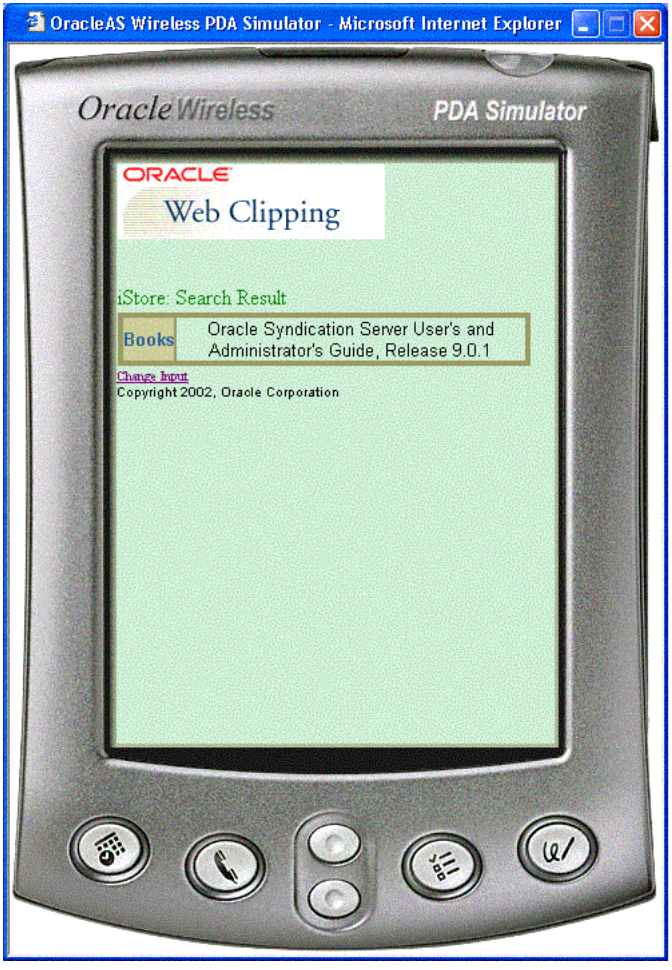


図 13-19 新しい値 Syndication が表示された Wireless PDA Simulator



c. OracleAS Wireless PDA Simulator のテスト・ページには、図 13-20「キーワード syndication を使用した Oracle Store 内の項目の検索結果が表示された Wireless PDA Simulator」のように、値 syndication に関連する Oracle Store 内の項目に対する検索結果が表示されます。

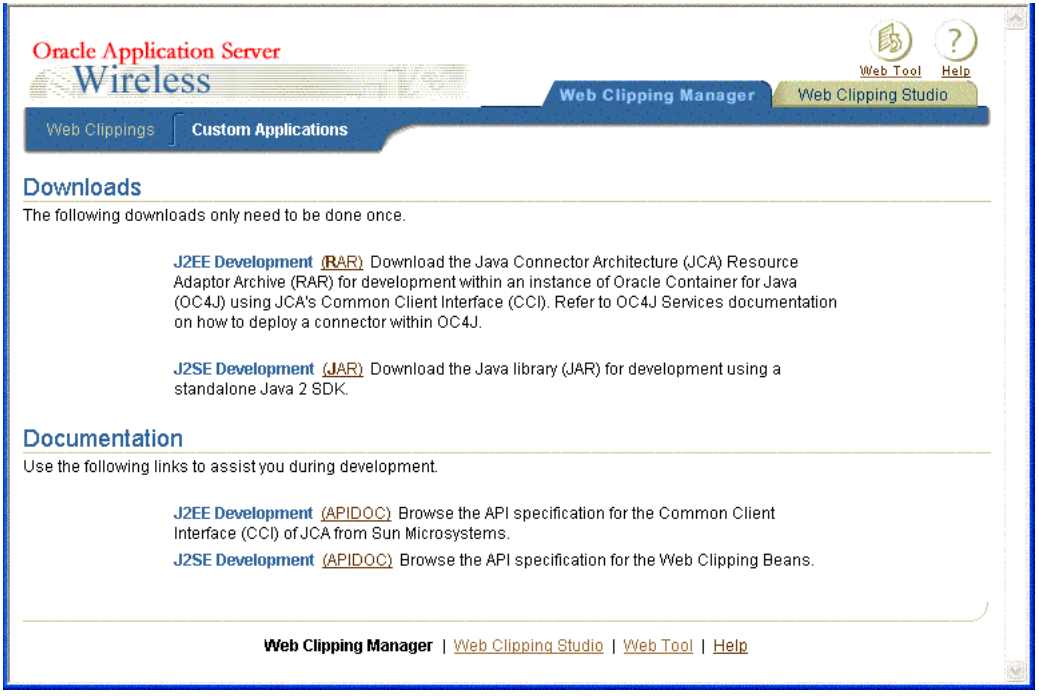
図 13-20 キーワード syndication を使用した Oracle Store 内の項目の検索結果が表示された Wireless PDA Simulator



カスタム・アプリケーションの作成

図 13-15「Web クリッピングの管理」に示した**「Web クリッピングの管理」**ページでは、カスタムの Wireless アプリケーションを作成できます。**「カスタム・アプリケーション」**タブをクリックします。図 13-21「カスタム・アプリケーションの開発ページ」に示す**「ダウンロード」**ページが表示されます。このページでは、次のいずれかを実行できます。

図 13-21 カスタム・アプリケーションの開発ページ



- Java Connector Architecture（JCA）の Common Client Interface（CCI）を使用した Oracle Container for J2EE（OC4J）のインスタンス内での開発用に、（J2EE 開発用の）の JCA Resource Adaptor Archive（RAR）をダウンロードできます。

  J2EE 開発の**「RAR」**リンクをクリックして、JCA RAR ファイル `webclipping.rar` をダウンロードします。

  最新の Oracle Application Server ドキュメントの OC4J サービスに関する項の指示に従って、使用している OC4J に RAR ファイルをデプロイします。Oracle Application Server ドキュメントのページを参照して、正しいバージョンの OC4J のドキュメントを検索します。

ライブラリを有効にするには、OC4J の再起動が必要な場合があります。

「Web クリッピング」タブに戻って、Java Server Page（JSP）形式の Web クリッピングを検索し、デプロイした RAR ファイルを使用して開発を開始します。

- スタンドアロン Java 2 SDK を使用した開発用に、（J2SE 開発用の）Java ライブラリ（JAR）をダウンロードします。

  J2SE 開発の**「JAR」**リンクをクリックして、Java ライブラリ・ファイル wcbean.jar をダウンロードします。

  スタンドアロン Java 2 SDK 環境では、コードのコンパイル時と実行時に、JAR ファイルをクラスパスに配置します。

  J2EE（OC4J）開発の場合、この JAR ファイルは、使用している Web アプリケーションがアクセスできる場所に配置します。たとえば、OC4J のルート・クラスパス、アプリケーション独自のライブラリ・パスまたは使用している Web アプリケーションが格納されている Web Archive（WAR）ファイルの WEB-INF/lib ディレクトリなどに配置します。Web クリッピングの Bean API を使用して開始する場合は、OC4J インスタンスを再起動する必要があります。

  スタブ Java ファイルのコンパイルと実行に必要な他のファイルは、`http_client.jar`、`Javax-ssl-1_2.jar`、`jssl-1_2.jar` と `xmlparserv2.jar` です。これらのファイルは、Oracle Application Server OC4J のインストールにあります。

  「Web クリッピング」タブに戻って、Java ソース・ファイル（`.Java`）形式の Web クリッピングを検索し、デプロイした JAR ファイルを使用して開発を開始します。

- **「Web クリッピング」**タブでは、次のいずれかを実行できます。

  – `.Java` ファイルの生成

    「Java のダウンロード」（図 13-15「Web クリッピングの管理」を参照）をクリックして、Web クリッピングから `.Java` ファイルを生成します。この `.Java` ファイルは、他の Java クラスを使用してアプリケーションにコンパイルし、スタンドアロン Java 2 SDK アプリケーションとして J2SE にデプロイできます。

  – `.jsp` ファイルの生成

    「JSP のダウンロード」（図 13-15「Web クリッピングの管理」を参照）をクリックして、Web クリッピングから `.jsp` ファイルを生成します。この `.jsp` ファイルは、J2EE OC4J にデプロイできるため、Web クリッピングを JSP として実行できます。

- ダウンロードした `.Java` ファイルまたは `.jsp` ファイルを使用して、独自のカスタム HTTP アプリケーションを作成できます。

  – `.Java` ファイルの使用

    Web クリッピングの Bean API の使用方法のサンプル・コードが含まれている `.Java` ファイルを起動し、前述の `wcbean.jar` ファイルとその依存性ライブラリを HTTP Server のライブラリ・パスに挿入すると、Wireless 開発者は、任意の標準

J2EE Container（OC4J など）を使用して独自の HTTP アプリケーションを作成できます。

– .jsp ファイルの使用

同様に、Java Connector Architecture API の使用方法を示すサンプル・コードが含まれている .jsp ファイルを起動することで、Wireless 開発者は、OC4J を使用して独自の HTTP アプリケーションを作成できます（RAR ファイルは OC4J に準拠しているためこれはハード要件です）。

– モバイル UI の作成

OracleAS Wireless 準拠にするために、HTTP アプリケーションでは、その出力を OracleAS Wireless XML、XHTML/MP または XHTML/XForms のいずれかでレンダリングする必要があります。Wireless 開発者は、モバイル・リンクやイメージの追加も含めてこの HTTP アプリケーションのモバイル UI を完全に制御できるようになります。デフォルトの JSP コードは、データを HTML の順序付けられていないリストとしてレンダリングするため、Wireless 開発者は、そのレンダリングを OracleAS Wireless 準拠のレンダリングに変更する必要があることに注意してください。

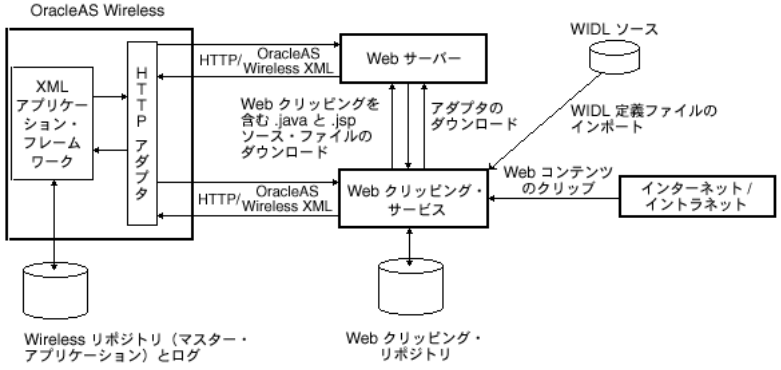
– Wireless アプリケーションの作成

HTTP アプリケーションの作成後、Wireless 開発者は、そのアプリケーションを使用して、HTTP Adaptor を使用した Wireless アプリケーションを作成できます。

既存のトランスコーディング・テクノロジからの移行

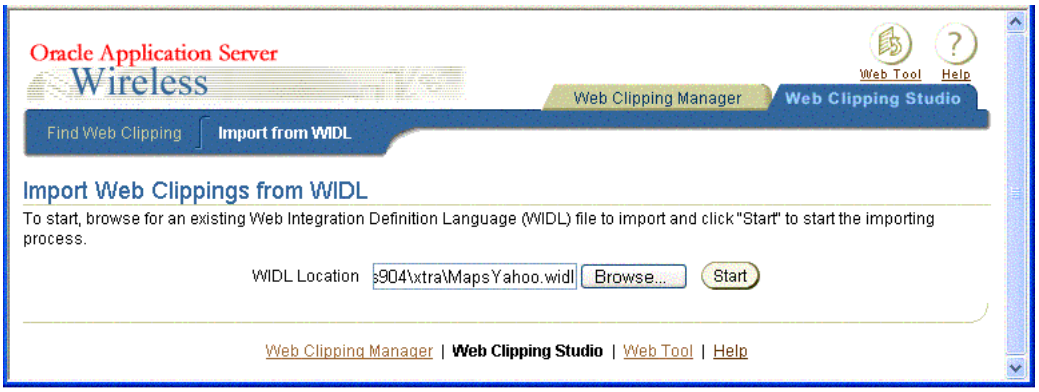
従来の OracleAS Wireless では、トランスコーディングまたは Web コンテンツの再利用は、Web Methods の Web 統合テクノロジを使用して実行していました。外部 Web コンテンツのフェッチ方法とフェッチ場所について記録した指示は、Web Integration Definition Language（WIDL）ファイルに記録されていました。WIDL は、World Wide Web のドキュメント・ベースのリソースに対してサービス・ベースのアーキテクチャを実装するメタ言語です。このリリースでトランスコーディングの役割を果たすのは Web クリッピングです。開発者やユーザーは、Web クリッピング語句に同じセットの指示を再作成する必要はありません。WIDL ファイルを Web クリッピングにインポートできる移行パスがあるため、前に取得した指示を再利用できます。これは、補強された Web クリッピング・サーバーのアーキテクチャ（図 13-22「WIDL 定義ファイルのインポートが示された Web クリッピング・サーバーのアーキテクチャ」参照）に、Web クリッピングを作成する代替手段として追加された WIDL ソースとともに示されています。

図 13-22　WIDL 定義ファイルのインポートが示された Web クリッピング・サーバーのアーキテクチャ



1. WIDL 定義ファイルからインポートして、そこから Web クリッピングを作成するには、**「Web クリッピング・スタジオ」**タブをクリックし、**「Web クリッピングの検索」**ページで**「WIDL からインポート」**サブタグを選択します。

図 13-23 WIDL 定義から Web クリッピングへのインポート



2. 図 13-23「WIDL 定義から Web クリッピングへのインポート」に示されている**「WIDL からインポート」**ページでは、既存の WIDL ファイルをインポートして Web クリッピングを作成し、それをリポジトリに保存できます。

   WIDL 定義ファイルをインポートする手順は、次のとおりです。

   a. **「WIDL ロケーション」**フィールドに開始ポイントの URL を入力し、**「開始」**をクリックします。または、システムの WIDL 定義ファイルがあるディレクトリを検索して、**「WIDL ロケーション」**フィールドに移入する WIDL ファイルをクリックします。次に、**「開始」**をクリックして WIDL 定義ファイルのインポートを開始します。図 13-24「インポートする WIDL サービスの選択、パラメータの選択および選択した各パラメータに対するデフォルト値の指定（ページの上部）」および図 13-25「インポートする WIDL サービスの選択、パラメータの選択および選択した各パラメータに対するデフォルト値の指定（ページの下部）」のように、新しいセクション**「WIDL サービスを選択してインポートします。」**を含む**「WIDL からインポート」**ページが再度表示されます。

図 13-24 インポートする WIDL サービスの選択、パラメータの選択および選択した各パラメータに対するデフォルト値の指定（ページの上部）

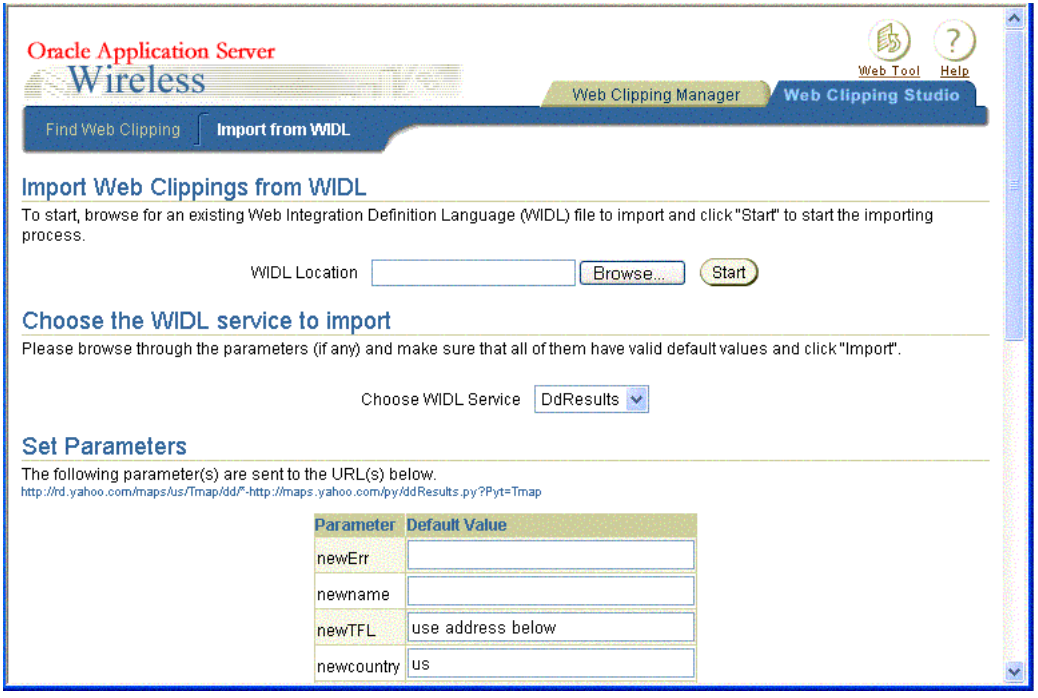


図 13-25 インポートする WIDL サービスの選択、パラメータの選択および選択した各パラメータに対するデフォルト値の指定（ページの下部）

b. インポートするサービスが WIDL ファイルに複数含まれている場合は、**「WIDL サービスを選択してインポートします。」**セクションの下にある **WIDL サービスの選択**ドロップダウン・ボックスからインポートするサービスを選択します。

c. WIDL サービスでフォーム・ベースの送信を実行した場合は、**「パラメータ」**列のドロップダウン・ボックスから必要なパラメータを選択し、さらに選択した各パラメータに対してデフォルト値を入力します。

d. **「続行」**をクリックします。WIDL サービスがインポートされ、Web クリッピングが作成されます。**Web クリッピングとクリッピング属性の検索**ページが表示されます。このページでは、新しい Web クリッピングの属性を編集できます。Web クリッピングにフォーム・ベースの送信が含まれている場合は、フォームの入力情報もカスタマイズできます。**「説明」**フィールドなどのクリッピング属性を編集した後は、**「OK」**をクリックして変更内容を保存し、**「Web クリッピングの管理」**ページに戻ります。

「Web クリッピングの管理」ページで、デフォルトのアプリケーションを作成した後（13-20 ページの「デフォルトのアプリケーションの作成」を参照）は、アプリケーションをユーザー・グループに公開できるようにアプリケーション・リンクを作成して、ワイヤレス・デバイスのモバイル・ユーザーからアクセスできるようにします。またはカスタム・アプリケーションを作成し、Web クリッピングから `.Java` ファイルまたは `.jsp` ファイルのいずれかを生成します（13-26 ページの「カスタム・アプリケーションの作成」を参照）。

Web クリッピング・サービスのカスタマイズ

OracleAS Wireless 管理者が入力パラメータをパラメータ化およびカスタマイズ可能に設定している場合、Wireless ユーザーは、テスト・ページの**「入力の変更」**をクリックしてフォームの任意の入力パラメータ値をカスタマイズできます（詳細は、ヘルプの Web クリッピング・アプリケーション属性の編集とパラメータのカスタマイズに関するヘルプ・トピックを参照）。

OracleAS Wireless 管理者の管理タスク

OracleAS Wireless 管理者による実行が必要な管理タスクは、次のとおりです。

- HTTP または HTTPS プロキシ設定の構成。プロキシ・サーバーの構成方法の詳細は、『Oracle Application Server Wireless 管理者ガイド』のサーバー構成に関する項を参照してください。
- セキュリティの構成。

  Oracle Wallet Manager によって生成された信頼できるサーバー証明書ファイル `ca-bundle.txt` は、Web クリッピング・サーバーの機能に付属しています。このファイルは、UNIX の場合は *<ORACLE_HOME>*`/portal/conf` に、Windows の場合は *<ORACLE_HOME>*`¥portal¥conf` にあります。このファイルには、HTTPS を使用して一部のセキュアなサーバーにナビゲートするときに使用する信頼できるサーバー証明書の初期リストが含まれています。ただし、このリストは、Web 上に存在している可能性があるサーバー証明書すべてを含んだ完全なリストではありません。したがって、追加の信頼できるサーバー証明書を、アクセス対象の新しい信頼できるサイト用に認識するには、このファイルを構成または拡張する必要があります。この信頼できる証明書ファイルの構成方法または拡張方法の詳細は、13-44 ページの「WML Translator のデプロイと構成」を参照してください。

  注意： `ca-bundle.txt` ファイルは、Oracle Application Server Portal が構成されていない場合でも、そのまま存在して機能します。

- 記録するイベントのレンダリングと有用なレポートの生成。

  Web クリッピングでは、記録するイベントのレンダリングが可能です。したがって、管理者はイベント・ログを問い合せ、請求に使用するレポートなどの有用なレポートを生成できます。レンダリング・イベントのロギングを有効化する方法および一連の PL/SQL プロシージャを利用して記録されたイベントを操作して有用なレポートを生成するために方法の詳細は、13-46 ページの「WML Translator の使用」を参照してください。

セキュリティの構成

OracleAS Wireless 管理者がセキュアなサイトにナビゲートすると、セキュアな情報を提供する Web サイトでは、通常、サイト自体を識別する証明書を管理者に返します。OracleAS Wireless 管理者が証明書を受け入れると、その証明書はブラウザの信頼できる証明書リストに挿入され、ブラウザとサーバー間にセキュアなチャネルがオープンできます。Web ブラウザと同様に、Web クリッピング・サーバーは、外部 Web サイトに対して HTTP クライアントとして動作します。Web クリッピング・サーバーは、信頼できるサイトの追跡を続けるために、これらのサイトの証明書を格納する `ca-bundle.crt` という名前のファイルを利用します。

付属の `ca-bundle.txt` ファイルは、Oracle Wallet Manager からエクスポートされたバージョンの信頼できるサーバー証明書ファイルです。Oracle Wallet Manager 内のデフォルトの信頼できるサーバー証明書には、Web 上に存在している可能性のあるサーバー証明書すべてが含まれているわけではありません。このため、OracleAS Wireless 管理者が HTTPS を使用してセキュアなサーバーにナビゲートした場合に、Web クリッピング・スタジオで「SSL ハンドシェイクに失敗しました」という例外を受け取る場合があります。この問題を解決するためには、`ca-bundle.crt` ファイルを、アクセスする新しい信頼できるサイトで補強する必要があります。OracleAS Wireless 管理者は、次の操作を実行して、付属の `ca-bundle.crt` ファイルを拡張する必要があります。

1. ブラウザ（可能な場合は Internet Explorer）を使用して、アクセスする各外部 HTTPS Web サイトから BASE64 フォーマットのルート・サーバー証明書をダウンロードします。この証明書は、信頼できる証明書ファイルから欠落している証明書です。
2. Oracle Wallet Manager を使用して各証明書をインポートします。
3. 信頼できるサーバー証明書をファイルにエクスポートし、`ca-bundle.crt` ファイルをそのファイルで置換します。

Oracle Wallet Manager の詳細は、Oracle Technology Network（OTN-J）（`http://otn.oracle.co.jp/`）の Oracle9*i* リリース 2（9.0.2）ドキュメント・セクションにある『Oracle Advanced Security 管理者ガイド』の第 17 章「Oracle Wallet Manager の使用」を参照してください。

記録するイベントのレンダリングと有用なレポートの生成

Web クリッピングでは、記録するイベントのレンダリングが可能です。したがって、管理者はイベント・ログを問い合せ、請求の目的で使用するレポートなどの有用なレポートを生成できます。イベント・ロギングを有効化するには、管理者は、<ORACLE_HOME>/j2ee/OC4J_Wireless/applications/webclipping/webclipping-web/WEB-INF/web.xml にある web.xml ファイル内の context-param を手動で変更する必要があります。参照する context-param には、oracle.webclipping.LogBusiness と同じ param-name があり、false のデフォルトの param-value があります。レンダリング・イベントのロギングを有効化するには、管理者は、その値を true に変更する必要があります。このパラメータを設定した後は、OC4J_Wireless インスタンスを再起動してこの変更をリフレッシュします。DCM を使用してこれを実行する方法は、OC4J のガイドを参照してください。

ロギングが有効化された後、管理者はインフラストラクチャ・データベースにある一連の PL/SQL プロシージャを利用して、記録されたイベントを操作できます。管理者は、最初に SYSDBA でインフラストラクチャ・データベースに接続し、次の行を実行して自分のユーザーを WCRSYS に変更する必要があります。

```
ALTER SESSION SET CURRENT_SCHEMA=WCRSYS;
```

管理者は、ユーザー WCRSYS として、次の PL/SQL プロシージャとファンクションを利用して、記録されたイベントを操作できます。これらのプロシージャとファンクションの大部分で前置きとして使用するレコード・タイプは、WWWCP_API_REGISTRY.REC_RENDER_EVENT です。

```
/**
* This describes a record type used to return a single clipping rendering
* event.
*
* This structure is used by the lookup APIs to encapsulate the
* information that is retrieved from the wwwcp_render_log$ table.
* It is used to describe the cursor that will be returned as an OUT
* parameter of lookup_render_events.
*
* @field clip_id           The clip id that allows the fetching of the
*                          other facets of the clipping definition to
*                          populate the events table.
* @field clip_description  Textual description of the clip rendered.
* @field clip_title        Title of the clip that was rendered.
* @field clip_timeout      Timeout in milliseconds that allows the clipping
*                          that was rendered to be timed out. This could be
*                          an indication of the quality of service.
* @field effective_url     The url where the clip resides. This is usually
*                          the last url declared in the clipping definition
*                          clipping definition, where the clip would reside.
* @field render_status     A number that indicates either success or
*                          failure of the rendering call.
```

```
* @field render_type       Tells what type of rendering is in question,
*                          whether it be for Portal Show Mode, for
*                          Wireless default SimpleResult show mode, or
*                          for the Wireless connector show mode.
* @field render_start      Starting time of the rendering.
* @field render_end        Ending time of the rendering.
* @field fuzzy_used        Indicates whether fuzzy match was kicked in.
* @field db_lookup         The time it takes in milliseconds to look up
*                          the clipping definition from the database.
* @field http_latency      The time it takes to reach the first byte to
*                          read from the http remote site.
* @field render_user       The user (together with a possible realm) for
*                          which the rendering was done.
* @field error_cause       The cause of the error if the status indicated
*                          a failure of the rendering.
* @field event_description This field provides more information about
*                          the logged event.
* @field clip_input        The input provided to render this clipping. It
*                          is usually provided in the form of an HTTP URL
*                          query string like abc=def.
* @field clip_output       The possible output (only if it's small) of
*                          rendering, if it gives any more hints on the
*                          billing process.
*/
type rec_render_event is record (
    clip_id           integer,
    clip_description  varchar2(1024),
    clip_title        varchar2(512),
    clip_timeout      integer,
    effective_url     varchar2(2048),
    render_status     integer,
    render_type       integer,
    render_start      date,
    render_end        date,
    fuzzy_used        integer,
    db_lookup         integer,
    http_latency      integer,
    render_user       varchar2(512),
    error_cause       varchar2(256),
    event_description varchar2(256),
    clip_input        varchar2(128),
    clip_output       varchar2(256));
```

このレコード・タイプで使用するカーソル・タイプは、WWWCP_API_REGISTRY.RENDER_EVENT_CURSOR です。WWWCP_API_REGISTRY 内の次のプロシージャは、レンダリング・イベント・ログを参照できるように定義されています。

```
/**
 * This API looks up rows of render events from the wwwcp_render_log$
 * table, filtered by a specific clip id.
 *
 * The full rendering event information structure is returned as iterable
 * by the cursor returned.
 *
 * @param p_clip_id          The clip id as a filtering mechanism of the
 *                           events that were logged for this particular
 *                           clip.
 * @param p_cv_render_events The list of render events that are associated
 *                           with this clip.
 */
 procedure lookup_render_events(
     p_clip_id          in  integer,
     p_cv_render_events out render_event_cursor
 );

/**
 * This API looks up rows of render events from the wwwcp_render_log$
 * table, filtered by a specific effective url.
 *
 * This is useful for business queries that center on not the clip id but by
 * the actual location of where the clips are found. For example, if Yahoo's
 * partner, namely someone using Web Clippings, can do a search of all the
 * events that went to http://my.yahoo.com, it can find out how to charge
 * each user for what they have rendered.
 *
 * The full rendering event information structure is returned as iterable
 * by the cursor returned.
 *
 * @param p_effective_url    The effective url, starting with which points
 *                           to the web page containing the clip.
 * @param p_cv_render_events The list of render events that are associated
 *                           with this clip.
 */
 procedure lookup_render_events(
     p_effective_url    in  varchar2,
     p_cv_render_events out render_event_cursor
 );

/**
 * This API looks up rows of render events from the wwwcp_render_log$
 * table, filtered by the domain.
 *
 * This is useful for business queries that center on not the clip id but by
 * the domain of where the clips are found. For example, if Yahoo can do a
```

```
 * search of all the events that went to *.yahoo.com, it can find out how to
 * charge each user for what they have rendered.
 *
 * The full rendering event information structure is returned as iterable
 * by the cursor returned.
 *
 * @param p_effective_domain The domain of the effective url, starting with
 *                           which points to the web page containing the
 *                           clip. This domain is provided in the format
 *                           "oracle.com" or "yahoo.com".
 * @param p_cv_render_events The list of render events that are associated
 *                           with this clip.
 */
 procedure lookup_render_events(
     p_effective_domain in  varchar2,
     p_cv_render_events out render_event_cursor
 );

 /**
 * This API looks up rows of render events from the wwwcp_render_log$
 * table, filtered by the user that requested the render.
 *
 * This is useful for business queries that center on not the clip id but by
 * the user to visualize how much a user has rendered, regardless of where
 * the clipping is from, so it can charge for a flat fee based on for
 * example, the number of render events for a particular user.
 *
 * The full rendering event information structure is returned as iterable
 * by the cursor returned.
 *
 * @param p_render_user      The user on behalf of whom the render request
 *                           is made.
 * @param p_cv_render_events The list of render events that are associated
 *                           with this clip.
 */
 procedure lookup_render_events(
     p_render_user      in  varchar2,
     p_cv_render_events out render_event_cursor
);
```

記録されたイベントが割当済のデータベース表領域を超えていないことを確認するのも管理者の役割です。したがって、次のプロシージャは、レンダリング・イベント・ログからエントリを削除できるように定義されています。

```
/**
 * This API removes all render events from the wwwcp_render_log$ table.
 *
```

```
 * This is useful as a cleansing measure that an organization can do between
 * the different phases of rolling out their Web Clipping usage initiative.
 * It can be used to reset the stage before going into development testing
 * phase, or when the rollout is going live and the company wishes to begin
 * charging its subscribers for their usages, or another possible cleansing
 * may occur at for example, every month end or year end when the data
 * recorded is no longer useful.
 */
 procedure remove_all_render_events;

/**
 * This API removes all render events from the wwwcp_render_log$ table for
 * a given rendering user.
 *
 * This is useful as a cleansing measure that an organization can do for
 * example, labeling the development user to be some constant user id and
 * then removing all rendering events when it goes live.
 *
 * @param p_render_user The user on behalf of whom the render request is
 *                      made.
 */
 procedure remove_render_events(p_render_user in varchar2);

/**
 * This API removes all render events from the wwwcp_render_log$ table for
 * a given starting or stopping time before or after the query time.
 *
 * This is useful as a cleansing measure that an organization can do for
 * example, all rendering events that started or stopped before a certain
 * date are deemed development efforts and therefore should be disregarded.
 *
 * @param p_query_time   The date/time with which to query.
 * @param p_query_before Whether the query is for before or after. A value
 *                       of 1 signifies that the query is for comparing
 *                       times before the p_query_time while a 0 value.
 *                       implies comparing times after the p_query_time.
 * @param p_query_start  Whether or not the query is for start time.
 *                       A value of 1 means true while 0 means false.
 */
 procedure remove_render_events(
     p_query_time   in date,
     p_query_before in number,
     p_query_start  in number
 );
```

WML Translator

OracleAS Wireless WML Translator は、WML で開発されたアプリケーションをすべての Web 対応ワイヤレス・デバイス用にリフォーマットします。実行時に、WML で開発されたコンテンツは OracleAS Wireless XML に変換された後、適切なデバイス固有のマークアップ言語に変換されます。

通常、WML 1.x（WML 1.3 まで）のソース・ドキュメントはサポートされます。各ソース・ドキュメントは次の順序で処理されます。

- ソース WML の妥当性がチェックされます。有効な XML であるか。有効な WML であるか。
- "card" 要素の下の隣接するすべての "p" 要素がグループ化されます。グループ化された "p" 要素が整理統合されます。
- ソース WML が OracleAS Wireless XML に変換されます。
- ナビゲーション要素（ある場合は、13-44 ページの「WML Translator のデプロイと構成」を参照）が、結果の Wireless XML に追加されます。
- 結果の Wireless XML の URL が絶対 URL に変換され（未変換の場合）、さらに OracleAS Wireless Portal を指し示すようにリライトされます。

表 13-1 に、WML のソース要素と OracleAS Wireless XML の変換済要素の比較を簡略に示します。

表 13-1 ソース要素と変換済要素の関連

WML のソース要素	Wireless XML の変換済要素	説明
wml	SimpleResult	
head	n/a	子要素 "meta" は SimpleMeta に変換されます。
template	n/a	属性の "@onenterforward"、"@onenterbackward"、"@ontimer" および子要素の "do"、"onevent" は、WML 仕様（次の "card" 要素を参照）に基づいて認められます。
card	SimpleContainer	属性の "@onenterforward"、"@onenterbackward"、"@ontimer" および子要素の "do"、"onevent" は、"template" 要素の対応する属性 / 子要素と組み合せて処理されます。詳細は、次の "card/@onenterforward"、"card/@onenterbackward" および "card/@ontimer" を参照してください。

表 13-1 ソース要素と変換済要素の関連（続き）

WML のソース要素	Wireless XML の変換済要素	説明
card/@onenterforward（適用可能な場合は template/@onterforward）	SimpleBind	子要素 SimpleMatch（子要素 SimpleEvent/@type="onenterforward" 付き）および子要素 SimpleTask（card/@onenterforward に割当済の子要素 SimpleGo/@target 付き）
card/@onterbackward（適用可能な場合は template/@onenterbackward）	SimpleBind	子要素 SimpleMatch（子要素 SimpleEvent/@type="onenterbackward" 付き）と子要素 SimpleTask（card/@onenterforward に割当済の子要素 SimpleGo/@target 付き）
card/@ontimer（適用可能な場合は template/@ontimer）	SimpleTimer	
card/onevent（適用可能な場合は template/onevent）	SimpleBind または SimpleTimer	event/@type="ontimer" の場合は SimpleTimer が、それ以外の場合は、対応する子要素付きの SimpleBind（SimpleEvent & SimpleTask）が生成されます。
card/do n（適用可能な場合は template/do）	SimpleBind	"do" に子要素 "go" がある場合は、子要素（SimpleMatch/SimpleKey、SimpleTask/SimpleSubmit および SimpleDisplay）付きの SimpleBind が生成されます。"do" に子要素 "prev" がある場合は、子要素（SimpleMatch/SimpleKey、SimpleTask/Prev および SimpleDisplay）付きの SimpleBind が生成されます。また、"do" に子要素 "refresh" がある場合は、子要素（SimpleMatch/SimpleKey、SimpleTask/SimpleRefresh および SimpleDisplay）付きの SimpleBind が生成されます。
p	SimpleForm、SimpleMenu または SimpleText	"p" に "input" 要素または "select" 要素（option/@value 付き）がある場合は SimpleForm が、"p" に "select/option" 要素（"@onpick" 属性のみ付き）がある場合は SimpleMenu が生成されます。それ以外は、SimpleText が生成されます。

表 13-1 ソース要素と変換済要素の関連（続き）

WML のソース要素	Wireless XML の変換済要素	説明
pre		処理されるのは、"a"、"anchor"、"do"、"u"、"br"、"b"、"i"、"em"、"strong" などの子要素のみです。
input	SimpleFormItem	通常 "input" 要素の外にあるタイトルは、SimpleFormItem にコピーされます。
select	SimpleMenuItem、SimpleHref または SimpleFormOption	select/option/@onpick（子 "event" なし）は SimpleMenu/SimpleMenuItem に、select/option/@onpick（子 "event" あり）は SimpleMenu/SimpleBind/SimpleMatch/SimpleMItem に、select/@value は SimpleForm/SimpleFormSelect/SimpleFormOption にそれぞれ変換されます。
a	SimpleHref	
anchor	SimpleHref	"anchor" に "go" 要素がある場合は SimpleHref が生成され、"anchor" に "prev" または "refresh" 要素がある場合、現行リリースでは何も生成されません。
img	SimpleImage	イメージ調整は、904 リリースから使用できます。
br	SimpleBreak	
b、strong、em、big	SimpleStrong	
i、small	SimpleEm	
u	SimpleUnderline	
text node	SimpleTextItem	通常、テキスト・ノードは、次の 2 つの場合を除いて、そのままコピーされます。1.1. ノードが "input"、"select" の直前の兄弟の場合。2. ノードが "formatting" 要素（"em"、"b"、"u" など）の唯一の子でない場合。
table	SimpleTable	
tr	SimpleRow	
td	SimpleCol	

WML Translator はパススルー・モードをサポートしていません。エンド・ユーザーのデバイスが WML をサポートしている場合でも、WML Translator は、ソースの WML を OracleAS Wireless XML フォーマットに変換します。

通常、URL は Wireless and Voice Portal を指し示すようにリライトされます。ただし、イメージの URL はリライトされません。イメージ調整は、このリリースで使用できます。たとえば、次の WML 要素は、

```
<img src="/images/yahooicon.wbmp" alt="Yahoo!"/>
```

次のように変換されます。

```
<SimpleImage src="http://wap.yahoo.com/images/yahooicon.wbmp" vspace="0" hspace="0"
valign="bottom" alt="Yahoo!" addImageExtension="auto"/>
```

このイメージは、エンド・ユーザーのデバイス・モデルに基づいて調整およびレンダリングされます。

現行リリースの WML Translator には、特定の制約があります。

- OracleAS Wireless XML には、WML から Wireless XML への完全な変換を妨げるいくつかの制限があります。最も注意が必要な短所は、Wireless XML にはイベント処理モデルがないことです。
- WML Translator を装備した WML ユーザー・エージェントには制限があります。ナビゲーション履歴がメンテナンスされません。WML 変数のサポートはかなり制限されています。
- WML スクリプトはまだサポートされていません。
- アンカーの送信 : 子 "go" を持つ "anchor" 要素がある場合、このカードに表示できるのは、1 つの "anchor" のみです。

WML Translator のデプロイと構成

WML Translator は、OracleAS Wireless アプリケーションとしてデプロイされます。以前のバージョンと同様、エンド・ユーザーは「Commerce」→「Translator」をクリックしてアクセスできます。ただし、次の 3 つのサービス入力パラメータはサポートされなくなりました。

```
ORACLE_SERVICES_COMMERCE_TRANSLATOR_DEFAULT_CONNECTION
ORACLE_SERVICES_COMMERCE_TRANSLATOR_HELPER_WML
ORACLE_SERVICES_COMMERCE_TRANSLATOR_XSL_WML_FILENAME
```

次のサービス入力パラメータは引き続きサポートされています。

```
ORACLE_SERVICES_COMMERCE_TRANSLATOR_SHOW_GOHOME
```

次のサービス入力パラメータが、ナビゲーション・サポートを強化するために追加されました。

```
ORACLE_SERVICES_TRANSCODER_NAVIGATION
```

ORACLE_SERVICES_TRANSCODER_NAVIGATION に有効な値がある場合、ORACLE_SERVICES_COMMERCE_TRANSLATOR_SHOW_GOHOME は無視されます。

ORACLE_SERVICES_TRANSCODER_NAVIGATION は、サーバーのローカル・ファイル・システム上の URL またはファイル経由でアクセスできる XML ファイルを指し示している必要があります。XML には、ナビゲーション仕様が含まれています。次にサンプル・ナビゲーション XML を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<Navigation>
<NavigationItems>
<Item target="%value home.url%"
label="Home"
showAs="Link"
preferredLocation="Header" />
<Item target="%value service.parent.url%"
label_prefix="Back"
showAs="Link" />
<Item target="http://www.oraclemobile.com"
label="OracleMobile"
showAs="Button"
preferredLocation="Footer" />
</NavigationItems>
</Navigation>
```

各ナビゲーション項目には、表 13-2 で説明する属性があります。

表 13-2 ナビゲーション項目の属性

属性名	内容	必須	受け入れる値	デフォルト値
target	アクセス先	必須	完全修飾された URL またはモバイル・コンテンツのプレースホルダ（ポータル・ホームやサービス・ホームなど）。	n/a
label	エンド・ユーザーに表示されるラベル	オプション	文字列	n/a
label_prefix	ラベルの接頭辞	オプション	ポータル・ホームなどのモバイル・コンテキストのわかりやすい値のみ。	
label_suffix	ラベルの接尾辞	オプション	ポータル・ホームなどのモバイル・コンテキストのわかりやすい値のみ。	
showAs	ラベルの表示方法	オプション	メニュー項目、リンクまたはボタン	button
preferredLocation	ラベルの表示場所	オプション	ヘッダーまたはフッター。	header

次に、ナビゲーション XML のスキーマを示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:element name="Navigation">
<xs:complexType>
<xs:all>
<xs:element ref="NavigationItems" minOccurs="0"/>
</xs:all>
</xs:complexType>
</xs:element>
<xs:element name="NavigationItems">
<xs:complexType>
<xs:sequence>
<xs:element ref="Item" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Item">
<xs:complexType>
<xs:attribute name="target" type="xs:string" use="required"/>
<xs:attribute name="label" type="xs:string" use="optional"/>
<xs:attribute name="label_prefix" type="xs:string" use="optional"/>
<xs:attribute name="label_suffix" type="xs:string" use="optional"/>
<xs:attribute name="showAs" type="xs:string" use="optional"/>
<xs:attribute name="preferredLocation" type="xs:string" use="optional"/>
</xs:complexType>
</xs:element>
</xs:schema>
```

WML Translator の使用

WML Translator は、OracleAS Wireless アプリケーションとしてデプロイされます。アプリケーションの URL は omp://oracle/services/commerce/translator です。このアプリケーションは、WML ソース URL をリクエスト・パラメータ XLTORSITE に渡すことで起動できます。たとえば、www.oraclemobile.com を起動するには、次の URL を OracleAS Wireless XML で使用できます。

```
omp://oracle/services/commerce/translator?XLTORSITE=http%3A%2F%2Fwww.oraclemobile.com
```

14

ロケーション・サービスの使用

この章では、ロケーション・ベースのアプリケーションの開発者を対象として、概念と使用方法について説明します。項ごとに様々なトピックを記載しています。

- ロケーション・サービスの概要
- ロケーション・ベースのアプリケーションの開発
- モバイル・ポジショニングの有効化
- ロケーション・イベント・サーバー
- リージョン・モデル・ツールの使用
- 外部コンテンツ・プロバイダの統合
- モバイル・ポジショニング・プロバイダの統合

ロケーション・サービスの概要

ロケーション・ベースのアプリケーションの開発者には、次の特殊サービスが必要です。

- モバイル・ポジショニング : モバイル・ユーザーのロケーションを特定します。
- ジオコーディング : 地理上の座標を住所に関連付けます。
- マッピング : 地点、地点のセット、ルートまたは運転経路用のグラフィカル・マップを提供します。
- ルーティング : 運転方向を提供します。
- ビジネス・ディレクトリ（イエロー・ページ）: ビジネスをリージョン別のカテゴリ順または名前順に列挙します。
- トラフィック : 事故、工事および交通量に影響するその他の事象の情報を提供します。

複数の会社が、この種の特殊なコンテンツおよびアプリケーションを提供しています。たとえば、ビジネス・ディレクトリ用のカテゴリが用意されている Web サイトや、運転方向を提供しているサイトがあります。OracleAS Wireless フレームワークに基づいてモバイル・アプリケーションを作成する開発者にとっては、特殊なコンテンツとサービスを使用できることによる利点があります。アクセス先として必要なすべてのサービスへのカスタム・インタフェースを、アプリケーションごとに作成するのは非効率的です。

OracleAS Wireless のロケーション・アプリケーション・コンポーネントは、ジオコーディングの実行、運転方向の提供およびビジネス・ディレクトリの検索に使用する API（Application Programming Interface）のセットです。既存の重要なプロバイダを API にマップするサービス・プロキシが組み込まれており、将来は追加のプロバイダにも対応する予定です。

OracleAS Wireless アプリケーションの開発者は、アプリケーションに変更を加えなくても、汎用インタフェースを使用して様々なサービス・プロバイダにアクセスできます。また、インフラストラクチャを使用し、品質、可用性またはコストなどの基準に基づいてサービスの優先順位を設定できます。サービス・プロバイダにとっても、コンテンツと特殊機能をすべての OracleAS Wireless アプリケーション開発者がそのまま使用できることによる利点があります。

この項では、ロケーション・アプリケーション・コンポーネント API の概要、Javadoc で生成された詳細なドキュメントとオンライン例の検索方法、各コンポーネントを使用する上で前提となる概念と使用方法について説明します。項ごとに様々なトピックを記載しています。

- スタート・ガイド
- ロケーション関連情報でのシステム・マネージャのインタフェースの使用
- ロケーション・サービスのアーキテクチャ
- ロケーション・サービスのカテゴリ
- サービス・プロバイダ

- ジオコーディング・サービス
- ロケーション・マーク
- マッピング・サービス
- ルーティング・サービス
- ビジネス・ディレクトリ（イエロー・ページ）サービス
- トラフィック・サービス

スタート・ガイド

OracleAS Wireless のロケーション・アプリケーション・コンポーネントの使用を開始する手順は、次のとおりです。

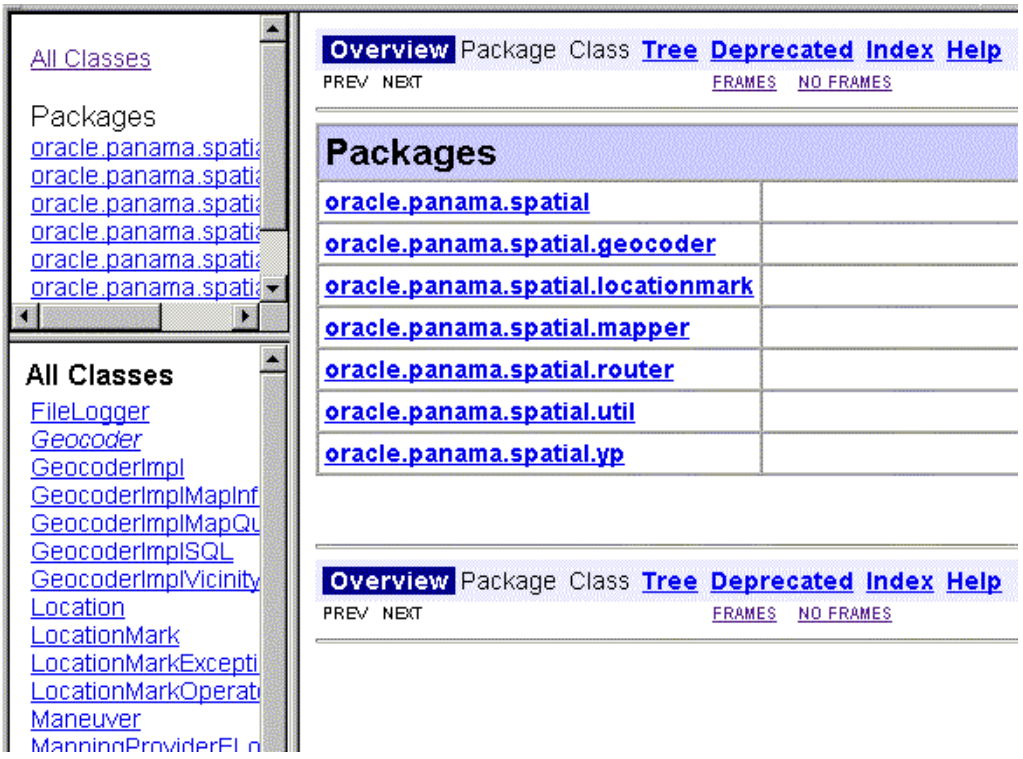
1. サンプル・プログラムを使用したりアプリケーションを作成する前に、この項の概念および使用方法の説明を読みます。
2. `sample` ディレクトリに移動します。このディレクトリにサンプル・ファイルが格納されています。このディレクトリにある `Readme.txt` ファイルを読み、提供されるファイルを調べて、ニーズを満たしているファイルを使用してください。
3. Javadoc ドキュメントを表示し、パッケージとクラスの詳細な参照情報を調べます。Javadoc ドキュメントを表示するには、Web ブラウザで次のファイルを開きます。

   iAS-Wireless-Home`/wireless/doc/index.html`

   iAS-Wireless-Home は、OracleAS Wireless のホーム・ディレクトリです。

   図 14-1 に、`index.html` の表示内容の一部を示します。ナビゲートし、パッケージとクラスの詳細情報を調べます。

図 14-1 Javadoc ドキュメント



ロケーション関連情報でのシステム・マネージャのインタフェースの使用

Enterprise Manager 内で OracleAS Wireless システム・マネージャ（システム・マネージャと呼ばれます）のインタフェースを使用すると、構成操作の実行、およびロケーション・アプリケーション・コンポーネントに関連する情報の検索ができます。

1. Wireless サーバーの**「システム」**タブのページで、**「サイト管理」**をクリックします。
2. クリックして**「コンポーネント構成」**を展開します。

図 14-2 に、「コンポーネント構成」セクションを展開した状態のシステム・マネージャのページを示します。

図 14-2 Wireless システム・マネージャの「コンポーネント構成」セクション

図 14-2 のように、「コンポーネント構成」セクションの「関連ロケーション」セクションには、次のリンクが含まれています。

- **「ロケーション管理」**は、モバイル・ポジショニング構成、モバイル・ポジショニング・プロバイダの情報と構成、およびモバイル ID を提供します。
- **「ロケーション・サービス」**は、ジオコーディング、ルーティング、マッピング、トラフィックおよびビジネス・ディレクトリ・サービスに関連する構成オプションを提供します。
- **「ロケーション・イベント・サーバー」**は、ロケーション・イベント・サーバー（14-124 ページの「ロケーション・イベント・サーバー」を参照）に関連するオプションを提供します。
- **「ロケーション・マーク・アドレス書式」**では、ロケーション・マーク・アドレスのフィールドを指定します。

ロケーション・サービスのアーキテクチャ

ロケーション・サービスは、図 14-3 に示すアーキテクチャに基づいています。

図 14-3 ロケーション・サービスのアーキテクチャ

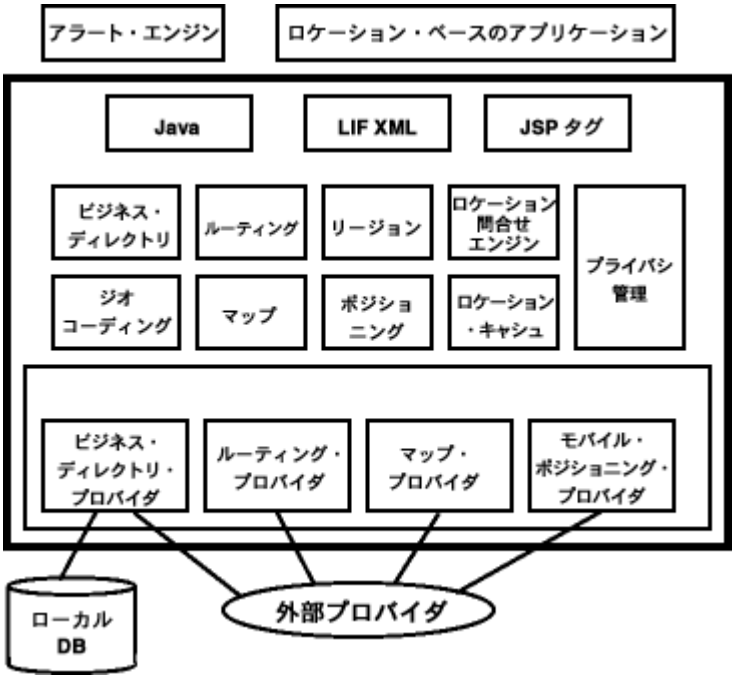


図 14-3 は、次の事項を示しています。

- アラート・エンジンおよびロケーション・ベースのアプリケーションは、（太線で囲まれた）アーキテクチャの外側にありますが、アーキテクチャと通信します。
- このアーキテクチャでは、Java、XML および JSP タグを使用するリクエストの処理が可能です。

- この処理は、特定のアクティビティや各種のサービスを処理するコンポーネントによって行われます。これらのコンポーネントには、ビジネス・ディレクトリ、ジオコーディング、ルーティング、マップのサポート、リージョンのサポート、ポジショニング、ロケーションの問合せ、ロケーション・キャッシュおよびプライバシ管理があります。
- プロバイダ接続フレームワークは、データやサービスのローカル・ソースおよび外部ソースと通信し、使用可能な様々なプロバイダのコンポーネント（ビジネス・ディレクトリ、ルーティング、マッピング、モバイル・ポジショニングなど）を備えています。

ロケーション・サービスのカテゴリ

ロケーション・サービスは、ジオコーディング、マッピング、ルーティング、ビジネス・ディレクトリ（イエロー・ページ）およびトラフィックという主要カテゴリで提供されます。

他の項では、ロケーション・サービスの外部プロバイダを指定して構成する方法と、各種サービスの詳細について説明します。

前述のサービスは、すべて SpatialManager Java クラスで管理されます。SpatialManager Java クラスの定義は、次のとおりです。

```
package oracle.panama.spatial;
import ...;
public class SpatialManager
{
  public static synchronized Geocoder  getGeocoder() {...}
  public static synchronized Router  getRouter() {...}
  public static synchronized YPFinder  getYPFinder() {...}
  public static synchronized Mapper getMapper() {...}
  public static synchronized TrafficReporter getTrafficReporter() {...}
}
```

サービス・プロバイダ

通常、ロケーション・サービスのコアとなる実際の計算は、外部プロバイダ側で実行されます。外部プロバイダには、インターネットや他の通信手段を介してアクセスする場合と、ローカルの場合があります。OracleAS Wireless のロケーション・アプリケーション・コンポーネントの API では、汎用フレームワークで結果の通信と調整が実行されるため、通常、ユーザーは特定のサービスがどのプロバイダから提供されているかに気づきません。また、この API により、実装に伴うアプリケーション開発者の労力と特定のプロバイダへの依存性が最小限に抑えられます。

ほとんどのサービスについて、初期プロバイダ・セットへのアクセスが組み込まれています。プロバイダによって、詳細な構成情報を組み込んでいる場合と、そうでない場合があります。（構成情報のないプロバイダの場合、通常はそのユーザー名とパスワードの使用権を購入した後で、必要な情報を受け取ります。）

Enterprise Manager インタフェースを介して OracleAS Wireless システム・マネージャを使用することで、追加のプロバイダへのアクセスを提供できます。新規プロバイダを追加した場合や、そのプロバイダが使用するインタフェースが既存のプロバイダとは異なる場合は、新規プロバイダのフォーマットと Wireless のロケーション・アプリケーション・コンポーネントの API の間の変換を行うための Java クラスを作成する必要があります。（このプログラムを `ProviderImpl` 属性として指定します。）また、プログラムの実装用クラス・ファイルをクラス・パスに追加する必要があります。

1 つのサービスに複数のプロバイダを使用すると、そのサービスの信頼性が向上します。API が失敗するのは、すべてのプロバイダが障害を起こした場合、または Web アクセスが一時的に使用できなくなった場合のみです。プロバイダは作業環境リストで指定するため、次のいずれかの状況が発生した場合など、優先プロバイダが要求されたサービスを実行できない場合は、API が自動的にフェイルオーバーします。

- プロバイダに一時的に Web を介してアクセスできなくなった場合。
- 要求したとおりのサービスがプロバイダでサポートされていない場合。
- リクエストが正しく指定されていない場合（存在しない住所など）。

プロバイダの選択

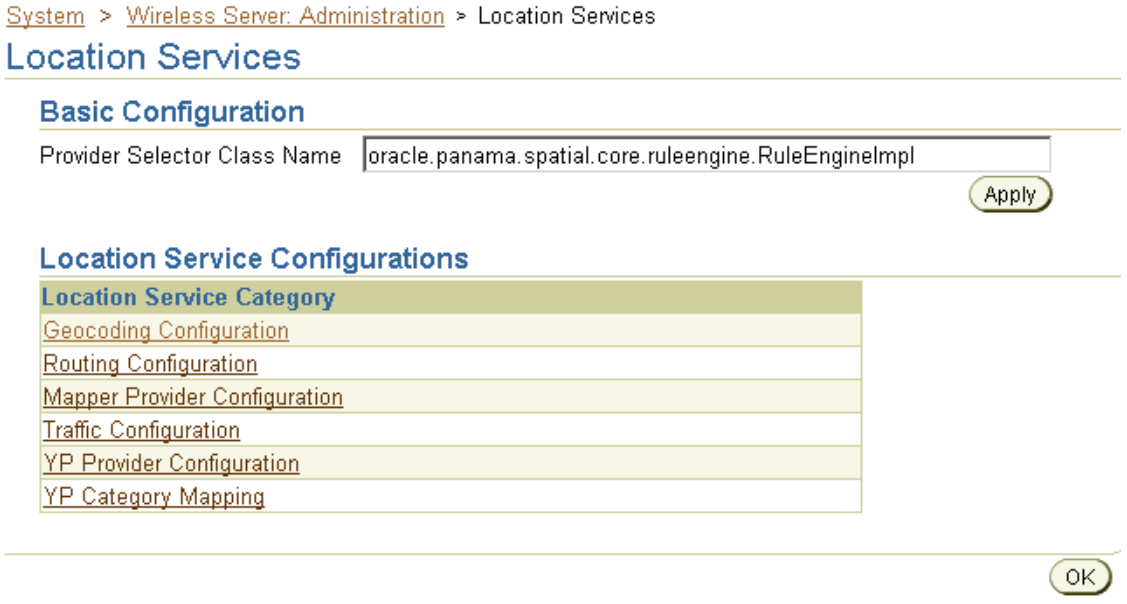
ロケーション・サービスは、プロバイダ・リストを使用して、プロバイダ間のフェイルオーバーをサポートします。プロバイダが試行される順序は、優先順位順にすると理想的です。優先順位には、単純なプロバイダのランキングを使用する方法と、リージョン、時間、パフォーマンス、信頼性およびコストを反映させる方法があります。どの基準を使用した場合も、プロバイダは、その優先順位を決定するプロバイダ選択フレームワークによって評価されます。

プロバイダ選択フレームワークを、この項の説明に従って構成する必要があります。サービス・リクエストをフレームワークが満たさない場合は、プロバイダ選択フレームワークの実装が適切に構成されていないか、すべてのプロバイダが障害を起こしています。問題や障害に関する情報は、コンソールのログまたはログ・ファイルから確認できます。詳細は、14-14 ページの「プロバイダ選択情報のロギング」を参照してください。

使用するプロバイダ選択フレームワークを選択する必要があります。フレームワークを選択するには、OracleAS Wireless システム・マネージャを使用して次の手順で操作します。

1. Wireless サーバーの**「システム」**タブのページで、**「サイト管理」**をクリックします。
2. クリックして**「コンポーネント構成」**を展開します。
3. 「構成」サブセクションで**「ロケーション・サービス」**をクリックします。図 14-4 のように「ロケーション・サービス」ページが表示されます。

図 14-4「ロケーション・サービス」ページ



「基本構成」の下の**「プロバイダ・セレクタ・クラス名」**にプロバイダ選択フレームワークの実装を入力します。選択したプロバイダ選択フレームワークの実装により、プロバイダ選択に使用できる規則の複雑度が決定されます。次の実装を使用できます。

- `oracle.panama.spatial.core.ruleengine.SimpleRuleEngineImpl`

  この単純な実装は、いずれかで成功するまですべてのプロバイダを試行します。プロバイダの試行順序は、プロバイダ構成リストで指定されます。

  ただし、この実装は、プロバイダがカバーしていないリージョンに関する問合せを発行する場合があるため、他の実装よりも時間がかかる可能性があります。また、プロバイダによっては、要求されたリージョンをカバーしていない場合、失敗するかわりに最善

の努力を試みるため、この実装は不適当な選択になる可能性があります。たとえば、ヨーロッパのみをカバーするプロバイダが、サンフランシスコからボストンにルーティングするリクエストを受信するとします。このプロバイダは最善の努力を試み、ボストンとサンフランシスコのそれぞれに最も近いヨーロッパの場所を代用します。

- `oracle.panama.spatial.core.ruleengine.RuleEngineImpl`

  この実装では、特定の国で十分な有効範囲を提供しているかどうかに基づいて、プロバイダを選択できます。この基準を満たしているすべてのプロバイダが、プロバイダ構成リストに指定されている順序で試行されます。

  この実装を使用すると、ある国で有効範囲を提供していないプロバイダや、基準を満たすサービスを提供していないプロバイダ（たとえば、コストが高すぎたりサービス品質が不十分な場合など）を試行して、時間を無駄にすることがありません。ただし、この選択フレームワークの場合は、必要な構成操作がやや多くなります。たとえば、国と国エイリアスのリストを、プロバイダごとに指定する必要があります（ただし、このような構成例は用意されています）。

- `oracle.panama.spatial.core.ruleengine.ExtendedRuleEngineImpl`

  この実装は、プロバイダのプロパティの変更にあわせて自動的に調整されます。この実装は、各プロバイダのパフォーマンスおよび信頼性を動的に測定します。これらの統計に基づいて、プロバイダ・リストが動的にランキングしなおされます。

  この実装は、現在、最速かつ最も信頼性の高いプロバイダを自動的に優先します。また、負荷が大きくなって速度が低下し、他のプロバイダと同程度になるまでは、ほぼ常に最速かつ最も信頼性の高いサービスが使用されるという点で、ロード・バランシングにも使用できます。特定の時点から、他のプロバイダは以前よりも多くのリクエストを受けることになります。

プロバイダ情報の構成 プロバイダ情報を構成するには、「ロケーション・サービス」ページの「ロケーション・サービス構成」の下で（14-9 ページの「プロバイダの選択」の図 14-4 を参照）、構成対象として適切なタイプのサービスを選択します。

- **「ジオコーディング構成」**
- **「ルーティング構成」**
- **「マッパー・プロバイダ構成」**
- **「トラフィック構成」**
- **「YP プロバイダ構成」**

プロバイダ情報（14-12 ページの「プロバイダの構成」を参照）は、どのタイプのサービス（ジオコーディング、マッピング、ルーティング、トラフィックおよび YP）の場合もほとんど同じです。

ジオコーディングとその他のサービスの場合、国名と国エイリアス（14-12 ページの「国エイリアスの構成」を参照）と住所書式（14-13 ページの「住所書式（国際）の構成」を参照）の構成情報の指定が必要になることがあります。

管理者が、(OracleAS Wireless システム・マネージャを使用して) Web サービス・プロキシを含むようにプロバイダ・リストを構成している場合は、すべてのロケーション・サービス・リクエストが自動的 (かつ透過的) に Web サービスを使用します。Web サービスの使用方法の詳細は、14-107 ページの「Web サービスの使用」を参照してください。

注意： YP カテゴリの情報を除き、すべてのロケーション・サービス構成情報は、XML 構成ファイル `site_cfg_bootstrap.xml` 内で (Wireless により) 内部的にメンテナンスされます。ただし、このファイルは直接変更しないでください。構成情報の変更には OracleAS Wireless システム・マネージャのインタフェースを使用します。

プロバイダの構成 次のパラメータを使用して、プロバイダの順序付きリストを構成します。

- 「プロバイダ名」:ID として機能するプロバイダ名。
- 「プロバイダ実装クラス名」: このプロバイダのプロキシを実装するクラス (変換およびプロバイダとの通信用)。
- 「`URL`」: プロバイダへのアクセスに使用する静的 URL 接頭辞。
- 「ユーザー名」: プロバイダによって決定されるユーザー名。
- 「パスワード」: ユーザー名と組み合せて使用されるパスワード。
- 「パラメータ」: プロバイダのプロキシのカスタマイズと構成に必要なパラメータ。
- 「`ISO` ロケール」: 国 ID がセミコロンで区切られたリスト (国エイリアス・リストで指定、14-12 ページの「国エイリアスの構成」を参照)。
- 「コーポレート `URL`」: プロバイダのコーポレート URL (広告として使用)。
- 「サービス・バージョン」: このプロキシで使用するプロバイダのサービス・バージョン。
- 「コーポレート・ロゴ `URL`」: プロバイダのコーポレート・ロゴの URL (広告として使用)。

国エイリアスの構成 国エイリアスの構成は国名に関連しており、特定の国を表す 1 つの標準識別子のシノニムです。この標準識別子には、ISO 名 (アメリカは US、ドイツは DE など) を使用する必要がありますが、他の識別子を指定できます。

エイリアスは、`oracle.panama.spatial.core.ruleengine.RuleEngineImpl` プロバイダ選択フレームワークの実装と組み合せて使用されます。各プロバイダを、ID で指定した国のセット用に構成します。たとえば、アメリカの住所をジオコーディングするなどのサービス・リクエストが送信されると、標準 ID である US を検索するために国エイリアス表が参照されます。その後は、有効範囲となっている国のリストのうち、US のプロバイダのみが試行されます。

一部の国 ID の既存のエイリアスとして構成されていない国名を使用する場合は、かわりに ID「不明」を使用します。この場合は、有効範囲となっている国のリストのうち、「不明」のプロバイダが試行されます。

単純なプロバイダ選択フレームワークの実装（`oracle.panama.spatial.core.ruleengine.SimpleRuleEngineImpl`）を使用する場合、国エイリアスはプロバイダ選択には不要です。

住所書式（国際）の構成 住所書式の構成を使用して、国際住所書式を指定します。API の `oracle.panama.spatial.intladdress` パッケージでは、このリストを使用して住所に含まれる構成要素（US、フランス、ドイツ、中国など）と入出力での表示方法が決定されます。

国際住所フレームワークを構成するには、OracleAS Wireless システム・マネージャを介してアクセス可能なリポジトリ内の住所書式リストを使用します。この構成では、住所のすべての構成要素、構成要素のエイリアス、都市、州および番地などの標準的な概念へのマッピングを指定します。次のことを決定するために、テキスト表現の書式も構成します。

- 住所を複数行に分割する場合の通常の方法
- 各構成要素の順序
- オプションの構成要素と必須の構成要素

このアプローチでは、ユーザーが国固有の書式で住所を表示し、入力するには、その書式を指定する必要があります。指定しないと、システムではユーザーに対して国の前に州と郡のどちらを指定するように求めればよいかがわかりません。

このアプローチの利点は、次のとおりです。

- ユーザーには、郵送に必要な住所書式と一致するフォームが表示されます。
- 各構成要素が単に 1 ～ 3 行目のどこかに含まれているのではなく、個別に認識され、意味のある方法で識別されている場合、システムでは住所の分析が容易になります。
- 住所の入力と出力の両方について、ローカルの住所書式に自動的に適応する場合、アプリケーションでの検索はより高度になります。
- アメリカ以外のカスタマの場合、アメリカでの書式とは異なり、ローカルの住所書式で表される方が理解しやすくなります。
- アプリケーションを様々な国にあわせてリライトする必要はありません。すべてはフレームワークにより自動的に処理されます。

複数の国際住所書式が用意されています。2 つの例を次に示します。

アメリカの場合

```
{name}
{house number/house} {street}[ Apt {apt}]
{city} {state} {postal code}[-{postal code ext}]
{country}
```

ドイツの場合

```
{Name/name}
{Strasse/street/first line} {Hausnummer/house}[ Wohnung {Wohnung/apt}]
{PLZ/postal code} {Stadt/city}
 [{Bundesland/state}]
{Land/country}
```

構文上の注意：

- { }（中カッコ）は住所の構成要素を囲みます。
- /（スラッシュ）は、構成要素内の代替エイリアスを区切ります。
- [ ]（大カッコ）はオプションの要素を囲みます。
- 大カッコ以外のカッコの外側にある構成要素は、引用符を使用して住所から区切ります（123 Main Street Apt 4 に含まれている Apt など）。

国際住所書式に関連するプログラミング情報と例は、14-97 ページの「国際住所」を参照してください。

プロバイダ選択情報のロギング

プロバイダ選択フレームワークの実装では OracleAS Wireless のログ・ファイル（`sys_panama.log` など）に、プロバイダの選択とその成否が記録されます。たとえば、次のイベントを調べることができます。

- ジオコーディング、マッピングなど（他のタイプのサービス）のマルチプレクサが初期化されていること。
- プロバイダ選択フレームワークの実装が初期化されていること。
- ジオコーディング、マッピングなど（他のタイプのサービス）のプロキシが初期化されていること。
- 特定のプロバイダが試行されたこと。
- 特定のプロバイダが失敗したこと。
- 特定のプロバイダが成功したこと。
- すべてのプロバイダが失敗したこと。

プロバイダ・パフォーマンス情報のロギング

プロバイダ選択フレームワークの実装では、プロバイダのパフォーマンスの統計が記録されます。プロバイダへのリクエストごとに、プロバイダの成功または失敗に関係なく、次の情報が記録されます。

- プロバイダ名
- プロバイダ・プロキシの Java クラス
- 所要時間（ミリ秒単位）
- 成功（true または false）
- リクエストの発行時刻（タイムスタンプ）

パフォーマンス情報は、表 `PTG_LBS_LOG` に書き込まれます。Wireless システム・マネージャを使用してこの情報を確認する手順は、次のとおりです。

1. Wireless サーバーの**「システム」**タブのページで、**「サイト・パフォーマンス」**をクリックします。
2. **「コンポーネント・パフォーマンス」**セクションで、**「関連ロケーション」**をクリックします。

ジオコーディング・サービス

ジオコーディング API は、特定の住所の地理的位置を提供します。Wireless のユーザーの場合、住所はロケーションを指定する最も一般的な方法です。ただし、付近のレストランの位置を検索したり、運転方向を提供する場合、住所のテキスト表現は、最初にジオコーディングする、つまり地理的座標に変換しなければ役に立たないことがあります。

ジオコーディングする住所には、標準的な郵便に似たテキスト表現があります。戻される結果は、住所に対応する経度と緯度です。たとえば、ジオコーディングへの入力に次の情報が含まれているとします。

- firmName: "Oracle"
- firstLine: "1 Oracle Drive"
- secondLine: ""
- lastLine: "Nashua NH 03062"
- matchMode: "tight"

この例では、結果は Point(x = -71.455, y = 42.7117) となります。

ユーザーは曖昧な住所を指定する可能性があるため、`GeocodeResult` には単一のオブジェクトではなく Location オブジェクトの配列が含まれています。

ジオコーディング API

この項では、ロケーション・アプリケーション・コンポーネント用のジオコーディング API について説明します。

次の 2 つクラス Point および Location はこの API 全体で使用され、ジオコーディング固有ではありません。ただし、ここで説明するのは、この 2 つのクラスが入力と出力の両面でジオコーディング・サービスの中心となるコンポーネントを表しているためです。

Point クラス Point クラスでは、経度と緯度の座標点が定義されます。地図上の地点を表すために、ラベルと半径を表す追加の値を使用できます。ラベルと半径は、地図に表示される他の機能には使用されません。

Location クラス Location クラスでは、住所、経度および緯度を使用してロケーションが定義されます。ロケーション・オブジェクトが firstLine、secondLine および lastLine を使用して構成されている場合、一部の外部プロバイダでは都市や州が正しく識別されない可能性があります。これは、lastLine には国固有の比較的柔軟な書式で都市、州および郵便番号を使用できるためです。

特定の部分文字列を、都市を表す構成要素として識別できない場合、都市は「不明」になります。この場合、API 自体で複雑な分析を試行するかわりに、このタスクをエキスパート、つまり外部のジオコード・プロバイダに委ねます。

Geocoder インタフェース

Geocoder インタフェースは、アプリケーション・プログラマがジオコーディング・サービスにアクセスする方法を定義します。このインタフェースを実装するクラスのオブジェクトは、SpatialManager によって戻されます。

ロケーション・マーク

電話など、ある種のモバイル・デバイスは物理的な制約から、長い英数字文字列を入力したり表示するのは困難です。**ロケーション・マーク**により、簡潔でわかりやすい名前で識別される 1 つの空間情報が格納されます。たとえば、「My home」がロケーション・マーク名で、内部的な空間情報が「123 Main Street, Somewhere City, CA, 12345; Lon = -122.42, Lat = 37.58」となる場合があります。

ロケーション・マークを使用すると、ユーザーはモバイル・デバイスに文字列を入力する手間を省くことができます。各自のロケーション・マークをデスクトップ上で管理し、その名前をモバイル・デバイスから参照してロケーションにアクセスできます。通常、今日のロケーション認識アプリケーションでは、地点（住所や交差点など）のみが使用されています。この場合は、ジオコーディングにより空間情報を提供できます。ただし、ロケーション・マークは、1 つのポイントを中心とする円形の地域（つまり、ポイントと半径を指定する場合）、またはリージョン・モデル・ツール（14-130 ページの「リージョン・モデル・ツールの使用」を参照）で定義されたリージョンの場合もあります。

また、ロケーション・マークを使用すると、ユーザーは what-if シナリオを試行して、アプリケーションをデフォルト・ロケーションや現在のロケーションとは異なるロケーションにあるかのように動作させることができます。たとえば、エンタテインメント・サービス・アプリケーションのユーザーが実際にはボストンにいて、数日後にサンフランシスコに旅行するとします。このユーザーがデフォルトとして San Francisco にロケーション・マークを設定すると、San Francisco リージョンに関連する情報とともに表示されます。

各ユーザーはパーソナライズされたロケーション・マークを使用できます。これらは Wireless リポジトリに格納されます。

デフォルト・ロケーション・マークは、ユーザーごとに設定できます。あるユーザーにモバイル・ポジショニング情報がない場合は、そのユーザーのデフォルト・ロケーション・マーク（定義済の場合）が使用されます。たとえば、ユーザー Smith のデフォルト・ロケーション・マークがユーザーのオフィスで、Smith の現在のポジショニング情報がない場合、現在のロケーションは Smith のオフィスとみなされます。

ロケーション・マークは `LocationMark` クラスを使用して作成します。また、ユーザーがロケーション・マークを作成するには、Customization Portal にログインし、**「ロケーション・マーク」**タブをクリックして**「作成」**をクリックする方法もあります。

ロケーション・マークを使用してモバイル・ポジショニングを使用可能にする方法の詳細は、14-110 ページの「手動ポジショニング」を参照してください。

LANDMARK 表

Wireless のリポジトリ・スキーマには、新しい表 LANDMARK が追加されています。この表には、関連付けられているユーザーなど、各ロケーション・マークの詳細情報が含まれています。たとえば、複数のユーザーが、1 つのロケーション・マーク Office をそれぞれ異なるロケーションで使用している場合があります。

表 14-1 に、LANDMARK 表の列を示します。

表 14-1 LANDMARK 表の列

列名	型
objectId_	NUMBER(10)
name	VARCHAR2(32)
longitude	NUMBER
latitude	NUMBER
addrline1	VARCHAR2(256)
addrline2	VARCHAR2(256)
addrlastline	VARCHAR2(256)

表 14-1 LANDMARK 表の列（続き）

列名	型
block	VARCHAR2(256)
city	VARCHAR2(256)
country	VARCHAR2(256)
county	VARCHAR2(256)
firmname	VARCHAR2(256)
pcode	VARCHAR2(32)
pcode_ext	VARCHAR2(16)
state	VARCHAR2(256)
matchmode	VARCHAR2(32)
description	VARCHAR2(256)

マッピング・サービス

マッピング API は、次のいずれかのマップ・イメージを作成する機能を提供します。

- 単一の点（住所やロケーション・マークなど）
- 複数の点（複数の住所やロケーション・マークなど）
- ルート全体
- 単一の運転経路

マッピング API を使用すると、地図のサイズ（解像度）とイメージ・フォーマットを指定できます。

マッピング機能は、純マッピング・アプリケーションとして、またはルーティング・アプリケーションの一部としてユーザーに表示できます。ルーティング・アプリケーションでは、ルートと運転経路のマッピングがルーティング・プロバイダにより実行されます。ルーティング・サービスの詳細は、14-19 ページの「ルーティング・サービス」を参照してください。

ルーティング・サービス

ルーティング API は、始点、終点およびオプションの経由地リストに基づいて、ルーティング（運転方向）情報を提供します。すべての点は、緯度と経度のペアまたは住所として指定されます。

ルーティング結果は、経路セットで構成されます。A **経路**は、「左折して I-93 に」または「右手に向かってルート 3 に合流」などの運転指示に対応します。ルーティング結果には、走行合計時間の見積と距離も含まれています。オプションで、地図とルートの座標も要求できます。

ルーティング設定

ルーティングには、作業環境や要件などのルーティング・オプションを反映できます。これらのオプションは**ルーティング設定**と呼ばれるセットにまとめられています。ルーティング・オプションには、基本オプションと 2 次オプションの 2 種類があります。

基本オプションには、次の項目が含まれます。

- マップ（イメージ）を要求するかどうか。
- ジオメトリ（ルートの座標）を要求するかどうか。

2 次オプションには、次の項目が含まれます。

- 最短距離や最短走行時間などの最適化方法
- 有料道路の回避、フェリーの回避、高速道路の回避などのルート・プロパティ
- マップ・サイズ

2 次オプションは必須か任意かを指定できます。

- 2 次オプションが必須であってもプロバイダがサポートしていない場合、API は自動的に次のプロバイダにフェイルオーバーします。
- 2 次オプションが任意であってプロバイダがサポートしていない場合、API は別のプロバイダがそのオプションをサポートしているかどうかをチェックしません。

アプリケーション開発者が必須か任意かを指定せずに 2 次オプションを要求すると、次のデフォルト値が適用されます。

- 最適化メソッド : 任意
- フェリーの回避 : 任意
- 高速道路の回避 : 任意
- 有料道路の回避 : 任意
- 概観マップのサイズ : 必須
- 経路マップのサイズ : 必須

- 概略マップのスケールとズーム・レベル : 任意
- 経路マップのスケールとズーム・レベル : 任意

ルーティング結果

アプリケーションでは、戻されたルートの次のコンポーネントを問合せできます。

- 経路リスト
- 全距離
- 走行合計時間の見積り
- 概観マップ

概観マップには、出発地と目的地が表示され、ルートがハイライトされます。

ルーティング結果の一部として経路（運転方向）セットが戻されます。各経路は運転指示に対応し、次の情報が含まれます。

- テキストによる説明
- この経路またはこの経路までの移動距離（これまでに何マイル移動したか）
- 詳細な経路マップ
- ジオメトリ（座標点、緯度および経度のリスト）

ルートまたは経路全体のマップを、Java Image オブジェクトまたは URL を表す文字列として要求できます。

複数言語のサポート

ルーティング・プロバイダが複数の言語をサポートしている場合、この API は、ルーターへのリクエストで指定された Java `locale` オブジェクトに基づいて言語を選択します。言語設定によって、経路説明および距離の測定単位が異なる場合があります。

ルーティング API

この項では、ロケーション・アプリケーション・コンポーネント用のルーティング API について説明します。

Router インタフェース `Router` インタフェースは、アプリケーション・プログラマがルーティング・サービスにアクセスする方法を定義します。このインタフェースを実装するクラスのオブジェクトは、`SpatialManager` によって戻されます。

RoutingSettings クラス `RoutingSettings` クラスは、ルーティングに渡されるオプションのセットを定義します。ルーティング・オプションには、基本オプションと 2 次オプションの 2 種類があります。

基本オプションには、マップまたはジオメトリをリクエストするかどうかが含まれます。基本オプションは、RoutingSettings オブジェクトのコンストラクタで指定できます。

2 次オプションは、setSecondaryOption を使用して設定できます。最初のパラメータは RoutingOption オブジェクトで、RoutingOption クラスで定義する静的定数です。このパラメータは、値が設定されているオプションを識別します。2 番目のパラメータは、値を表す文字列です。

2 次オプションが必須かどうかは、setSecondaryOptionRequired で定義します。最初のパラメータは RoutingOption で、2 番目のパラメータによってオプション要件が必須かどうかが指定されます。このファンクションをコールしない場合は、デフォルト値が使用されます。

RoutingResult クラス RoutingResult クラスは、ルーティング結果を定義します。詳細は、14-20 ページの「ルーティング結果」を参照してください。

Maneuver クラス Maneuver クラスは、1 つのルート内に 1 つの経路を定義します（経路属性については、14-20 ページの「ルーティング結果」を参照）。

ビジネス・ディレクトリ（イエロー・ページ）サービス

ビジネス・ディレクトリ（イエロー・ページまたは YP）サービスは、特定の領域内で指定された名前またはカテゴリと一致するビジネスのリストを提供します。

既存のプロバイダは、YP サービスを異なるインタフェースで提供しています。特に、YP カテゴリと階層構造もすべて異なっています。カテゴリは、一覧表形式で編成されている場合と、カテゴリおよびサブカテゴリからなる階層形式で編成されている場合があります。階層ツリーは、分岐が多く深い場合や分岐が少なく浅い場合、またはバランスが取れている場合や取れていない場合があります。

異なるプロバイダのサービスを統一するために、Oracle ビジネス・ディレクトリ・サービスでは、OracleAS Wireless 開発者が XML ファイルに定義しているカスタム階層が使用されます。この階層内の各リーフには、1 つ以上のプロバイダのカテゴリへの参照が設定されています。リーフでないノードにも、この参照を設定できます。このカスタム階層では、最初に優先カテゴリが定義されます。その後、OracleAS Wireless を使用している電話会社は、これらのカテゴリと外部プロバイダがサポートしている類似のセマンティクスを持ったカテゴリとの一致を試みます。

外部プロバイダへの参照を持ちカスタマイズされた階層は、順序付けられた階層構造が格納された XML ファイルで表されます。カテゴリ階層での表示順序には、様々なカテゴリの知名度を考慮できます。たとえば、画面サイズが限られているデバイスの場合、アプリケーションでは最も一般的なカテゴリの中で選択肢を制限できます。

イエロー・ページ・プロバイダ間で異なるアプローチ

複数のプロバイダが Web 上で YP サービスを提供していますが、各プロバイダのアプローチには大きな違いがあり、汎用インタフェースは提供されていません。また、それぞれのアプローチはプロバイダのメソドロジで確定されてはおらず、変更されることを予想できます。

各種アプローチの統一パターンは、ビジネスが件名別とロケーション別に分類されることです。ロケーション・コンポーネントは、郵便番号または都市と州の組合せを使用してロケーションを決定できるという点で十分に認識されています。

これに対して、ビジネス・カテゴリの実装方法は統一されていません。カテゴリの一覧表を提供し、ユーザーに単純な部分文字列の一致により選択させているプロバイダや、サブカテゴリを 3 または 4 レベルの階層に編成しているプロバイダがあり、後者の場合、分岐は 20 ～ 50、または 100 以上になることもあります。ユーザーは、階層のルート（デフォルト）から横断を開始できます。または、階層内の適切な開始ポイントと一致するキーワードを入力することもできます。このキーワードによる一致では、単純な部分文字列の検索よりも詳細で、より適切な選択結果が得られます。

ビジネス・ディレクトリのカテゴリ構成

ビジネス・カテゴリとカテゴリ階層のサポートは、XML 構成ファイルを介して提供されます。（ビジネス・ディレクトリ・プロバイダ情報の表示と変更には OracleAS Wireless システム・マネージャを使用する必要がありますが、ビジネス・ディレクトリ・カテゴリ情報の表示と変更には XML ファイルを使用してください。）

例 14-1 のカテゴリ階層定義ファイルは、ビジネス・ディレクトリ・カテゴリのカスタム階層を表しています。各カテゴリには、必要な数のサブカテゴリを含めることができます。ネスト・レベルの制限はありません。1 つのカテゴリを複数のビジネス・ディレクトリ・コンテンツ・プロバイダにリンクできます。このファイルで得られる柔軟性により、14-22 ページの「イエロー・ページ・プロバイダ間で異なるアプローチ」で説明したように、各種のビジネス・ディレクトリ・サービス・プロバイダの様々なアプローチに対処できます。

例 14-1 ビジネス・ディレクトリのカテゴリ階層定義ファイル

```
<?xml version="1.0" standalone="yes"?>
<Categories>
  ...
  <Category
    CategoryName  = "Berry crops">
    <Provider
      Name  = "..."
      Parameter  = "..."/>
      <Category
      CategoryName  = "Cranberry farm">
      <Provider
        Name  = "..."
        Parameter  = "..."/>
    </Category>
```

```
  </Category>
  ...
  <Category
    CategoryName  = "Ornamental nursery products">
    <Provider
      Name  = "..."
      Parameter  = "..."/>
    <Category
      CategoryName  = "Florists' greens and flowers">
      <Provider
        Name  = "..."
        Parameter  = "..."/>
    </Category>
      <Category
      CategoryName  = "Bulbs and seeds">
      <Provider
        Name  = "..."
        Parameter  = "..."/>
    </Category>
  </Category>
  <Category
    CategoryName  = "Crops grown under cover">
    <Provider
      Name  = "..."
      Parameter  = "..."/>
    <Category
      CategoryName  = "Mushrooms grown under cover">
      <Provider
        Name  = "..."
        Parameter  = "..."/>
    </Category>
  </Category>
  ...
</Categories>
```

ビジネス・ディレクトリ（イエロー・ページ）API

アプリケーション開発者は、YPFinder インタフェース内のファンクションを使用してカテゴリ階層を横断できます。結果となるカテゴリには、次のリストを要求できます。

- ビジネスのリスト
- 直接のサブカテゴリのリスト
- 部分文字列を含む直接または間接的なサブカテゴリのリスト

YPFinder インタフェース YPFinder インタフェースは、アプリケーション・プログラマが YP サービスにアクセスする方法を定義します。このインタフェースを実装するクラスのオブジェクトは、SpatialManager によって戻されます。

このクラスのオブジェクトにより、ユーザーは次のビジネスを問合せできます。

- 特定の州内のビジネス
- 特定の都市内のビジネス
- 特定の郵便番号に該当するビジネス
- 特定の半径に含まれるビジネス
- 特定の半径内で最も近接している *n* 件のビジネス

これらのリージョン・タイプごとに、次の条件でビジネスを検索できます。

- 指定したビジネス名またはキーワードと一致
- 指定したカテゴリと一致
- 指定したビジネス名またはキーワードおよび指定したカテゴリの両方と一致
- ビジネス名またはカテゴリに含まれるキーワードと一致

YPCategory クラス YPCategory クラスは、階層の一部である単一のカテゴリを定義します。このクラスでは、ユーザーはそのカテゴリ内のビジネスにアクセスできます。また、カテゴリ内で特に次のサブカテゴリを検索できます。

- 直接のすべてのサブカテゴリ
- キーワードと一致する直接または間接的なすべてのサブカテゴリ
- 指定した名前を持つサブカテゴリ

最もポピュラーなアプリケーションの 1 つは、指定したキーワードと一致するルートのサブカテゴリを検索することです。

YPBusiness クラス `YPBusiness` クラスは、単一のビジネスを定義します。このクラスは住所（`Location` インタフェース）を表し、電話番号、説明および一致するカテゴリのリストも含まれます。これらのビジネスごとに、あるカテゴリのすべてのビジネス、またはすべてのカテゴリを取得できます。たとえば、特定の書店が書店とカフェという 2 つのカテゴリに含まれている場合があります。

トラフィック・サービス

トラフィック API は、主要都市の道路網の交通量に影響のある条件の情報を提供します。通常、このようなエリアはさらに中心市街、西地区、東地区などの小さいエリアに分割されています。リアルタイムのトラフィック・レポートでは、条件が短期間（5 分ごとなど）で更新されるため、営業車両の管理や個人の移動にとって重要な情報を提供します。

トラフィック・レポートの主要なコンポーネントは事象です。**事象**は、交通量に影響すると思われるイベントです。事象の例には、事故、建設作業および交通渋滞（通常または予想外）などがあります。事象ごとに、事象のタイプ、発生場所（ルート番号、ロケーションまたはリージョンなど）、ルート沿いの方向（北部方面など）、予想される遅延および交通渋滞の長さなどの情報が含まれます。

現行のリリースでは、事象ベースのトラフィック情報について次の問合せがサポートされています。

- 都市レベルの問合せ : 都市全体のトラフィックに関連する事象を戻します。
- ルート・レベルの問合せ : 都市内の指定したルートでのトラフィックに関連する事象を戻します。
- 緯度と経度（ポイント）または住所と半径レベルの問合せ : 要求した半径内でのトラフィックに関連する事象を戻します。

トラフィック問合せの例では、次のようなトラフィック・レポートが戻されます。

- 都市（Boston など）
- 都市のルート（Boston の I-93 South など）
- （ルート、都市）のコレクションとして戻される予定ルート（Nashua、NH から Boston、MA など）
- ロケーション（緯度、経度）とロケーションからの半径で構成される移動範囲
- 特定の住所の近接度（One Oracle Drive、Nashua、NH 03062 など）

トラフィック API は、リクエストを処理してレスポンスを戻します。リクエストとレスポンスには、Java フォーマットまたは XML フォーマットを使用できます。XML フォーマットによる XML リクエストとレスポンスの例は、14-26 ページの「トラフィックに関する XML リクエストとレスポンス」を参照してください。トラフィック Java API の説明は、14-28 ページの「トラフィック Java API」を参照してください。

注意： 現行のリリースのサンプル構成ファイルには、トラフィック・サービス・プロバイダは含まれていません。

トラフィック・レポートのキャッシュ

トラフィック・レポート情報は、都市レベルでキャッシュされます。初めてフェッチされた都市のトラフィック・レポートは、トラフィック・レポート・キャッシュに書き込まれます。キャッシュされたレポートは、最大キャッシュ経過期間（15 分など）の後は無効とみなされます。この期間はシステム・マネージャを使用して設定できます。

都市に関してキャッシュされたトラフィック・レポートを更新するには、ネットワーク上でトラフィック・サービス・プロバイダにラウンドトリップする操作が必要です。キャッシュされたレポートは、次の両方の条件と一致する場合にのみ更新されます。

- 特定の都市、またはその都市に全体または一部が含まれている空間ジオメトリ（ルートまたはポイントと半径）の問合せが行われた（つまり、問合せ対象のジオメトリと都市のジオメトリに空間的な相互作用がある）場合。
- 都市に関してキャッシュされたレポートが、最大キャッシュ経過期間を超えている場合。

トラフィックに関する XML リクエストとレスポンス

例 14-2 に、ボストンのトラフィック情報に関する都市レベルのリクエストを XML フォーマットで示します。

例 14-2 Boston に関するトラフィック・リクエスト

```
<?xml version="1.0" encoding="UTF-8"?>
<traffic_request>
  <query_list>
    <query_info
query_type="city_level_query"
city_name="boston"
state_name="MA"
country_name="US"
    />
  </query_list>
</traffic_request>
```

例 14-3 に、ボストンのトラフィック情報に関する XML フォーマットのレスポンスを示します。

例 14-3 ボストンに関するトラフィック・レスポンス

```
<?xml version="1.0"  encoding="UTF-8"?>
<traffic_response>
 <report_list>
  <traffic_report>
    <provider
    name="Trafficstation"
    covered_city_name="Boston"
    state_name="MA"
    country_name="US"/>
    <report_time month="6" day="19" year="2001" hour="5" minute="28" meridian =
      "PM"/>
      <unit distance_unit="MILES" time_unit="MINUTE"/>
    <incident_list>
      <incident id = "1">
        <incident_type>ACCIDENT</incident_type>
        <description>CAR ACCIDENT</description>
        <route type = "Interstate" name = "I-93" direction = "SOUTH"/>
        <geo_location  longitude = "-71.0607" latitude  = "42.3659" radius =
           "5.0"/>
        <location_range>
          <at_location>EXIT 26</from_location>
        </location_range>
        <time_range>
          <from_time  month = "6" day = "19" year = "2001" hour = "5" minute =
             "28" meridian = "PM"/>
          <to_time  month = "6" day = "19" year = "2001" hour = "5" minute =
             "28" meridian = "PM"/>
         </time_range>
        <severity>HEAVY</severity>
        <speed>15.0</speed>
        <impact>EXPECT DELAY</impact>
        <advice>TAKE LEFT LANE</advice>
      </incident>
      <incident id = "2">
        <incident_type>CONSTRUCTION</incident_type>
        <description>REGULAR MAINTENANCE</description>
        <route type = "Interstate" name = "I-95" direction = "NORTH"/>
        <geo_location  longitude = "-71.3555" latitude  = "42.3601" radius =
          "30.0"/>
        <location_range>
          <at_location>EXIT 36</at_location>
        </location_range>
```

```
        <time_range>
          <at_time  month = "6" day = "19" year = "2001" hour = "5" minute =
             "28" meridian = "PM"/>
         </time_range>
        <severity>MINOR</severity>
        <speed>35.0</speed>
        <impact>EXPECT DELAY</impact>
        <advice>USE I-495</advice>
      </incident>
    </incident_list>
  </traffic_report>
 </report_list>
</traffic_response>
```

トラフィック Java API

この項では、ロケーション・アプリケーション・コンポーネント用のトラフィック Java API について説明します。

CityInfo クラス CityInfo クラスは、1 つの都市の名前、州名および国名を提供します。このクラスの一般的な用途は、都市名、州名（オプション）および国名を使用して CityInfo インスタンスを作成し、それを都市レベル、ルート・レベルまたは都市のポイントおよび半径レベルでトラフィック・レポートに対する問合せに渡すことです。

City インタフェース City インタフェースは、指定されたサービス・プロバイダからの 1 つの都市に関する情報を提供します。この情報には、都市名、州名、国名およびルート情報が含まれます。City インスタンスは、TrafficReport インタフェースから取得できます。

RouteInfo クラス RouteInfo クラスは、ルートの名前とタイプを提供します。このクラスの一般的な用途は、RouteInfo インスタンスを作成し、それをルート・レベルのトラフィック・レポートの問合せに渡すことです。

TrafficRoute インタフェース TrafficRoute インタフェースは、指定されたサービス・プロバイダからの 1 つのルートに関する情報を提供します。この情報には、ルート名、ルートのタイプ、ルートを表すジオメトリおよび都市名が含まれます。TrafficRoute インスタンスは、TrafficIncident インタフェースから取得できます。

TrafficReport インタフェース TrafficReport インタフェースは、レポート時刻、事象数、プロバイダ情報、都市および事象など、事象ベースのトラフィック・レポートに関する情報を提供します。レポートを作成して、アプリケーションのユーザーまたは管理者に表示できます。

TrafficIncident インタフェース TrafficIncident インタフェースは、重大度、タイプ、説明、発生したルートと方向、ロケーション、時間帯、影響およびアドバイスなど、トラフィック関連の事象に関する情報を提供します。

TrafficReporter インタフェース TrafficReporter インタフェースは、様々な問合せに基づいてトラフィック・レポートを戻すファンクションを提供します。次の種類の問合せがサポートされます。

- 都市に関する情報（都市名、州名 [オプション]、国名）を指定すると、そのレポートが戻されます。
- ルート（方向付きまたは方向なし）とそのルートを含む都市に関する情報を指定すると、そのレポートが戻されます。
- ポイントの緯度と経度の座標と半径を指定すると、エリアのレポートが戻されます。
- ロケーションの住所と半径を指定すると、エリアのレポートが戻されます。

SpatialManager.createLocation() を使用して Location のインスタンスを取得する場合は、都市名と国名を指定する必要があります。これらの各情報は LastLine 属性を使用して組み合せないでください。Point ジオメトリの値を null に設定すると、自動ジオコーディングを回避できます。

TrafficCityManager インタフェース TrafficCityManager インタフェースは 2 つのファンクションを提供します。一方はトラフィック情報が提供されるすべての都市を取得するファンクションで、他方は指定された都市のルート情報を取得するファンクションです。この 2 つのファンクション・コールの一般的な用途は、アプリケーションでサポートされている都市とルートのドロップダウン・リストを作成することです。

トラフィック・サービスの構成

Wireless のインストール中にリージョン・モデリング・データと都市有効範囲データがリポジトリにロードされた後、トラフィック・プロバイダとそのプロバイダでサポートされている都市を追加できます。

トラフィック・プロバイダの追加 新規のトラフィック・サービス・プロバイダに対するサポートを追加する手順は、次のとおりです。

1. システム・マネージャを使用して、トラフィック・プロバイダ情報とトラフィック・レポートのキャッシュ時間を設定します。
2. この新規プロバイダでサポートされている都市ごとに、リージョン・モデル・ツール（14-130 ページの「リージョン・モデル・ツールの使用」を参照）を使用して、有効な GEOMETRY 列の値など、その都市の CITY 表にエントリがあるかどうかを調べます。ジオメトリなど、都市のエントリがない場合は追加します。
3. この都市の ID を取得してメモします。
4. SQL*Plus を使用して Wireless リポジトリに接続します。
5. このトラフィック・サービス・プロバイダでサポートされている都市ごとに、CITY_COVERAGE 表の COVERED_BY_TRAFFIC 列の値を 'Y' に設定し、都市の ID 値を使用して更新を実行します。次に例を示します。

```
UPDATE city_coverage SET covered_by_traffic = 'Y' WHERE id = 12345;
COMMIT;
```

この都市の CITY_COVERAGE 表にエントリが存在しない場合は、1 行を追加して COVERED_BY_TRAFFIC 列の値を 'Y' に設定し、この表内の ID 値が CITY 表内の都市の ID 値と同じであることを確認します。次に例を示します。

```
INSERT INTO city_coverage (id, name, state_name, country_name,
     covered_by_traffic) VALUES (10750, 'BOSTON', 'MA', 'US', 'Y');
COMMIT;
```

プロバイダでサポートされている都市の追加 既存のトラフィック・サービス・プロバイダの新規の都市に対するサポートを追加する手順は、次のとおりです。

1. この新規プロバイダでサポートされている都市ごとに、リージョン・モデル・ツール（14-130 ページの「リージョン・モデル・ツールの使用」を参照）を使用して、有効な GEOMETRY 列の値など、その都市の CITY 表にエントリがあるかどうかを調べます。ジオメトリなど、都市のエントリがない場合は追加します。
2. この都市の ID を取得してメモします。
3. SQL*Plus を使用して Wireless リポジトリに接続します。
4. この都市の CITY_COVERAGE 表にエントリがある場合は、CITY_COVERAGE 表の COVERED_BY_TRAFFIC 列の値を 'Y' に設定し、都市の ID 値を使用して更新を実行します。次に例を示します。

   ```
   UPDATE city_coverage SET covered_by_traffic = 'Y' WHERE id = 12345;
   COMMIT;
   ```

   この都市の CITY_COVERAGE 表にエントリが存在しない場合は、1 行を追加して COVERED_BY_TRAFFIC 列の値を 'Y' に設定し、この表内の ID 値が CITY 表内の都市の ID 値と同じであることを確認します。次に例を示します。

   ```
   INSERT INTO city_coverage (id, name, state_name, country_name,
        covered_by_traffic) VALUES (10750, 'BOSTON', 'MA', 'US', 'Y');
   COMMIT;
   ```

ロケーション・ベースのアプリケーションの開発

次のいずれかのアプローチを使用して、ロケーション・ベースのアプリケーションを開発できます。

- Mobile XML タグまたは HTML タグ、あるいはその両方とオラクル社が提供するカスタム・タグを含む JavaServer Pages（JSP）ファイルの作成（14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）
- JSP ファイルでの Java API の使用（14-96 ページの「ロケーション Java API の使用」を参照）
- Web サービスの使用（14-107 ページの「Web サービスの使用」を参照）

Java Application Programming Interface（API）を使用するよりは、JSP ファイルでタグを使用する方が容易で便利ですが、API を使用することで柔軟性が得られ、アプリケーション・ロジックを制御できます。

JavaServer Pages（JSP）ファイルの作成

アダプタを作成する必要がない場合は、ユーザーにロケーション・ベースの機能を提供する JavaServer Pages（JSP）ファイルを作成します。

この項では、オラクル社が提供するタグの使用の詳細を説明します。それぞれの項に例が含まれています。

表 14-2 に、ロケーション・サービス用の JSP タグと各タグで指定する情報を、各タグが役立つアプリケーションのタイプ別に示します。

表 14-2　ロケーション・サービス用の JSP タグ

カテゴリ	タグ
全般	geometry
	point
ジオコーディング	address
	geocode
	iterateGeocodes
	iterateReverseGeocodes
	listGeocodes
	listReverseGeocodes
マッピング	map

表 14-2（続き）ロケーション・サービス用の JSP タグ（続き）

カテゴリ	タグ
ルーティング	drivingDistance
	drivingTime
	iterateManeuvers
	listManeuvers
	route
ビジネス・ディレクトリ（YP）	businesses
	category
	iterateBusinesses
	iterateBusinessesInCity
	iterateBusinessesInCorridor
	iterateBusinessesInPostalCode
	iterateBusinessesInRadius
	iterateBusinessesInState
	iterateBusinessesNearestTo
	iterateCategoriesMatchingKeyword
	iterateChildCategories
	listBusinessesInCity
	listBusinessesInCorridor
	listBusinessesInPostalCode
	listBusinessesInRadius
	listBusinessesInState
	listBusinessesNearestTo
	listCategoriesMatchingKeyword
	listChildCategories

表 14-2（続き）ロケーション・サービス用の JSP タグ（続き）

カテゴリ	タグ
ソート	iterateByDistance
	iterateByDrivingDistance
	iterateByName
	iterateByRegionName
	listByDistance
	listByDrivingDistance
	listByName
	listByRegionName
ロケーション・マーク	defaultLocationMark
	iterateLocationMarks
	listLocationMarks
モバイル・ポジショニング	mobilePos
コミュニティ	addMembers
	createPrivateCommunity
	createSharedCommunity
	createSystemCommunity
	deleteCommunity
	getCommunity
	listAllMembers
	listCreatedCommunities
	listCreatedPrivateCommunities
	listCreatedSharedCommunities
	listCreatedSystemCommunities
	removeAllMembers
	removeMembers
	setCommunityName

一方が iterate で始まり、もう一方が list で始まる、類似した名前を持つタグのペアが多数あります（たとえば、iterateManeuvers と listManeuvers があります）。

- iterate で始まる名前を持つタグは、コレクションを作成し、そのコレクション内の各項目を個別に表すため、ユーザーは特定の処理を実行できます。たとえば、Web ページで各項目の後に横罫線を表示したり、各項目の前に静的テキストを表示できます。
- list で始まる名前を持つタグは、戻り項目のフォーマットされていないリストを表します。これらのタグは、スクリプトやアルゴリズムで処理する目的で、単一のデータ・リストを渡す場合に役立ちますが、通常、Web ページに直接表示するデータには使用されません。

ロケーション・サービス用の JSP タグは接頭辞とともに使用し、接頭辞は JSP ファイルで指定する必要があります。次の例では、loc 接頭辞を定義しています。この接頭辞は、特定のタグの他の例に使用されています。

```
<%@ taglib uri="LocationTags" prefix="loc" %>
```

次の例に、address タグとともに使用されている loc 接頭辞を示します。

```
<loc:address name="hq" type="oracle.panama.model.Location"
    businessName="Oracle Headquarters" firstLine="500 Oracle Parkway"
    city="Redwood City" state="CA" postalCode="94065" country="US"/>
```

14-34 ページの「ロケーション・サービス用の JSP の例」では、ロケーション・サービス用の包括的な JSP の例について説明します。以降の各項（タグ名のアルファベット順）には、各タグに使用可能なすべてのパラメータの参照情報、つまりパラメータ名、説明および必須かどうかを記載しています。パラメータが必須の場合は、タグに含める必要があります。パラメータが必須でない場合は、省略するとサービス・プロバイダにより解析が実行されます。それぞれのタグの項にも、短い例が含まれています。

ロケーション・サービス用の JSP の例

この項では、ロケーション・サービスに関連する操作を実行するための JSP コードの例を示します。これらの例では、住所は該当するタグ（<map> または <route>）の points 属性に指定されています。

例 14-4 では、2 つのロケーションの大きいマップと小さいマップが表示されます。

例 14-4 JSP タグを使用したマッピング

```
<%@ taglib uri="LocationTags" prefix="loc" %>

<%!
  public String transformString(String orig)
  {
    String result = "";
    for (int i=0;i<orig.length();i++)
    {
```

```
      if (orig.charAt(i) == '&')      result = result + "&amp;";
      else if (orig.charAt(i) == '<') result = result + "&lt;";
      else if (orig.charAt(i) == '>') result = result + "&gt;";
      else                            result = result + orig.charAt(i);
    }
    return result;
  }
%>

<SimpleResult>
    <loc:address
      name="NEDC"
      type="oracle.panama.model.Location"
      businessName="NEDC"
      firstLine="1 Oracle Dr"
      city="Nashua"
      state="NH"
      postalCode="03062"
      country="US"/>
    <loc:map
      name="NEDCSmall" type="oracle.panama.spatial.jsptags.beans.Map" xres="400"
         yres="300" points="NEDC">
    </loc:map>

    <loc:address
      name="HQ"
      type="oracle.panama.model.Location"
      businessName="HQ"
      firstLine="500 Oracle Parkway"
      city="Redwood City"
      state="CA"
      postalCode="94065"
      country="US"/>
    <loc:map name="HQSmall" type="oracle.panama.spatial.jsptags.beans.Map"
       xres="400" yres="300" points="HQ">
    </loc:map>

    <loc:map name="BothSmall" type="oracle.panama.spatial.jsptags.beans.Map"
       xres="400" yres="300" points="NEDC HQ"/>
    <loc:map name="NEDCLarge" type="oracle.panama.spatial.jsptags.beans.Map"
       xres="800" yres="600" points="NEDC"/>
    <loc:map name="HQLarge"   type="oracle.panama.spatial.jsptags.beans.Map"
       xres="800" yres="600" points="HQ"/>
    <loc:map name="BothLarge" type="oracle.panama.spatial.jsptags.beans.Map"
       xres="800" yres="600" points="NEDC HQ"/>
```

```
    <SimpleImage target="<%= transformString(NEDCLarge.toString()) %>"
       src="<%= transformString(NEDCSmall.toString()) %>"/>

    <SimpleImage target="<%= transformString(HQLarge.toString()) %>"
       src="<%= transformString(HQSmall.toString()) %>"/>

    <SimpleImage target="<%= transformString(BothLarge.toString()) %>"
       src="<%= transformString(BothSmall.toString()) %>"/>
</SimpleResult>
```

例 14-5 では、2 つのロケーション間のルートと運転方向（経路）が表示されます。

例 14-5 JSP タグを使用したルーティング

```
<%@ taglib uri="LocationTags" prefix="loc" %>

<%!
  public String transformString(String orig)
  {
    String result = "";
    for (int i=0;i<orig.length();i++)
    {
      if (orig.charAt(i) == '&')      result = result + "&amp;";
      else if (orig.charAt(i) == '<') result = result + "&lt;";
      else if (orig.charAt(i) == '>') result = result + "&gt;";
      else                            result = result + orig.charAt(i);
    }
    return result;
  }
%>

<SimpleResult>
  <loc:address
    name="NEDC"
    type="oracle.panama.model.Location"
    businessName="NEDC"
    firstLine="1 Oracle Dr"
    city="Nashua"
    state="NH"
    postalCode="03062"
    country="US"/>
  <loc:address
    name="HQ"
    type="oracle.panama.model.Location"
    businessName="HQ"
    firstLine="500 Oracle Parkway"
    city="Redwood City"
```

```
    state="CA"
    postalCode="94065"
    country="US"/>
  <loc:route name="myRoute" type="oracle.panama.spatial.jsptags.beans.Route"
     xres="800" yres="600" points="NEDC HQ">
  </loc:route>

  <SimpleImage src="<%= transformString(myRoute.getMap()) %>"/>

  <SimpleText>
    <loc:iterateManeuvers name="aManeuver"
type="oracle.panama.spatial.jsptags.beans.Maneuver" routeID="myRoute">
      <SimpleTextItem>
        <%= aManeuver.getNarrative() %>
      </SimpleTextItem>
    </loc:iterateManeuvers>
  </SimpleText>
</SimpleResult>
```

例 14-6 では、ロケーションから指定した範囲内にある名前別のビジネス・ディレクトリ（YP）情報、特に、Oracle 本社に最も近接している 10 軒のスターバックスを含むマップが表示されます。

例 14-6 JSP タグを使用した名前別のビジネス・ディレクトリ（YP）

```
<%@ taglib uri="LocationTags" prefix="loc" %>

<%!
  public String transformString(String orig)
  {
    String result = "";
    for (int i=0;i<orig.length();i++)
    {
      if (orig.charAt(i) == '&')      result = result + "&amp;";
      else if (orig.charAt(i) == '<') result = result + "&lt;";
      else if (orig.charAt(i) == '>') result = result + "&gt;";
      else                            result = result + orig.charAt(i);
    }
    return result;
  }
%>

<SimpleResult>
  <loc:address
    name="HQ"
    type="oracle.panama.model.Location"
    businessName="HQ"
```

```
    firstLine="500 Oracle Parkway"
    city="Redwood City"
    state="CA"
    postalCode="94065"
    country="US"/>

  <loc:businesses
    name="starbucks"
    type="java.util.Collection"
    businessName="Starbucks"
    centerID="HQ"
    nearestN="10"/>
  <loc:map name="starbucksMap" type="oracle.panama.spatial.jsptags.beans.Map"
     xres="800" yres="600" points="starbucks">
  </loc:map>

  <SimpleImage src="<%= transformString(starbucksMap.toString()) %>"/>

  <SimpleText>
  <loc:iterateBusinesses name="singleStarbucks" type="oracle.panama.model.Point"
     collection="starbucks">
    <SimpleTextItem> <%= singleStarbucks %> </SimpleTextItem>
  </loc:iterateBusinesses>
  </SimpleText>
</SimpleResult>
```

例 14-7 では、指定したエリア内のカテゴリ別ビジネス・ディレクトリ（YP）情報、特に、California 州 San Francisco にあるすべての自動車ディーラー（新車）を含むマップが表示されます。

例 14-7 JSP タグを使用したカテゴリ別のビジネス・ディレクトリ（YP）

```
<%@ taglib uri="LocationTags" prefix="loc" %>

<%!
  public String transformString(String orig)
  {
    String result = "";
    for (int i=0;i<orig.length();i++)
    {
      if (orig.charAt(i) == '&')      result = result + "&amp;";
      else if (orig.charAt(i) == '<') result = result + "&lt;";
      else if (orig.charAt(i) == '>') result = result + "&gt;";
      else                            result = result + orig.charAt(i);
    }
    return result;
```

```
  }
%>

<SimpleResult>
  <loc:category name="automotive"       type="oracle.panama.spatial.yp.YPCategory"
categoryName="Automotive">
  </loc:category>

  <loc:category name="automotiveDealers"
     type="oracle.panama.spatial.yp.YPCategory" categoryName="Dealers"
     parentCategory="automotive">
  </loc:category>

  <loc:category name="newAutomotiveDealers"
     type="oracle.panama.spatial.yp.YPCategory" categoryName="New"
     parentCategory="automotiveDealers">
  </loc:category>

  <loc:businesses name="dealers" type="java.util.Collection"
     categoryID="newAutomotiveDealers" country="US" state="CA"
     city="San Francisco"/>

  <loc:map name="dealerMap" type="oracle.panama.spatial.jsptags.beans.Map"
     xres="800" yres="600" points="dealers">
  </loc:map>

  <SimpleImage src="<%= transformString(dealerMap.toString()) %>"/>

  <SimpleText>
  <loc:iterateBusinesses name="dealer" type="oracle.panama.model.Point"
     collection="dealers">
    <SimpleTextItem>
      <%= transformString(dealer.toString()) %>
    </SimpleTextItem>
  </loc:iterateBusinesses>
  </SimpleText>
</SimpleResult>
```

addMembers

addMembers タグでは、モバイル・コミュニティに 1 人以上のメンバーを追加します。モバイル・コミュニティの説明は、14-118 ページの「モバイル・コミュニティ」を参照してください。

表 14-4 に、addMembers タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-3 addMembers タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：add_members	はい
type	オブジェクトのタイプ。Boolean にする必要があります（操作に成功すると TRUE、操作に成功しないと FALSE です）。	はい
userName	操作をリクエストする Oracle Application Server Wireless ユーザーの名前。デフォルトは現行のユーザーです。	いいえ
communityID	メンバーを追加するコミュニティに関連付けられた変数の名前。例：comm_private	はい
communityMembers	コミュニティに追加される Oracle Application Server Wireless ユーザーがスペースで区切られたリスト。	はい

次の例では、ユーザー Mike のリクエストによって、ユーザー Song を変数 comm_private に関連付けられたモバイル・コミュニティに追加しています。また、このコミュニティのメンバーの java.util.Enumeration オブジェクトを作成し、表示しています。

```
<loc:addMembers
      name="add_members"
      type="Boolean"
      userName="Mike"
      communityID="comm_private"
      communityMembers="Song" />
<loc:listAllMembers
      name="list_all_mem1"
      type="java.util.Enumeration"
      communityID="comm_private" />
<%= list_all_mem1.toString() %>
```

address

address タグでは、ジオコーディングするか、マップに配置するか、ルートの始点住所または終点住所として使用するか、ビジネス・ディレクトリ問合せの中心として使用する住所を指定します。

表 14-4 に、address タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-4 address タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：office	はい
type	オブジェクトのタイプ。oracle.panama.model.Location にする必要があります。	はい
businessName	指定した住所にあるビジネスまたは他のエンティティの説明的な名前。例：My office	いいえ
firstLine	番地。	いいえ
city	都市名。	いいえ
state	2 文字の州コード（US）または郡コード（カナダ）。	いいえ
postalCode	郵便番号。	いいえ
country	国名。	いいえ
provider	優先順位がある場合は、リクエストに対して最初に選択されるプロバイダの名前。	いいえ

次の例では、2 つの住所（ある人のオフィスと自宅）間にルート myRoute を作成して、ルートのマップとそれに続く横罫線を表示し、（iterateManeuvers タグと getMap および getNarrative ファンクション・コールを使用して）各運転経路とそれに続く横罫線を表しています。また、各運転経路の説明はリンクであり、ユーザーがクリックするとその経路マップを表示できます。

```
<loc:route name="myRoute" type="oracle.panama.spatial.jsptags.beans.Route"
xres="800" yres="600">
   <loc:address
        name="office"
        type="oracle.panama.model.Location"
        businessName="My office"
        firstLine="1 Oracle Dr"
        city="Nashua"
        state="NH"
        postalCode="03062"
        country="US"/>
```

```
   <loc:address
        name="home"
        type="oracle.panama.model.Location"
        businessName="My home"
        firstLine="2 Royal Crest Dr"
        city="Nashua"
        state="NH"
        postalCode="03060"
        country="US"/>
</loc:route>

<img src="<%= myRoute.getMap() %>">
<HR>

<loc:iterateManeuvers name="aManeuver"
type="oracle.panama.spatial.jsptags.beans.Maneuver" routeID="myRoute">
      <a href="<%= aManeuver.getMap() %>">
        <%= aManeuver.getNarrative() %>
      </a>
      <HR>
</loc:iterateManeuvers>
```

businesses

businesses タグでは、1 つ以上の属性を共有するビジネスの（oracle.panama.spatial.yp.YPBusiness オブジェクトの）コレクションを作成します。

表 14-5 に、businesses タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-5　businesses タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例： mikes_hardware_stores	はい
type	オブジェクトのタイプ。java.util.Collection にする必要があります。	はい
businessName	指定した住所にあるビジネスまたは他のエンティティの説明的な名前。例：Mike's Hardware	いいえ
categoryID	ビジネス・サービス・カテゴリの変数名。例：Automotive	いいえ
keyword	name または categoryID 内の検索用文字列。例：French	いいえ
city	都市名。	いいえ
state	2 文字の州コード（US）または郡コード（カナダ）。	いいえ

表 14-5（続き）businesses タグのパラメータ（続き）

パラメータ名	説明	必須
postalCode	郵便番号。	いいえ
country	国名。	いいえ
centerID	検索を開始する中心ポイントとして使用するポイントの変数名（住所の場合など）。centerID を指定した場合は、radius または nearestN も指定する必要があります。	いいえ
radius	検索対象となる円の半径の長さ（m 単位）。radius を指定した場合は、centerID も指定する必要があります。	いいえ
nearestN	問合せ要件を満たす最も近接している結果の最大数（たとえば、ホテルまたはユーザーの現在の位置に最も近接している 3 軒の銀行の検索など）。nearestN を指定した場合は、centerID も指定する必要があります。	いいえ
provider	優先順位がある場合は、リクエストに対して最初に選択されるプロバイダの名前。	いいえ

次の businesses タグの例では、アメリカのカリフォルニア州内のビジネス Borders をすべて指定します。businesses タグを map タグで囲むと、各 Borders 書店を含むラベル付きのマップが作成されます。

```
<loc:map name="map1" type="oracle.panama.spatial.jsptags.beans.Map"
       xres="1000" yres="500">
   <loc:businesses name="bord" type="java.util.Collection" businessName="Borders"
        country="US" state="CA"/>
</loc:map>
```

category

category タグでは、ビジネス・カテゴリ（oracle.panama.spatial.yp.YPCategory オブジェクト）を作成します。

表 14-6 に、category タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-6 category タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：cat_dealers	はい
type	オブジェクトのタイプ。oracle.panama.spatial.yp.YPCategory にする必要があります。	はい

表 14-6（続き）category タグのパラメータ（続き）

パラメータ名	説明	必須
parentCategory	親カテゴリの指定を含むオブジェクトの名前（親オブジェクトは、あらかじめ category タグを使用して作成します）。このパラメータを指定しない場合は、ルートとみなされます。	いいえ
categoryName	カテゴリ名。例：Dealers。	はい

次の例では 2 つの category タグを使用しています。最初の category タグでは、カテゴリ Automotive を指定するオブジェクト cat_auto を作成します。2 番目の category タグでは、カテゴリ Dealers を親カテゴリ cat_auto（Automotive）の子として指定するオブジェクト cat_dealers を作成します。

```
<loc:category name="cat_auto" type="oracle.panama.spatial.yp.YPCategory"
   categoryName="Automotive" />
<loc:category name="cat_dealers" type="oracle.panama.spatial.yp.YPCategory"
   parentCategory="cat_auto" categoryName="Dealers" />
```

createPrivateCommunity

createPrivateCommunity タグでは、プライベート・モバイル・コミュニティを作成します。コミュニティのタイプを含むモバイル・コミュニティの説明は、14-118 ページの「モバイル・コミュニティ」を参照してください。

表 14-7 に、createPrivateCommunity タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-7　createPrivateCommunity タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：comm_private	はい
type	オブジェクトのタイプ。oracle.panama.model.Community にする必要があります。	はい
userName	コミュニティの所有者になる Oracle Application Server Wireless ユーザーの名前。デフォルトは現行のユーザーです。	いいえ
communityName	コミュニティの説明的な名前。例：My Private Community	はい
communityMembers	コミュニティに追加される Oracle Application Server Wireless ユーザーがスペースで区切られたリスト（ユーザーが追加されていない場合）。	いいえ
returnNullIfExists	TRUE（デフォルト）では、そのコミュニティがすでに存在する場合、NULL 値が戻されます。FALSE では、そのコミュニティがすでに存在する場合、既存のコミュニティが戻されます。	いいえ

次の例では、ユーザー Mike が所有するプライベート・コミュニティを作成し、2 人のユーザー（Mike および Jing）を追加しています。コミュニティがすでに存在する場合、コミュニティは作成されません。また、コミュニティに関する情報を表示しています。

```
<loc:createPrivateCommunity
      name="comm_private"
      type="oracle.panama.model.Community"
      userName="Mike"
      communityName="My Private Community"
      communityMembers="Mike Jing"
      returnNullIfExists="FALSE" />
<%= comm_private.toString() %>
```

createSharedCommunity

createSharedCommunity タグでは、共有モバイル・コミュニティを作成します。コミュニティのタイプを含むモバイル・コミュニティの説明は、14-118 ページの「モバイル・コミュニティ」を参照してください。

表 14-8 に、createSharedCommunity タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-8　createSharedCommunity タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：comm_shared	はい
type	オブジェクトのタイプ。oracle.panama.model.Community にする必要があります。	はい
userName	コミュニティの所有者になる OracleAS Wireless ユーザーの名前。デフォルトは現行のユーザーです。	いいえ
communityName	コミュニティの説明的な名前。例：My Shared Community	はい
communityMembers	コミュニティに追加される Oracle Application Server Wireless ユーザーがスペースで区切られたリスト（追加されていない場合）。	いいえ
returnNullIfExists	TRUE（デフォルト）では、そのコミュニティがすでに存在する場合、NULL 値が戻されます。FALSE では、そのコミュニティがすでに存在する場合、既存のコミュニティが戻されます。	いいえ

次の例では、ユーザー Mike が所有する共有コミュニティを作成し、2 人のユーザー（Mike および Jing）を追加しています。コミュニティがすでに存在する場合、コミュニティは作成されません。また、コミュニティに関する情報を表示しています。

```
<loc:createSharedCommunity
      name="comm_shared"
      type="oracle.panama.model.Community"
      userName="Mike"
      communityName="My Shared Community"
      communityMembers="Mike Jing"
      returnNullIfExists="FALSE" />
<%= comm_private.toString() %>
```

createSystemCommunity

createSystemCommunity タグでは、システム・モバイル・コミュニティを作成します。コミュニティのタイプを含むモバイル・コミュニティの説明は、14-118 ページの「モバイル・コミュニティ」を参照してください。

表 14-9 に、createSystemCommunity タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-9 createSystemCommunity タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：comm_system	はい
type	オブジェクトのタイプ。oracle.panama.model.Community にする必要があります。	はい
userName	コミュニティの所有者になる Oracle Application Server Wireless ユーザーの名前。デフォルトは現行のユーザーです。	いいえ
communityName	コミュニティの説明的な名前。例：My System Community	はい
communityMembers	コミュニティに追加される Oracle Application Server Wireless ユーザーがスペースで区切られたリスト（追加されていない場合）。	いいえ
returnNullIfExists	TRUE（デフォルト）では、そのコミュニティがすでに存在する場合、NULL 値が戻されます。FALSE では、そのコミュニティがすでに存在する場合、既存のコミュニティが戻されます。	いいえ

次の例では、ユーザー Mike が所有するシステム・コミュニティを作成し、2 人のユーザー（Mike および Jing）を追加しています。コミュニティがすでに存在する場合、コミュニティは作成されません。また、コミュニティに関する情報を表示しています。

```
<loc:createSystemCommunity
      name="comm_system"
      type="oracle.panama.model.Community"
      userName="Mike"
      communityName="My System Community"
      communityMembers="Mike Jing"
      returnNullIfExists="FALSE" />
<%= comm_private.toString() %>
```

defaultLocationMark

defaultLocationbMark タグでは、指定したユーザーのデフォルト・ロケーション・マークを表すオブジェクトを作成します。このタグを使用すると、ユーザーのデフォルト・ロケーション・マークを検索できます。

表 14-10 に、defaultLocationbMark タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-10　defaultLocationMark タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：user_loc	はい
type	オブジェクトのタイプ。oracle.panama.model.LocationMark にする必要があります。	はい
userName	デフォルト・ロケーション・マークを検索する対象の Oracle Application Server Wireless ユーザーの名前。デフォルトは現行のユーザーです。	いいえ

次の例では、ユーザー Mike のデフォルト・ロケーション・マークを表すオブジェクトを作成し、そのオブジェクトに関する情報を表示しています。

```
<loc:defaultLocationMark
      name="user_mark"
      type="oracle.panama.model.LocationMark"
      userName="Mike"  />
<%= user_mark.toString() %>
```

deleteCommunity

deleteCommunity タグでは、プライベート、共有またはシステム・モバイル・コミュニティを削除します。コミュニティのタイプを含むモバイル・コミュニティの説明は、14-118 ページの「モバイル・コミュニティ」を参照してください。

表 14-11 に、deleteCommunity タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-11 deleteCommunity タグのパラメ ータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：delete_comm1.	はい
type	オブジェクトのタイプ。Boolean にする必要があります（操作に成功すると TRUE、操作に成功しないと FALSE です）。	はい
userName	操作をリクエストする Oracle Application Server Wireless ユーザーの名前。デフォルトは現行のユーザーです。	いいえ
communityName	コミュニティの説明的な名前（変数名ではない）。例：My Private Community	はい

次の例では、ユーザー Mike のリクエストによって、コミュニティ My Private Community を削除し、操作の結果（TRUE または FALSE）を表示しています。

```
<loc:deleteCommunity
      name="delete_comm1"
      type="Boolean"
      userName="Mike"
      communityName="My Private Community"  />
<%= delete_comm1.toString() %>
```

drivingDistance

drivingDistance タグは、プロバイダによって決定される、ルートまたは運転経路の走行距離を表します。

表 14-12 に、drivingDistance タグのパラメータを示します。オプションとして route および maneuver が示されますが、これらのパラメータの 1 つを指定する必要があります（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-12　drivingDistance タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：drive_dist	はい
type	オブジェクトのタイプ。String にする必要があります。	はい
route	ルートの名前。例：myRoute	いいえ
maneuver	経路の名前。	いいえ

次の例では、2 つの住所（ある人のオフィスと自宅）間にルート myRoute を作成し、そのルートの走行距離を表示しています。

```
<loc:route name="myRoute" type="oracle.panama.spatial.jsptags.beans.Route"
xres="800" yres="600">
   <loc:address
        name="office"
        type="oracle.panama.model.Location"
        businessName="My office"
        firstLine="1 Oracle Dr"
        city="Nashua"
        state="NH"
        postalCode="03062"
        country="US"/>
   <loc:address
        name="home"
        type="oracle.panama.model.Location"
        businessName="My home"
        firstLine="2 Royal Crest Dr"
        city="Nashua"
        state="NH"
        postalCode="03060"
        country="US"/>
</loc:route>
<loc:drivingDistance name="drive_dist" type="String" route="myRoute" />
<%= drive_dist.toString() %>
```

drivingTime

drivingTime タグでは、ルートの走行時間の見積りを含むオブジェクトを作成します。

表 14-13 に、drivingTime タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-13 drivingTime タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：drive_time	はい
type	オブジェクトのタイプ。String にする必要があります。	はい
route	ルートの名前。例：myRoute	はい

次の例では、2 つの住所（ある人のオフィスと自宅）間にルート myRoute を作成し、そのルートの走行時間を表示しています。

```
<loc:route name="myRoute" type="oracle.panama.spatial.jsptags.beans.Route"
xres="800" yres="600">
   <loc:address
        name="office"
        type="oracle.panama.model.Location"
        businessName="My office"
        firstLine="1 Oracle Dr"
        city="Nashua"
        state="NH"
        postalCode="03062"
        country="US"/>
   <loc:address
        name="home"
        type="oracle.panama.model.Location"
        businessName="My home"
        firstLine="2 Royal Crest Dr"
        city="Nashua"
        state="NH"
        postalCode="03060"
        country="US"/>
</loc:route>
<loc:drivingTime name="drive_time" type="String" route="myRoute" />
<%= drive_time.toString() %>
```

geocode

geocode タグでは、ジオコーディングするか、マップに配置するか、ルートの始点住所または終点住所として使用するか、ビジネス・ディレクトリ問合せの中心として使用する住所を指定します。

表 14-14 に、geocode タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-14　geocode タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：hardware1	はい
type	オブジェクトのタイプ。oracle.panama.model.Location にする必要があります。	はい
businessName	指定した住所にあるビジネスまたは他のエンティティの説明的な名前。例：Mike's Hardware	いいえ
houseNumber	住所の番地番号。	いいえ
streetName	番地。	はい
secondLine	番地の 2 行目。	いいえ
intersection	houseNumber が指定されていない場合は、交差する番地。	いいえ
city	都市名。	はい
state	2 文字の州コード（US）または郡コード（カナダ）。	はい
postalCode	郵便番号（主要な部分）。例：01742	はい
postalCodeExt	4 桁の追加番号など、郵便番号の拡張部分。	いいえ
country	国名または国コード。	はい
makeCorrections	ジオコーディング・プロバイダで住所のつづりの間違いを訂正する場合は TRUE、ジオコーディング・プロバイダで住所のつづりの間違いを訂正しない場合は FALSE。	はい
provider	優先順位がある場合は、リクエストに対して最初に選択されるサービス・プロバイダの名前。	いいえ

次の geocode タグの例では、ジオコーディングする（店舗 Mike's Hardware の）住所を指定しています。

```
<loc:geocode
  name = "hardware1"
  type = "oracle.panama.model.Location"
  businessName = "Mike's Hardware"
```

```
  houseNumber = "22"
  streetName = "Monument Sq"
  city = "Concord"
  state = "MA"
  postalCode = "01742"
  country = "US"
  makeCorrections = "TRUE"  />
```

geometry

geometry タグでは、タイプ oracle.panama.model.Point のポイントの java.util.List オブジェクトを作成します。

表 14-15 に、geometry タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。points、route または maneuver のパラメータを指定する必要があります。

表 14-15　geometry タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：my_geom	はい
type	オブジェクトのタイプ。java.util.List にする必要があります。	はい
points	ジオメトリを構成するポイントの変数の名前。	いいえ
route	ジオメトリを構成するルートの変数の名前。このパラメータを指定する場合、ルート・オブジェクトを作成した route タグには、requestGeom="TRUE" パラメータを指定しておく必要があります。	いいえ
maneuver	ジオメトリを構成する経路の変数の名前。	いいえ

次の例では、あるユーザーのオフィスの住所と自宅の住所の間にルートを作成し、geometry タグを使用してルート沿いのポイントのフォーマットされていないリストを作成しています。

```
<loc:route name="myRoute" type="oracle.panama.spatial.jsptags.beans.Route"
xres="800" yres="600" requestGeom="true">
   <loc:address
        name="office"
        type="oracle.panama.model.Location"
        businessName="My office"
        firstLine="1 Oracle Dr"
        city="Nashua"
        state="NH"
        postalCode="03062"
        country="US"/>
   <loc:address
```

```
        name="home"
        type="oracle.panama.model.Location"
        businessName="My home"
        firstLine="2 Royal Crest Dr"
        city="Nashua"
        state="NH"
        postalCode="03060"
        country="US"/>
</loc:route>

<loc:geometry name="my_geom" type="java.util.List" route="myRoute" />
<%= my_geom.toString() %>
```

getCommunity

getCommunity タグでは、プライベート、共有またはシステム・モバイル・コミュニティの指定された名前に関連付けられたオブジェクトを戻します。コミュニティのタイプを含むモバイル・コミュニティの説明は、14-118 ページの「モバイル・コミュニティ」を参照してください。

表 14-16 に、getCommunity タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-16　getCommunity タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：get_comm.	はい
type	オブジェクトのタイプ。oracle.panama.model.Community にする必要があります。	はい
userName	操作をリクエストする Oracle Application Server Wireless ユーザーの名前。デフォルトは現行のユーザーです。	いいえ
communityName	コミュニティの説明的な名前（変数名ではない）。例：My Private Community	はい

次の例では、ユーザー Mike のリクエストによって、コミュニティ My Private Community を戻し、そのコミュニティに関する情報を表示しています。

```
<loc:getCommunity
      name="get_comm"
      type="oracle.panama.model.Community"
      userName="Mike"
      communityName="My Private Community"  />
<%= get_comm.toString() %>
```

iterateBusinesses

iterateBusinesses タグは、businesses タグで戻されるコレクション内の個々のビジネスを表します。

表 14-17 に、iterateBusinesses タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-17 iterateBusinesses タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：singleStarbucks	はい
type	オブジェクトのタイプ。oracle.panama.spatial.yp.YPBusiness にする必要があります。	はい
collection	戻されるコレクションの名前。例：starbucks	はい
provider	優先順位がある場合は、リクエストに対して最初に選択されるプロバイダの名前。	いいえ

次の例では、Oracle 本社に最も近接している 10 軒のスターバックスを示すマップを作成し、ロケーションごとにそれぞれの情報とそれに続く横罫線を表示しています。情報の各行はリンクであり、ユーザーがクリックしてそのロケーションと周辺地域の詳細なマップを表示できます。

```
<loc:address
      name="HQ"
      type="oracle.panama.model.Location"
      businessName="HQ"
      firstLine="500 Oracle Parkway"
      city="Redwood City"
      state="CA"
      postalCode="94065"
      country="US"/>
<loc:map name="starbucksMap" type="oracle.panama.spatial.jsptags.beans.Map"
xres="800" yres="600">
    <loc:businesses
        name="starbucks"
        type="java.util.Collection"
        businessName="Starbucks"
        centerID="HQ"
        nearestN="10"/>
</loc:map>

<img src="<%= starbucksMap %>">
<HR>
```

```
<loc:iterateBusinesses name="singleStarbucks"
type="oracle.panama.spatial.yp.YPBusiness" collection="starbucks">
  <loc:map name="singleStarbucksMap" type="oracle.panama.spatial.jsptags.beans.Map"
xres="800" yres="600" points="singleStarbucks"/>
  <a href="<%= singleStarbucksMap %>">
    <%= singleStarbucks %>
  </a>
  <HR>
</loc:iterateBusinesses>
```

iterateBusinessesInCity

iterateBusinessesInCity タグは、指定した都市内の個々のビジネスを表します。

表 14-18 に、iterateBusinessesInCity タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-18　iterateBusinessesInCity タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：a_business_city	はい
type	オブジェクトのタイプ。oracle.panama.spatial.yp.YPBusiness にする必要があります。	はい
businessName	ビジネスの説明的な名前。例：Starbucks	いいえ
categoryID	カテゴリに関連付けられた変数の名前（カテゴリが問合せに含まれる場合）。	いいえ
keyword	キーワード（キーワードが問合せに含まれる場合）。	いいえ
city	都市名。	はい
state	2 文字の州コード（US）または郡コード（カナダ）。	はい
country	国名。	はい
provider	優先順位がある場合は、リクエストに対して最初に選択されるプロバイダの名前。	いいえ

次の例では、California 州 San Francisco にあるすべての Borders 書店を個別に表しています。ロケーションごとに、そのロケーションに関する情報とそれに続く横罫線が表示されます。

```
<loc:iterateBusinessesInCity name="a_business_city"
   type="oracle.panama.spatial.yp.YPBusiness"
   city="San Francisco" state="CA" country="US" businessName="Borders">
```

```
   <%= a_business_city.toString() %>
   <HR>
</loc:iterateBusinessesInCity>
```

iterateBusinessesInCorridor

iterateBusinessesInCorridor タグは、コリドー内の個々のビジネスを表します。コリドーとは、ルートの作成時に交差点や出口などを表すポイントのコレクションです。

表 14-19 に、iterateBusinessesInCorridor タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaScript Pages（JSP）ファイルの作成」を参照）。

表 14-19　iterateBusinessesInCorridor タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：hardware_1	はい
type	オブジェクトのタイプ。oracle.panama.spatial.yp.YPBusiness にする必要があります。	はい
businessName	ビジネスの説明的な名前。例：Starbucks	いいえ
categoryID	カテゴリに関連付けられた変数の名前（カテゴリが問合せに含まれる場合）。	いいえ
keyword	キーワード（キーワードが問合せに含まれる場合）。	いいえ
corridorID	コリドーに関連付けられた変数の名前。	はい
radiusInMeters	コリドー内の各ポイントを中心とした m 単位の半径。	はい
provider	優先順位がある場合は、リクエストに対して最初に選択されるプロバイダの名前。	いいえ

次の例では、オフィスと他の場所の間にルートを作成し、そのルートに関連付けられたコリドー内のすべてのポイントから 3000m 以内のスターバックスを個別に表しています。この例では、ロケーションごとに、そのロケーションに関する情報とそれに続く横罫線が表示されます。

```
<loc:route name="myRoute" type="oracle.panama.spatial.jsptags.beans.Route"
xres="800" yres="600" requestGeom="true">
    <loc:address
        name="office"
        type="oracle.panama.model.Location"
        businessName="Some office"
        firstLine="500 Oracle Parkway"
        city="Redwood City"
        state="CA"
        postalCode="94065"
```

```
        country="US"/>
    <loc:address
        name="ucsb"
        type="oracle.panama.model.Location"
        businessName="UCSB"
        firstLine="6750 El Colegio Rd"
        city="Goleta"
        state="CA"
        postalCode="93117"
        country="US"/>
</loc:route>

<loc:geometry name="myRouteGeom" type="java.util.List" route="myRoute"/>

<loc:iterateBusinessesInCorridor name="a_business_corridor"
   type="oracle.panama.spatial.yp.YPBusiness"
   businessName="Starbucks" corridorID="myRouteGeom" radiusInMeters="3000">
   <%= a_business_corridor.toString() %>
   <HR>
</loc:iterateBusinessesInCorridor>
```

iterateBusinessesInPostalCode

iterateBusinessesInPostalCode タグは、指定した郵便番号に該当する個々のビジネスを表します。

表 14-20 に、iterateBusinessesInPostalCode タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-20　iterateBusinessesInPostalCode タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：a_business_pcode	はい
type	オブジェクトのタイプ。oracle.panama.spatial.yp.YPBusiness にする必要があります。	はい
businessName	ビジネスの説明的な名前。例：Starbucks	いいえ
categoryID	カテゴリに関連付けられた変数の名前（カテゴリが問合せに含まれる場合）。	いいえ
keyword	キーワード（キーワードが問合せに含まれる場合）。	いいえ
postalCode	郵便番号。	はい

表 14-20（続き）iterateBusinessesInPostalCode タグのパラメータ（続き）

パラメータ名	説明	必須
country	国名。	はい
provider	優先順位がある場合は、リクエストに対して最初に選択されるプロバイダの名前。	いいえ

次の例では、アメリカ合衆国内の郵便番号 93117 に該当するすべてのスターバックスを個別に表しています。ロケーションごとに、そのロケーションに関する情報とそれに続く横罫線が表示されます。

```
<loc:iterateBusinessesInPostalCode name="a_business_pcode"
   type="oracle.panama.spatial.yp.YPBusiness"
   postalCode="93117" country="US" businessName="Starbucks">
   <%= a_business_pcode.toString() %>
   <HR>
</loc:iterateBusinessesInPostalCode>
```

iterateBusinessesInRadius

`iterateBusinessesInRadius` タグは、ポイントを中心に指定した半径（m 単位）を持つ円形の地域内にある個々のビジネスを表します。

表 14-21 に、`iterateBusinessesInRadius` タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-21　iterateBusinessesInRadius タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：`a_business_radius`	はい
type	オブジェクトのタイプ。`oracle.panama.spatial.yp.YPBusiness` にする必要があります。	はい
businessName	ビジネスの説明的な名前。例：`Starbucks`	いいえ
categoryID	カテゴリに関連付けられた変数の名前（カテゴリが問合せに含まれる場合）。	いいえ
keyword	キーワード（キーワードが問合せに含まれる場合）。	いいえ
centerID	問合せの中心ポイントに関連付けられた変数の名前。	はい
radiusInMeters	`centerID` を中心とした円の半径を表す数値（m 単位）。	はい
provider	優先順位がある場合は、リクエストに対して最初に選択されるプロバイダの名前。	いいえ

次の例では、Oracle 本社の住所に関連付けられたポイントから 5000m（5Km）以内のすべてのスターバックスを個別に表しています。ロケーションごとに、そのロケーションに関する情報とそれに続く横罫線が表示されます。

```
<loc:address
      name="HQ"
      type="oracle.panama.model.Location"
      businessName="HQ"
      firstLine="500 Oracle Parkway"
      city="Redwood City"
      state="CA"
      postalCode="94065"
      country="US"/>
<loc:iterateBusinessesInRadius name="a_business_radius"
   type="oracle.panama.spatial.yp.YPCategory"
   businessName="Starbucks" centerID="HQ"  radiusInMeters="5000">
   <%= a_business_radius.toString() %>
   <HR>
</loc:iterateBusinessesInRadius>
```

iterateBusinessesInState

`iterateBusinessesInState` タグは、指定した州内の個々のビジネスを表します。

表 14-22 に、`iterateBusinessesInState` タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-22　iterateBusinessesInState タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：`a_business_state`	はい
type	オブジェクトのタイプ。`oracle.panama.spatial.yp.YPBusiness` にする必要があります。	はい
businessName	ビジネスの説明的な名前。例：`Starbucks`	いいえ
categoryID	カテゴリに関連付けられた変数の名前（カテゴリが問合せに含まれる場合）。	いいえ
keyword	キーワード（キーワードが問合せに含まれる場合）。	いいえ
state	2 文字の州コード（US）または郡コード（カナダ）。	はい
country	国名。	はい
provider	優先順位がある場合は、リクエストに対して最初に選択されるプロバイダの名前。	いいえ

次の例では、California 州にあるすべてのスターバックスを個別に表しています。ロケーションごとに、そのロケーションに関する情報とそれに続く横罫線が表示されます。

```
<loc:iterateBusinessesInState name="a_business_state"
   type="oracle.panama.spatial.yp.YPBusiness"
   state="CA" country="US" businessName="Starbucks">
   <%= a_business_state.toString() %>
   <HR>
</loc:iterateBusinessesInState>
```

iterateBusinessesNearestTo

iterateBusinessesNearestTo タグは、ポイントを中心に指定した半径（m 単位）を持つ円形の地域内にある個々のビジネスを表します。

表 14-23 に、iterateBusinessesNearestTo タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-23　iterateBusinessesNearestTo タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：a_business_nearest	はい
type	オブジェクトのタイプ。oracle.panama.spatial.yp.YPBusiness にする必要があります。	はい
businessName	ビジネスの説明的な名前。例：Starbucks	いいえ
categoryID	カテゴリに関連付けられた変数の名前（カテゴリが問合せに含まれる場合）。	いいえ
keyword	キーワード（キーワードが問合せに含まれる場合）。	いいえ
centerID	問合せの中心ポイントに関連付けられた変数の名前。	はい
n	centerID から最も近接しているビジネスの数。	はい
provider	優先順位がある場合は、リクエストに対して最初に選択されるプロバイダの名前。	いいえ

次の例では、Oracle 本社の住所に関連付けられたポイントに最も近接している 10 軒のスターバックスを個別に表しています。ロケーションごとに、そのロケーションに関する情報とそれに続く横罫線が表示されます。

```
<loc:address
      name="HQ"
      type="oracle.panama.model.Location"
      businessName="HQ"
```

```
      firstLine="500 Oracle Parkway"
      city="Redwood City"
      state="CA"
      postalCode="94065"
      country="US"/>
<loc:iterateBusinessesNearestTo name="a_business_nearest"
   type="oracle.panama.spatial.yp.YPBusiness"
   businessName="Starbucks" centerID="HQ"  n="10">
   <%= a_business_nearest.toString() %>
   <HR>
</loc:iterateBusinessesNearestTo>
```

iterateByDistance

iterateByDistance タグは、コレクション内の個々のポイントを、指定したポイントからの距離順にソートして表します。距離は、地球の曲率に従った直線で測定されます。

表 14-24 に、iterateByDistance タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-24　iterateByDistance タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：iter_dist	はい
type	オブジェクトのタイプ。oracle.panama.model.Point にする必要があります。	はい
collection	距離順にソートされるポイントのコレクションに関連付けられた変数の名前。	はい
centerID	距離を計算する中心ポイントとして使用するポイントの変数名（住所の場合など）。	はい

次の例では、Oracle 本社に最も近接している 10 軒のスターバックスのコレクションを作成しています。iterateByDistance タグを使用して、個々のロケーションを本社からの距離順にソートして表し、各ロケーションに関する情報とそれに続く横罫線を表示しています。

```
<loc:address
      name="HQ"
      type="oracle.panama.model.Location"
      businessName="HQ"
      firstLine="500 Oracle Parkway"
      city="Redwood City"
      state="CA"
      postalCode="94065"
      country="US"/>
<loc:businesses
```

```
      name="starbucks"
      type="java.util.Collection"
      businessName="Starbucks"
      centerID="HQ"
      nearestN="10"/>
<loc:iterateByDistance name="iter_dist" type="oracle.panama.model.Point"
       collection="starbucks" centerID="HQ">
   <%= iter_dist.toString() %>
   <HR>
</loc:iterateByDistance>
```

iterateByDrivingDistance

iterateByDrivingDistance タグは、コレクション内の個々のポイントを、ルーティング・プロバイダが決定した指定のポイントからの走行距離順にソートして表します。

> **注意：** 走行距離順のソートは、ルーティング・プロバイダによって実行されます。したがって、このタグは走行距離順のソートをサポートするプロバイダのみと併用できます。

表 14-25 に、iterateByDrivingDistance タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-25 iterateByDrivingDistance タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：iter_drive	はい
type	オブジェクトのタイプ。oracle.panama.model.Point にする必要があります。	はい
collection	走行距離順にソートされるポイントのコレクションに関連付けられた変数の名前。	はい
centerID	走行距離を計算する中心ポイントとして使用するポイントの変数名（住所の場合など）。	はい
provider	優先順位がある場合は、リクエストに対して最初に選択されるプロバイダの名前。	いいえ

次の例では、Oracle 本社に最も近接している 10 軒のスターバックスのコレクションを作成し、iterateByDrivingDistance タグを使用して、ロケーションを本社からの走行距離順にソートし、各ロケーションに関する情報とそれに続く横罫線を表示しています。

```
<loc:address
      name="HQ"
```

```
      type="oracle.panama.model.Location"
      businessName="HQ"
      firstLine="500 Oracle Parkway"
      city="Redwood City"
      state="CA"
      postalCode="94065"
      country="US"/>
<loc:businesses
      name="starbucks"
      type="java.util.Collection"
      businessName="Starbucks"
      centerID="HQ"
      nearestN="10"/>
<loc:iterateByDrivingDistance name="iter_drive" type="oracle.panama.model.Point"
       collection="starbucks" centerID="HQ">
   <%= iter_drive.toString() %>
   <HR>
</loc:iterateByDrivingDistance>
```

iterateByName

iterateByName タグは、コレクション内の個々のポイントをビジネス名順にソートして表します。

表 14-26 に、iterateByName タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-26　iterateByName タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：iter_name	はい
type	オブジェクトのタイプ。oracle.panama.model.Point にする必要があります。	はい
collection	名前順にソートされるポイントのコレクションに関連付けられた変数の名前。	はい

次の例では、以前に作成したコレクション bookstores 内の個々のビジネスを、名前順にソートして表しています。ビジネスごとに、そのロケーションに関する情報とそれに続く横罫線が表示されます。

```
<loc:iterateByName name="iter_name" type="oracle.panama.model.Point"
       collection="bookstores">
   <%= iter_name.toString() %>
   <HR>
</loc:iterateByName>
```

iterateByRegionName

iterateByName タグは、コレクション内の個々のポイントを、リージョン名順にソートして表します。リージョンは最初に国でソートされ、続いて州、都市、郵便番号でソートされます。

表 14-27 に、iterateByRegionName タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-27 iterateByRegionName タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：iter_reg_name	はい
type	オブジェクトのタイプ。oracle.panama.model.Point にする必要があります。	はい
collection	名前順にソートされるポイントのコレクションに関連付けられた変数の名前。	はい

次の例では、以前に作成したコレクション starbucks 内の個々のビジネスを、リージョン名順に（国、州、都市、郵便番号の順に）ソートして表しています。ビジネスごとに、そのロケーションに関する情報とそれに続く横罫線が表示されます。

```
<loc:iterateByRegionName name="iter_reg_name" type="oracle.panama.model.Point"
       collection="starbucks">
   <%= iter_reg_name.toString() %>
   <HR>
</loc:iterateByRegionName>
```

iterateCategoriesMatchingKeyword

iterateCategoriesMatchingKeyword タグでは、指定したキーワード値と一致するカテゴリのコレクションを作成し、各カテゴリを個別に表示します。

表 14-28 に、iterateCategoriesMatchingKeyword タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-28 iterateCategoriesMatchingKeyword タグのパラータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：a_category	はい
type	オブジェクトのタイプ。oracle.panama.spatial.yp.YPCategory にする必要があります。	はい

表 14-28 iterateCategoriesMatchingKeyword タグのパラータ（続き）

パラメータ名	説明	必須
parentCategory	親カテゴリの指定を含むオブジェクトの名前（親オブジェクトは、あらかじめ category タグを使用して作成します）。	いいえ
keyword	親カテゴリ名、または parentCategory が指定されていない場合はすべてのカテゴリ名の中で検索される語句。	はい

次の例では、キーワード restaurant と一致する個々のカテゴリを表しています。カテゴリごとに、完全修飾された名前とそれに続く横罫線が表示されます。

```
<loc:iterateCategoriesMatchingKeyword name="a_category"
   type="oracle.panama.spatial.yp.YPCategory"
   keyword="restaurant">
   <%= a_category.getFullyQualifiedName() %>
   <HR>
</loc:iterateCategoriesMatchingKeyword>
```

iterateChildCategories

iterateChildCategories タグでは、個別に表示される直下のサブカテゴリのコレクションを指定します。

表 14-29 に、iterateChildCategories タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-29 iterateChildCategories タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：hardware_stores	はい
type	オブジェクトのタイプ。oracle.panama.spatial.yp.YPCategory にする必要があります。	はい
parentCategory	親カテゴリの指定を含むオブジェクトの名前（親オブジェクトは、あらかじめ category タグを使用して作成します）。	いいえ

次の例では、変数 restaurant に関連付けられた親カテゴリの、すべての子カテゴリを個別に表しています。子カテゴリごとに、完全修飾された名前とそれに続く横罫線が表示されます。

```
<loc:iterateChildCategories name="a_child_category"
   type="oracle.panama.spatial.yp.YPCategory"
   parentCategory="restaurant">
   <%= a_child_category.getFullyQualifiedName() %>
   <HR>
</loc:iterateChildCategories>
```

iterateGeocodes

iterateGeocodes タグでは、個別に表示されるジオコーディングされた住所のコレクションを戻します。

表 14-30 に、iterateGeocodes タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-30 iterateGeocodes タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：main_bus	はい
type	オブジェクトのタイプ。oracle.panama.model.Location にする必要があります。	はい
businessName	指定した住所にあるビジネスまたは他のエンティティの説明的な名前。例：Mike's Hardware	いいえ
houseNumber	住所の番地番号。	いいえ
streetName	番地。	はい
secondLine	番地の 2 行目。	いいえ
intersection	houseNumber が指定されていない場合は、交差する番地。	いいえ
city	都市名。	はい
state	2 文字の州コード（US）または郡コード（カナダ）。	はい
postalCode	郵便番号（主要な部分）。例：01742	はい
postalCodeExt	4 桁の追加番号など、郵便番号の拡張部分。	いいえ
country	国名または国コード。	はい
provider	優先順位がある場合は、リクエストに対して最初に選択されるサービス・プロバイダの名前。	いいえ

次の iterateGeocodes タグの例は、New Hampshire 州 Nashua で郵便番号 03060 の Daniel Webster Highway にジオコーディングされた各住所を表します。ジオコーディングされた住所ごとに、横罫線および 1 行のテキストが表示され、改行後にプロバイダからの住所情報が表示されます。

```
<loc:iterateGeocodes
  name = "a_business"
  type = "oracle.panama.model.Location"
  streetName = "Daniel Webster Hwy"
  city = "Nashua"
  state = "NH"
  postalCode = "03060"
  country = "US">
      <HR>
      Another business in our community:
      <br>
<%= a_business.toString() %>
</loc:iterateGeocodes>
```

iterateLocationMarks

iterateLocationMarks タグは、Oracle Application Server Wireless ユーザーの個々のロケーション・マークを表します。

表 14-31 に、iterateLocationMarks タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-31　iterateLocationMarks タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：iter_marks	はい
type	オブジェクトのタイプ。oracle.panama.model.LocationMark にする必要があります。	はい
userName	ロケーション・マークを表示する対象の Oracle Application Server Wireless ユーザーの名前。デフォルトは現行のユーザーです。	いいえ

次の例では、変数 iter_marks に関連付けられたオブジェクトとして、ユーザー Mike の各ロケーション・マークを表し、オブジェクトごとに、そのオブジェクトの情報とそれに続く横罫線を表示しています。

```
<loc:iterateLocationMarks name="iter_marks"
       type="oracle.panama.model.LocationMark"
       userName="Mike" >
   <%= iter_marks.toString() %>
   <HR>
</loc:iterateLocationMarks>
```

iterateManeuvers

iterateManeuvers タグでは、運転経路のコレクションを作成し、経路を個別に表示します。

表 14-32 に、iterateManeuvers タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-32　iterateManeuvers タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：eManeuver	はい
type	オブジェクトのタイプ。oracle.panama.spatial.jsptags.beans.Maneuver にする必要があります。	はい
routeID	運転経路を表すルートの名前。	はい

次の例では、2 つの住所（ある人のオフィスと自宅）間にルート myRoute を作成して、ルートのマップとそれに続く横罫線を表示し、（iterateManeuvers タグと getMap および getNarrative ファンクション・コールを使用して）各運転経路とそれに続く横罫線を表しています。また、各運転経路の説明はリンクであり、ユーザーがクリックするとその経路マップを表示できます。

```
<loc:route name="myRoute" type="oracle.panama.spatial.jsptags.beans.Route"
xres="800" yres="600">
   <loc:address
        name="office"
        type="oracle.panama.model.Location"
        businessName="My office"
        firstLine="1 Oracle Dr"
        city="Nashua"
        state="NH"
        postalCode="03062"
        country="US"/>
   <loc:address
        name="home"
        type="oracle.panama.model.Location"
        businessName="My home"
        firstLine="2 Royal Crest Dr"
        city="Nashua"
        state="NH"
        postalCode="03060"
        country="US"/>
</loc:route>

<img src="<%= myRoute.getMap() %>">
```

```
<HR>

<loc:iterateManeuvers name="aManeuver"
type="oracle.panama.spatial.jsptags.beans.Maneuver" routeID="myRoute">
      <a href="<%= aManeuver.getMap() %>">
        <%= aManeuver.getNarrative() %>
      </a>
      <HR>
</loc:iterateManeuvers>
```

iterateReverseGeocodes

iterateReverseGeocodes タグは、個別に表示される逆ジオコーディングされた住所（指定したポイントにプロバイダによって関連付けられた住所）のコレクションを戻します。

表 14-33 に、iterateReverseGeocodes タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-33 iterateReverseGeocodes タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：hardware_1	はい
type	オブジェクトのタイプ。oracle.panama.model.Location にする必要があります。	はい
businessName	指定した住所にあるビジネスまたは他のエンティティの説明的な名前。例：Mike's Hardware	いいえ
firstLine	番地。	いいえ
city	都市名。	いいえ
state	2 文字の州コード（US）または郡コード（カナダ）。	いいえ
postalCode	郵便番号。	いいえ
country	国名。	いいえ
provider	優先順位がある場合は、リクエストに対して最初に選択されるプロバイダの名前。	いいえ

次の iterateReverseGeocodes タグの例は、指定したポイントにプロバイダによって関連付けられ、ジオコーディングされた各住所を表します。ジオコーディングされた住所ごとに、プロバイダからの住所情報とそれに続く横罫線が表示されます。

```
<loc:iterateReverseGeocodes
  name = "iter_rev"
  type = "oracle.panama.model.Location"
  lon = "-71.4424"
  lat = "42.712"
  label = "You Are Here" >
<%= iter_rev.toString() %>
<HR>
</loc:iterateReverseGeocodes>
```

listAllMembers

listAllMembers タグでは、指定したモバイル・コミュニティのすべてのメンバーのフォーマットされていないリストを作成します。

表 14-34 に、listAllMembers タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-34　listAllMembers タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：list_all_mem.	はい
type	オブジェクトのタイプ。java.util.Enumertion にする必要があります。	はい
communityID	メンバーをリストするコミュニティに関連付けられた変数の名前。	はい

次の例では、変数 comm_private に関連付けられたコミュニティのすべてのメンバーのフォーマットされていないリストを作成し、作成される java.util.Enumeration オブジェクトを表示しています。

```
<loc:listAllMembers
      name="list_all_mem"
      type="java.util.Enumeration"
      communityID="comm_private" />
<%= list_all_mem.toString() %>
```

listBusinessesInCity

listBusinessesInCity タグでは、指定した都市内のビジネスのフォーマットされていないリストを作成します。

表 14-35 に、listBusinessesInCity タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-35 listBusinessesInCity タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：all_businesses_city	はい
type	オブジェクトのタイプ。java.util.List にする必要があります。	はい
businessName	ビジネスの説明的な名前。例：Starbucks	いいえ
categoryID	カテゴリに関連付けられた変数の名前（カテゴリが問合せに含まれる場合）。	いいえ
keyword	キーワード（キーワードが問合せに含まれる場合）。	いいえ
city	都市名。	はい
state	2 文字の州コード（US）または郡コード（カナダ）。	はい
country	国名。	はい
provider	優先順位がある場合は、リクエストに対して最初に選択されるプロバイダの名前。	いいえ

次の例では、California 州 San Francisco にあるすべての Borders 書店のフォーマットされていないリストを作成し、そのリストを表示しています。

```
<loc:listBusinessesInCity name="all_businesses_city"
   type="java.util.List"
   city="San Francisco" state="CA" country="US" businessName="Borders">
</loc:listBusinessesInCity>
<%= all_businesses_city.toString() %>
```

listBusinessesInCorridor

listBusinessesInCorridor タグでは、コリドー内のビジネスのフォーマットされていないリストを作成します。コリドーとは、ルートの作成時に交差点や出口などを表すポイントのコレクションです。

表 14-36 に、listBusinessesInCorridor タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-36 listBusinessesInCorridor タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：hardware_1	はい
type	オブジェクトのタイプ。oracle.panama.model.Location にする必要があります。	はい
businessName	指定した住所にあるビジネスまたは他のエンティティの説明的な名前。例：Mike's Hardware	いいえ
categoryID	カテゴリに関連付けられた変数の名前（カテゴリが問合せに含まれる場合）。	いいえ
keyword	キーワード（キーワードが問合せに含まれる場合）。	いいえ
corridorID	コリドーに関連付けられた変数の名前。	はい
radiusInMeters	コリドー内の各ポイントを中心とした m 単位の半径。	はい
provider	優先順位がある場合は、リクエストに対して最初に選択されるプロバイダの名前。	いいえ

次の例では、オフィスと他の場所の間にルートを作成して、そのルートに関連付けられたコリドー内のすべてのポイントから 3000m 以内のスターバックスのフォーマットされていないリストを作成し、そのリストを表示しています。

```
<loc:route name="myRoute" type="oracle.panama.spatial.jsptags.beans.Route"
xres="800" yres="600" requestGeom="true">
    <loc:address
        name="office"
        type="oracle.panama.model.Location"
        businessName="Some office"
        firstLine="500 Oracle Parkway"
        city="Redwood City"
        state="CA"
        postalCode="94065"
        country="US"/>
    <loc:address
        name="ucsb"
        type="oracle.panama.model.Location"
```

```
        businessName="UCSB"
        firstLine="6750 El Colegio Rd"
        city="Goleta"
        state="CA"
        postalCode="93117"
        country="US"/>
</loc:route>

<loc:geometry name="myRouteGeom" type="java.util.List" route="myRoute"/>

<loc:listBusinessesInCorridor name="all_businesses_corridor"
   type="java.util.List"
   businessName="Starbucks" corridorID="myRouteGeom" radiusInMeters="3000">
</loc:listBusinessesInCorridor>
<%= all_businesses_corridor.toString() %>
```

listBusinessesInPostalCode

listBusinessesInPostalCode タグでは、指定した郵便番号に該当するビジネスのフォーマットされていないリストを作成します。

表 14-37 に、listBusinessesInPostalCode タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-37 listBusinessesInPostalCode タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：all_businesses_pcode	はい
type	オブジェクトのタイプ。java.util.List にする必要があります。	はい
businessName	ビジネスの説明的な名前。例：Starbucks	いいえ
categoryID	カテゴリに関連付けられた変数の名前（カテゴリが問合せに含まれる場合）。	いいえ
keyword	キーワード（キーワードが問合せに含まれる場合）。	いいえ
postalCode	郵便番号。	はい
country	国名。	はい
provider	優先順位がある場合は、リクエストに対して最初に選択されるプロバイダの名前。	いいえ

次の例では、アメリカ合衆国内の郵便番号 93117 に該当するすべてのスターバックスのフォーマットされていないリストを作成し、そのリストを表示しています。

```
<loc:listBusinessesInPostalCode name="all_businesses_pcode"
   type="java.util.List"
   postalCode="93117" country="US" businessName="Starbucks">
</loc:listBusinessesInPostalCode>
<%= all_businesses_pcode.toString() %>
```

listBusinessesInRadius

`listBusinessesInRadius` タグでは、ポイントを中心に指定した半径（m 単位）を持つ円形の地域内にあるビジネスのフォーマットされていないリストを作成します。

表 14-38 に、`listBusinessesInRadius` タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-38 listBusinessesInRadius タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：`all_businesses_radius`	はい
type	オブジェクトのタイプ。`java.util.List` にする必要があります。	はい
businessName	ビジネスの説明的な名前。例：`Starbucks`	いいえ
categoryID	カテゴリに関連付けられた変数の名前（カテゴリが問合せに含まれる場合）。	いいえ
keyword	キーワード（キーワードが問合せに含まれる場合）。	いいえ
centerID	問合せの中心ポイントに関連付けられた変数の名前。	はい
radiusInMeters	`centerID` を中心とした円の半径を表す数値（m 単位）。	はい
provider	優先順位がある場合は、リクエストに対して最初に選択されるプロバイダの名前。	いいえ

次の例では、Oracle 本社の住所に関連付けられたポイントから 5000m（5Km）以内のすべてのスターバックスのフォーマットされていないリストを作成し、そのリストを表示しています。

```
<loc:address
      name="HQ"
      type="oracle.panama.model.Location"
      businessName="HQ"
      firstLine="500 Oracle Parkway"
      city="Redwood City"
```

```
      state="CA"
      postalCode="94065"
      country="US"/>
<loc:listBusinessesInRadius name="all_businesses_radius"
   type="java.util.List"
   businessName="Starbucks" centerID="HQ"  radiusInMeters="5000">
</loc:listBusinessesInRadius>
<%= all_businesses_radius.toString() %>
```

listBusinessesInState

listBusinessesInState タグでは、指定した州内のビジネスのフォーマットされていないリストを作成します。

表 14-39 に、listBusinessesInState タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-39 listBusinessesInState タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：all_businesses_state	はい
type	オブジェクトのタイプ。java.util.List にする必要があります。	はい
businessName	ビジネスの説明的な名前。例：Starbucks	いいえ
categoryID	カテゴリに関連付けられた変数の名前（カテゴリが問合せに含まれる場合）。	いいえ
keyword	キーワード（キーワードが問合せに含まれる場合）。	いいえ
state	2 文字の州コード（US）または郡コード（カナダ）。	はい
country	国名。	はい
provider	優先順位がある場合は、リクエストに対して最初に選択されるプロバイダの名前。	いいえ

次の例では、California 州にあるすべてのスターバックスのフォーマットされていないリストを作成し、そのリストを表示しています。

```
<loc:listBusinessesInState name="all_businesses_state"
   type="java.util.List"
   state="CA" country="US" businessName="Borders">
</loc:listBusinessesInState>
<%= all_businesses_state.toString() %>
```

listBusinessesNearestTo

listBusinessesNearestTo タグでは、ポイントを中心に指定した半径（m 単位）を持つ円形の地域内にあるビジネスのフォーマットされていないリストを作成します。

表 14-40 に、listBusinessesNearestTo タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-40 listBusinessesNearestTo タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：all_businesses_nearest	はい
type	オブジェクトのタイプ。java.util.List にする必要があります。	はい
businessName	ビジネスの説明的な名前。例：Starbucks	いいえ
categoryID	カテゴリに関連付けられた変数の名前（カテゴリが問合せに含まれる場合）。	いいえ
keyword	キーワード（キーワードが問合せに含まれる場合）。	いいえ
centerID	問合せの中心ポイントに関連付けられた変数の名前。	はい
n	centerID から最も近接しているビジネスの数。	はい
provider	優先順位がある場合は、リクエストに対して最初に選択されるプロバイダの名前。	いいえ

次の例では、Oracle 本社の住所に関連付けられたポイントに最も近接している 10 軒のスターバックスのフォーマットされていないリストを作成し、そのリストを表示しています。

```
<loc:address
      name="HQ"
      type="oracle.panama.model.Location"
      businessName="HQ"
      firstLine="500 Oracle Parkway"
      city="Redwood City"
      state="CA"
      postalCode="94065"
      country="US"/>
<loc:listBusinessesNearestTo name="all_businesses_nearest"
   type="java.util.List"
   businessName="Starbucks" centerID="HQ"  n="10">
</loc:listBusinessesNearestTo>
<%= all_businesses_nearest.toString() %>
```

listByDistance

listByDistance タグでは、コレクション内のポイントを指定したポイントからの距離順にソートしたフォーマットされていないリストを作成します。距離は、地球の曲率に従った直線で測定されます。

表 14-41 に、listByDistance タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-41 listByDistance タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：list_dist	はい
type	オブジェクトのタイプ。java.util.List にする必要があります。	はい
collection	距離順にソートされるポイントのコレクションに関連付けられた変数の名前。	はい
centerID	距離を計算する中心ポイントとして使用するポイントの変数名（住所の場合など）。	はい

次の例では、Oracle 本社に最も近接している 10 軒のスターバックスのコレクションを作成し、listByDistance タグを使用して、ロケーションを本社からの距離順にソートしたフォーマットされていないリストを作成し、そのリストを表示しています。

```
<loc:address
      name="HQ"
      type="oracle.panama.model.Location"
      businessName="HQ"
      firstLine="500 Oracle Parkway"
      city="Redwood City"
      state="CA"
      postalCode="94065"
      country="US"/>
<loc:businesses
      name="starbucks"
      type="java.util.Collection"
      businessName="Starbucks"
      centerID="HQ"
      nearestN="10"/>
<loc:listByDistance name="list_dist" type="java.util.List"
   collection="starbucks" centerID="HQ">
</loc:listByDistance>
<%= list_dist.toString() %>
```

listByDrivingDistance

listByDrivingDistance タグでは、コレクション内のポイントを、ルーティング・プロバイダが決定した指定のポイントからの走行距離順にソートした、フォーマットされていないリストを作成します。

注意： 走行距離順のソートは、ルーティング・プロバイダによって実行されます。したがって、このタグは走行距離順のソートをサポートするプロバイダのみと併用できます。

表 14-42 に、listByDrivingDistance タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-42 listByDrivingDistance タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：hardware_1	はい
type	オブジェクトのタイプ。java.util.List にする必要があります。	はい
collection	走行距離順にソートされるポイントのコレクションに関連付けられた変数の名前。	はい
centerID	走行距離を計算する中心ポイントとして使用するポイントの変数名（住所の場合など）。	はい
provider	優先順位がある場合は、リクエストに対して最初に選択されるプロバイダの名前。	いいえ

次の例では、Oracle 本社に最も近接している 10 軒のスターバックスのコレクションを作成し、listByDrivingDistance タグを使用して、ロケーションを本社からの走行距離順にソートしたフォーマットされていないリストを作成し、そのリストを表示しています。

```
<loc:address
      name="HQ"
      type="oracle.panama.model.Location"
      businessName="HQ"
      firstLine="500 Oracle Parkway"
      city="Redwood City"
      state="CA"
      postalCode="94065"
      country="US"/>
<loc:businesses
      name="starbucks"
      type="java.util.Collection"
      businessName="Starbucks"
      centerID="HQ"
```

```
      nearestN="10"/>
<loc:listByDrivingDistance name="list_drive" type="java.util.List"
   collection="starbucks" centerID="HQ">
</loc:listByDrivingDistance>
<%= list_drive.toString() %>
```

listByName

listByName タグでは、コレクション内のポイントをビジネス名順にソートしたフォーマットされていないリストを作成します。

表 14-43 に、listByName タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-43 listByName タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：iter_name	はい
type	オブジェクトのタイプ。java.util.List にする必要があります。	はい
collection	名前順にソートされるポイントのコレクションに関連付けられた変数の名前。	はい

次の例では、（以前に作成した）コレクション bookstores 内のビジネスを名前順にソートしたフォーマットされていないリストを作成し、そのリストを表示しています。

```
<loc:listByName name="list_name" type="java.util.List"
   collection="bookstores">
</loc:listByName>
<%= list_name.toString() %>
```

listByRegionName

listByName タグでは、コレクション内のポイントをリージョン名順にソートしたフォーマットされていないリストを作成します。リージョンは最初に国でソートされ、続いて州、都市、郵便番号でソートされます。

表 14-44 に、listByName タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-44 listByRegionName タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：list_reg_name	はい
type	オブジェクトのタイプ。oracle.panama.model.Point にする必要があります。	はい
collection	名前順にソートされるポイントのコレクションに関連付けられた変数の名前。	はい

次の例では、（以前に作成した）コレクション starbucks 内のビジネスをリージョン名順に（国、州、都市、郵便番号の順に）ソートしたフォーマットされていないリストを作成し、そのリストを表示しています。

```
<loc:listByRegionName name="list_reg_name" type="java.util.List"
   collection="starbucks">
</loc:listByRegionName>
<%= list_reg_name.toString() %>
```

listCategoriesMatchingKeyword

listCategoriesMatchingKeyword タグでは、指定したキーワードと一致するビジネス・ディレクトリ・カテゴリのフォーマットされていないリストを作成します。

表 14-45 に、listCategoriesMatchingKeyword タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-45 listCategoriesMatchingKeyword タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：all_categories_key	はい
type	オブジェクトのタイプ。java.util.List にする必要があります。	はい

表 14-45 listCategoriesMatchingKeyword タグのパラメータ（続き）

パラメータ名	説明	必須
parentCategory	親カテゴリの指定を含むオブジェクトの名前（親オブジェクトは、あらかじめ category タグを使用して作成します）。	いいえ
keyword	親カテゴリ名、または parentCategory が指定されていない場合はすべてのカテゴリ名の中で検索される語句。	はい

次の例では、キーワード restaurant に一致するカテゴリのフォーマットされていないリストを作成し、そのリストを表示しています。

```
<loc:listCategoriesMatchingKeyword name="all_categories_key"
   type="java.util.List"
   keyword="restaurant">
</loc:listCategoriesMatchingKeyword>
<%= all_categories_key.toString() %>
```

listChildCategories

listChildCategories タグでは、直下のサブカテゴリのフォーマットされていないリストを作成します。

表 14-46 に、listChildCategories タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-46 listChildCategories タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：all_categories_child	はい
type	オブジェクトのタイプ。java.util.List にする必要があります。	はい
parentCategory	親カテゴリの指定を含むオブジェクトの名前（親オブジェクトは、あらかじめ category タグを使用して作成します）。	いいえ

次の例では、変数 restaurant に関連付けられた親カテゴリのすべての子カテゴリのフォーマットされていないリストを作成し、そのリストを表示しています。

```
<loc:listChildCategories name="all_categories_child"
   type="java.util.List"
   parentCategory="restaurant">
</loc:listChildCategories>
<%= all_categories_child.toString() %>
```

listCreatedCommunities

listCreatedCommunities タグでは、指定した Oracle Application Server のモバイル使用が所有しているすべてのモバイル・コミュニティ（プライベート、共有およびシステム）のフォーマットされていないリストを作成します。コミュニティのタイプを含むモバイル・コミュニティの説明は、14-118 ページの「モバイル・コミュニティ」を参照してください。

表 14-47 に、listCreatedCommunities タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-47 listCreatedCommunities タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：list_cr_comm	はい
type	オブジェクトのタイプ。java.util.List にする必要があります。	はい
userName	操作をリクエストする Oracle Application Server Wireless ユーザーの名前。デフォルトは現行のユーザーです。	はい

次の例では、ユーザー Mike のリクエストによって、すべてのモバイル・コミュニティのフォーマットされていないリストを作成し、そのリストを表示しています。

```
<loc:listCreatedCommunities
      name="list_cr_comm"
      type="java.util.List"
      userName="Mike" />
<%= list_cr_comm.toString() %>
```

listCreatedPrivateCommunities

listCreatedPrivateCommunities タグでは、指定した Oracle Application Server のモバイル使用が所有しているすべてのモバイル・プライベート・コミュニティのフォーマットされていないリストを作成します。コミュニティのタイプを含むモバイル・コミュニティの説明は、14-118 ページの「モバイル・コミュニティ」を参照してください。

表 14-48 に、listCreatedPrivateCommunities タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-48 listCreatedPrivateCommunities タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：`list_cr_priv_comm`	はい
type	オブジェクトのタイプ。`java.util.List` にする必要があります。	はい
userName	操作をリクエストする Oracle Application Server Wireless ユーザーの名前。デフォルトは現行のユーザーです。	はい

次の例では、ユーザー `Mike` のリクエストによって、すべてのモバイル・プライベート・コミュニティのフォーマットされていないリストを作成し、そのリストを表示しています。

```
<loc:listCreatedPrivateCommunities
      name="list_cr_priv_comm"
      type="java.util.List"
      userName="Mike" />
<%= list_cr_priv_comm.toString() %>
```

listCreatedSharedCommunities

`listCreatedSharedCommunities` タグでは、指定した Oracle Application Server のモバイル使用が所有しているすべてのモバイル共有コミュニティのフォーマットされていないリストを作成します。コミュニティのタイプを含むモバイル・コミュニティの説明は、14-118 ページの「モバイル・コミュニティ」を参照してください。

表 14-49 に、`listCreatedSharedCommunities` タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-49 listCreatedSharedCommunities タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：`list_cr_shar_comm`	はい
type	オブジェクトのタイプ。`java.util.List` にする必要があります。	はい
userName	操作をリクエストする Oracle Application Server Wireless ユーザーの名前。デフォルトは現行のユーザーです。	はい

次の例では、ユーザー Mike のリクエストによって、すべてのモバイル共有コミュニティのフォーマットされていないリストを作成し、そのリストを表示しています。

```
<loc:listCreatedSharedCommunities
      name="list_cr_shar_comm"
      type="java.util.List"
      userName="Mike" />
<%= list_cr_shar_comm.toString() %>
```

listCreatedSystemCommunities

listCreatedSystemCommunities タグでは、指定した Oracle Application Server のモバイル使用が所有しているすべてのモバイル・システム・コミュニティのフォーマットされていないリストを作成します。コミュニティのタイプを含むモバイル・コミュニティの説明は、14-118 ページの「モバイル・コミュニティ」を参照してください。

表 14-50 に、listCreatedSystemCommunities タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-50 listCreatedSystemCommunities タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：list_cr_sys_comm	はい
type	オブジェクトのタイプ。java.util.List にする必要があります。	はい
userName	操作をリクエストする Oracle Application Server Wireless ユーザーの名前。デフォルトは現行のユーザーです。	はい

次の例では、ユーザー Mike のリクエストによって、すべてのモバイル・システム・コミュニティのフォーマットされていないリストを作成し、そのリストを表示しています。

```
<loc:listCreatedSystemCommunities
      name="list_cr_sys_comm"
      type="java.util.List"
      userName="Mike" />
<%= list_cr_sys_comm.toString() %>
```

listGeocodes

listGeocodes タグでは、ジオコーディングされた住所のフォーマットされていないリストを作成します。

表 14-51 に、listGeocodes タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-51 listGeocodes タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：hardware1	はい
type	オブジェクトのタイプ。java.util.List にする必要があります。	はい
businessName	指定した住所にあるビジネスまたは他のエンティティの説明的な名前。例：Mike's Hardware	いいえ
houseNumber	住所の番地番号。	いいえ
streetName	番地。	はい
secondLine:	番地の 2 行目。	いいえ
intersection	houseNumber が指定されていない場合は、交差する番地。	いいえ
city	都市名。	はい
state	2 文字の州コード（US）または郡コード（カナダ）。	はい
postalCode	郵便番号（主要な部分）。例：01742	はい
postalCodeExt	4 桁の追加番号など、郵便番号の拡張部分。	いいえ
country	国名または国コード。	はい
provider	優先順位がある場合は、リクエストに対して最初に選択されるサービス・プロバイダの名前。	いいえ

次の listGeocodes タグの例は、New Hampshire 州 Nashua で郵便番号 03060 の Daniel Webster Highway にジオコーディングされたすべての住所を表します。

```
<loc:listGeocodes
  name = "all_businesses"
  type = "java.util.List"
  streetName = "Daniel Webster Hwy"
  city = "Nashua"
  state = "NH"
  postalCode = "03060"
  country = "US" />
```

listLocationMarks

listLocationMarks タグでは、OracleAS Wireless ユーザーのロケーション・マークのフォーマットされていないリストを作成します。

表 14-52 に、listLocationMarks タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-52 listLocationMarks タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：list_marks	はい
type	オブジェクトのタイプ。oracle.panama.model.LocationMark にする必要があります。	はい
userName	ロケーション・マークを列挙する対象の Oracle Application Server Wireless ユーザーの名前。デフォルトは現行のユーザーです。	いいえ

次の例では、変数 list_marks に関連付けられたオブジェクトとして、ユーザー Mike のロケーション・マークのフォーマットされていないリストを作成し、そのオブジェクトの情報を表示しています。

```
<loc:listLocationMarks name="list_marks"
        type="java.util.List"
       userName="Mike"  />
<%= list_marks.toString() %>
```

listManeuvers

listManeuvers タグでは、運転経路のフォーマットされていないリストを作成します。

表 14-53 に、listManeuvers タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-53 listManeuvers タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：hardware_1	はい
type	オブジェクトのタイプ。java.util.List にする必要があります。	はい
businessName	指定した住所にあるビジネスまたは他のエンティティの説明的な名前。例：Mike's Hardware	いいえ
firstLine	番地。	いいえ
city	都市名。	いいえ

表 14-53（続き）listManeuvers タグのパラメータ

パラメータ名	説明	必須
state	2 文字の州コード（US）または郡コード（カナダ）。	いいえ
postalCode	郵便番号。	いいえ
country	国名。	いいえ

次の例では、2 つの住所（ある人のオフィスと自宅）間にルート myRoute を作成して、ルートのマップとそれに続く横罫線を表示し、そのルートのすべての運転経路のフォーマットされていないリストを表しています。

```
<loc:route name="myRoute" type="oracle.panama.spatial.jsptags.beans.Route"
xres="800" yres="600">
   <loc:address
        name="office"
        type="oracle.panama.model.Location"
        businessName="My office"
        firstLine="1 Oracle Dr"
        city="Nashua"
        state="NH"
        postalCode="03062"
        country="US"/>
   <loc:address
        name="home"
        type="oracle.panama.model.Location"
        businessName="My home"
        firstLine="2 Royal Crest Dr"
        city="Nashua"
        state="NH"
        postalCode="03060"
        country="US"/>
</loc:route>
<loc:listManeuvers name="all_maneuvers" type="java.util.List" routeID="myRoute" />
<%= all_maneuvers.toString() %>
```

listReverseGeocodes

listReverseGeocodes タグでは、逆ジオコーディングされた住所（指定したポイントにプロバイダによって関連付けられた住所）のフォーマットされていないリストを作成します。

表 14-54 に、listReverseGeocodes タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-54 listReverseGeocodes タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：hardware_1	はい
type	オブジェクトのタイプ。java.util.List にする必要があります。	はい
businessName	指定した住所にあるビジネスまたは他のエンティティの説明的な名前。例：Mike's Hardware	いいえ
firstLine	番地。	いいえ
city	都市名。	いいえ
state	2 文字の州コード（US）または郡コード（カナダ）。	いいえ
postalCode	郵便番号。	いいえ
country	国名。	いいえ
provider	優先順位がある場合は、リクエストに対して最初に選択されるプロバイダの名前。	いいえ

次の listReverseGeocodes タグの例は、指定したポイントにプロバイダによって関連付けられた住所を表します。

```
<loc:listReverseGeocodes
  name = "list_rev"
  type = "java.util.List"
  lon = "-71.4424"
  lat = "42.712"
  label = "You Are Here" />

<%= list_rev.toString() %>
```

map

map タグでは、指定の解像度で次のいずれかを示すマップを指定します。

- 1 つ以上のポイント
- 1 つのルート
- 1 つの運転経路

表 14-55 に、map タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-55 map タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例 : myMap	はい
type	オブジェクトのタイプ。oracle.panama.spatial.jsptags.beans.Map にする必要があります。	はい
points	作成するマップに含めるポイントのコレクションの名前。	いいえ
route	作成するマップに含めるルートの名前。	いいえ
maneuver	作成するマップに含める経路の名前。	いいえ
xres	画面表示単位によるマップの幅。	はい
yres	画面表示単位によるマップの高さ。	はい
provider	優先順位がある場合は、リクエストに対して最初に選択されるプロバイダの名前。	いいえ

次の map タグの例では、幅 400 ピクセル、高さ 300 ピクセルのマップ NEDCSmall を作成しています。マップの中心ポイントは map タグ内の address タグで定義した住所です。

```
<loc:map name="NEDCSmall" type="oracle.panama.spatial.jsptags.beans.Map"
       xres="400" yres="300">
   <loc:address
     name="NEDC"
     type="oracle.panama.model.Location"
     businessName="NEDC"
     firstLine="1 Oracle Dr"
     city="Nashua"
     state="NH"
     postalCode="03062"
     country="US"/>
</loc:map>
```

mobilePos

mobilePos タグでは、モバイル・ユーザーに関するポジショニング情報を備えたオブジェクトを作成します。

表 14-56 に、mobilePos タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-56 mobilePos タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：position	はい
type	オブジェクトのタイプ。oracle.panama.model.Point にする必要があります。	はい
userName	ポジショニング情報をリクエストする対象の Oracle Application Server Wireless ユーザーの名前。デフォルトは現行のユーザーです。	いいえ
requestingUser	ポジショニング情報をリクエストする対象の Oracle Application Server Wireless ユーザーの名前。デフォルトは現行のユーザーです。リクエスト側ユーザーが、userName に関するポジショニング情報の取出しを許可されていない場合は、リクエストに失敗します。	いいえ
failoverToDefaultLocationMark	TRUE（デフォルト）では、ユーザーがポジショニングできない場合、および requestingUser がポジショニング情報の取出しを許可されていない場合に、userName のデフォルト・ロケーション・マークが使用されます。FALSE では、ユーザーがポジショニングできない場合、リクエストに失敗します。	いいえ

次の例では、ユーザー Mike に関するポジショニング情報を備えたオブジェクトを作成しています。デフォルトでは、現在位置が取得できない場合、そのユーザーのデフォルト・ロケーション・マークが使用されます。また、この例は、ポジショニング情報も表示しています。

```
<loc:mobilePos
      name="position"
      type="oracle.panama.model.Point"
      userName="Mike"  />
<%= position.toString() %>
```

point

point タグでは、WGS 84 座標系（Oracle Spatial SRID 値 8307）を使用して、ポイントの経度と緯度の値を指定します。

表 14-57 に、point タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-57 point タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：my_pt	はい
type	オブジェクトのタイプ。oracle.panama.model.Location にする必要があります。	はい
lon	ポイントの経度の値（WGS 84 座標系）。例：-75.3	はい
lat	ポイントの緯度の値（WGS 84 座標系）。例：45.71	はい

次の point タグの例では、西経 75.3 度、北緯 45.71 度にポイントを指定します。

```
<loc:point
  lon = "-73.5"
  lat = "45.71"  />
```

removeAllMembers

removeAllMembers タグでは、モバイル・コミュニティからすべてのメンバーを削除します。（このタグは、コミュニティを削除しません。コミュニティを削除するには、deleteCommunity タグを使用します。）モバイル・コミュニティの説明は、14-118 ページの「モバイル・コミュニティ」を参照してください。

表 14-58 に、removeAllMembers タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-58 removeAllMembers タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：remove_all_members	はい
type	オブジェクトのタイプ。Boolean にする必要があります（操作に成功すると TRUE、操作に成功しないと FALSE です）。	はい

表 14-58 removeAllMembers タグのパラメータ（続き）

パラメータ名	説明	必須
userName	操作をリクエストする Oracle Application Server Wireless ユーザーの名前。デフォルトは現行のユーザーです。	いいえ
communityID	すべてのメンバーを削除するコミュニティに関連付けられた変数の名前。例：comm_private	はい

次の例では、ユーザー Mike のリクエストによって、変数 comm_private に関連付けられたモバイル・コミュニティからすべてのメンバーを削除しています。また、このコミュニティのメンバーの java.util.Enumeration オブジェクトを作成しています。

```
<loc:removeAllMembers
      name="remove_all_members"
      type="Boolean"
      userName="Mike"
      communityID="comm_private"  />
<loc:listAllMembers
      name="list_all_mem3"
      type="java.util.Enumeration"
      communityID="comm_private" />
<%= list_all_mem3.toString() %>
```

removeMembers

removeMembers タグでは、モバイル・コミュニティから 1 人以上のメンバーを削除します。モバイル・コミュニティの説明は、14-118 ページの「モバイル・コミュニティ」を参照してください。

表 14-59 に、removeMembers タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-59 removeMembers タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：remove_members	はい
type	オブジェクトのタイプ。Boolean にする必要があります（操作に成功すると TRUE、操作に成功しないと FALSE です）。	はい
userName	操作をリクエストする Oracle Application Server Wireless ユーザーの名前。デフォルトは現行のユーザーです。	いいえ

表 14-59 removeMembers タグのパラメータ（続き）

パラメータ名	説明	必須
communityID	メンバーを削除するコミュニティに関連付けられた変数の名前。例：comm_private	はい
communityMembers	コミュニティから削除される Oracle Application Server Wireless ユーザーがスペースで区切られたリスト。	いいえ

次の例では、ユーザー Mike のリクエストによって、変数 comm_private に関連付けられたモバイル・コミュニティからユーザー Oscar および Maria を削除しています。また、このコミュニティのメンバーの java.util.Enumeration オブジェクトを作成しています。

```
<loc:removeMembers
      name="remove_members"
      type="Boolean"
      userName="Mike"
      communityID="comm_private"
      communityMembers="Oscar Maria" />
<loc:listAllMembers
      name="list_all_mem2"
      type="java.util.Enumeration"
      communityID="comm_private" />
<%= list_all_mem2.toString() %>
```

route

route タグでは、指定のマップ解像度を持つルートを指定します。ルートには、経路、概観マップおよび経路マップが含まれます。

表 14-60 に、route タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-60 route タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例：myRoute	はい
type	オブジェクトのタイプ。oracle.panama.spatial.jsptags.beans.Route にする必要があります。	はい
xres	表示されるルートの幅を示す画面表示単位の値。	はい
yres	表示されるルートの高さを示す画面表示単位の値。	はい

表 14-60 route タグのパラメータ （続き）

パラメータ名	説明	必須
requestGeom	TRUE では、ルートのジオメトリがプロバイダによって作成および提供されます（たとえば、geometry タグとともに使用します）。FALSE（デフォルト）では、ルートのジオメトリは作成されません。	いいえ
requestMap	TRUE（デフォルト）では、ルート・マップがプロバイダによって作成および提供されますが、そのマップは実際には表示されません （マップを表示するには、map タグを使用します）。FALSE では、ルート・マップは作成されません。プロバイダによっては、requestMap の設定に関係なく常にマップを提供する場合があることに注意してください。	いいえ
provider	優先順位がある場合は、リクエストに対して最初に選択されるプロバイダの名前。	いいえ

次の例では、2 つの住所（ある人のオフィスと自宅）間にルート myRoute を作成して、ルートのマップとそれに続く横罫線を表示し、（iterateManeuvers タグと getMap および getNarrative ファンクション・コールを使用して）各運転経路とそれに続く横罫線を表しています。また、各運転経路の説明はリンクであり、ユーザーがクリックするとその経路マップを表示できます。

```
<loc:route name="myRoute" type="oracle.panama.spatial.jsptags.beans.Route"
xres="800" yres="600">
   <loc:address
        name="office"
        type="oracle.panama.model.Location"
        businessName="My office"
        firstLine="1 Oracle Dr"
        city="Nashua"
        state="NH"
        postalCode="03062"
        country="US"/>
   <loc:address
        name="home"
        type="oracle.panama.model.Location"
        businessName="My home"
        firstLine="2 Royal Crest Dr"
        city="Nashua"
        state="NH"
        postalCode="03060"
        country="US"/>
</loc:route>

<img src="<%= myRoute.getMap() %>">
```

```
<HR>

<loc:iterateManeuvers name="aManeuver"
type="oracle.panama.spatial.jsptags.beans.Maneuver" routeID="myRoute">
      <a href="<%= aManeuver.getMap() %>">
        <%= aManeuver.getNarrative() %>
      </a>
      <HR>
</loc:iterateManeuvers>
```

setCommunityName

setCommunityName タグでは、ジオコーディングするか、マップに配置するか、ルートの始点住所または終点住所として使用するか、ビジネス・ディレクトリ問合せの中心として使用する住所を指定します。

表 14-61 に、address タグのパラメータを示します（この表の内容の説明は、14-31 ページの「JavaServer Pages（JSP）ファイルの作成」を参照）。

表 14-61 setCommunityName タグのパラメータ

パラメータ名	説明	必須
name	戻されるオブジェクトの名前。例 : hardware_1	はい
type	オブジェクトのタイプ。Boolean にする必要があります（操作に成功すると TRUE、操作に成功しないと FALSE です）。	はい
userName	操作をリクエストする Oracle Application Server Wireless ユーザーの名前。デフォルトは現行のユーザーです。	いいえ
communityID	すべてのメンバーを削除するコミュニティに関連付けられた変数の名前。例 : comm_shared	はい
newName	コミュニティに割り当てられる説明的な新しい名前。	はい

次の例では、ユーザー Mike のリクエストによって、変数 comm_shared に関連付けられた既存のコミュニティのコミュニティ名を設定しています。コミュニティ名は Renamed Shared Community の値に設定されます。また、この例では、操作の結果も表示しています（TRUE または FALSE）。

```
<loc:setCommunityName
      name="set_comm_name"
      type="Boolean"
      userName="Mike"
      communityID="comm_shared"
      newName="Renamed Shared Community" />
<%= set_comm_name.toString() %>
```

ロケーション Java API の使用

アプリケーションの動作に、JSP タグのみで実現するよりも厳密な制御が必要な場合には、ロケーション Java API を使用して、JSP ファイルやサーブレットなどでアプリケーションを作成できます。

この項では、ロケーション Java API の使用について説明します。JSP の概念や MobileXML アプリケーションの作成方法については、このマニュアルの他の章を参照してください。

ジオコーディング

ジオコーディング・アプリケーションでは、ユーザーに住所を要求し、アプリケーションがその住所をジオコーディングします。この種のアプリケーションの場合は、例 14-8 のように住所の `SimpleForm` オブジェクトを構成することから始めることができます。

例 14-8 SimpleForm オブジェクトの構成

```
<SimpleResult>
  <SimpleForm target="EnterAddress2.jsp">
    <SimpleFormItem name="businessName" title="Business Name" value="Oracle"/>
    <SimpleFormItem name="houseNum"     title="House Number"  value="500"/>
    <SimpleFormItem name="street"    title="Street"     value="Oracle Parkway"/>
    <SimpleFormItem name="city"      title="City"       value="Redwood City"/>
    <SimpleFormItem name="state"        title="State"         value="CA"/>
    <SimpleFormItem name="postalCode"   title="Postal Code"   value="94065"/>
    <SimpleFormItem name="country"      title="Country"       value="US"/>
  </SimpleForm>
</SimpleResult>
```

次回に（ユーザーが各フィールドに値を入力した後に）アプリケーションが起動すると、例 14-9 のようにデータにアクセスできます。

例 14-9 住所データへのアクセス

```
String
  businessName = request.getParameter("businessName"),
  houseNumber  = request.getParameter("houseNum"),
  streetName   = request.getParameter("street"),
  city         = request.getParameter("city"),
  state        = request.getParameter("state"),
  postalCode   = request.getParameter("postalCode"),
  country      = request.getParameter("country");
```

ジオコーディングの実行には、例 14-10 のようにコールを使用できます。（例 14-10 には示されていませんが、別の形式の `SpatialManager.createlocation` では、緯度と経度の座標を持つポイントを指定します。この場合、`Location` オブジェクトは作成されますが、ジオコーディングは実行されません。）

例 14-10 住所のジオコーディング

```
Location address =
  SpatialManager.createLocation(
    businessName,
    houseNumber,
    new String[] { streetName },
    null,
    city,
    state,
    postalCode,
    null,
    country);
```

結果の緯度と経度の値には、例 14-11 のようにアクセスできます。

例 14-11 ジオコーディングされた住所の値へのアクセス

```
address.getLongitude()
address.getLatitude()
address.getAddressLine1()
address.getCity()
address.getState()
```

getLongitude および getLatitude メソッドが Point インタフェースから継承されることに注意してください。

国際住所 ローカルの住所書式にあわせてさらに調整するために、oracle.panama.spatial.intladdress パッケージに用意されている国際住所書式オプションを使用できます（国際住所書式の詳細は、14-13 ページの「住所書式（国際）の構成」を参照。）ユーザーに住所を入力させるために必要なステップの数が 1 つ増えます。最初に、住所入力フォームとともに表示されるように、ユーザーが国（住所書式）を選択する必要があります。このフォームは選択する国に依存するため、2 つの異なるステップを 1 つにまとめることはできません。

例 14-12 では、ユーザーが住所書式（US、German、French など）を選択できるように、ドロップダウンの SimpleFormSelect 要素を作成します。

例 14-12 住所書式の選択

```
<SimpleResult>
  <SimpleMenu>
    <%
      java.util.Iterator it =
       oracle.panama.spatial.intladdress.IntlAddressManager.getAddressFormats();
      while(it.hasNext())
      {
        String name = (String)it.next();
```

```
    %>
      <SimpleMenuItem target="enterIntlAddress.jsp?name=<%= name %>">
        <%= name %>
      </SimpleMenuItem>
    <%
      }
    %>
  </SimpleMenu>
</SimpleResult>
```

次のステップは、指定された住所書式に関連するすべてのアドレス・コンポーネントを要求するフォームを提供することです。例 14-13 のように、コンポーネントは指定された国に基づいて動的に決定されます。

例 14-13 指定された国用のアドレス・コンポーネントの要求

```
<SimpleResult>
  <SimpleForm target="readIntlAddress.jsp?name=<%= request.getParameter("name")
                                                  %>">
    <%
      java.util.Iterator addressComponentNames =
        oracle.panama.spatial.intladdress.IntlAddressManager.getAddressFormat(
          request.getParameter("name")).getComponentNames();
      int num = 1;
      while(addressComponentNames.hasNext())
      {
        String name = (String)addressComponentNames.next();
    %>
      <SimpleFormItem name="component<%= num++ %>" title="<%= name %>"/>
    <%
      }
    %>
  </SimpleForm>
</SimpleResult>
```

例 14-14 では、結果が表示されます。表示されるコンポーネントと行数は、国に依存します。

例 14-14 国固有の書式による住所の表示

```
<SimpleResult>
  <SimpleText>
<%
  String name = request.getParameter("name");
  boolean finished = false;
  java.util.Vector components = new java.util.Vector();
  for(int i = 1; !finished; i++)
  {
```

```
    String component = request.getParameter("component" + i);
    if(component != null)
      components.add(component);
    else
      finished = true;
  }
  String componentArray[] = new String[components.size()];
  for(int i = 0; i < componentArray.length; i++)
    componentArray[i] = (String)components.get(i);
  oracle.panama.spatial.intladdress.IntlAddress loc =
    oracle.panama.spatial.intladdress.IntlAddressManager.createAddress(
      name,
      componentArray);

  java.util.Iterator lines = loc.getAddressLines(false, true);
  while(lines.hasNext())
  {
    %>
      <SimpleTextItem>
        <%= (String)lines.next() %>
      </SimpleTextItem>
    <%
  }
%>
  </SimpleText>
</SimpleResult>
```

ロケーション・マーク

アダプタではロケーション・マークを処理できます。例 14-15 では、ロケーション・マークを配列に取り出しています。（この例では、ロケーション・マークに関連のないコードは省略されています。）

例 14-15 ロケーション・マークの取得

```
…
LocationMark locMarks[] = sr.getSession().getUser().getLocationMarks();
…
```

LocationMark により Location（住所）が拡張されることに注意してください。

ルーティング

ユーザーにより入力された始点住所と終点住所間のルーティング情報を提供するアダプタを作成できます。アダプタでは、次の操作を行う必要があります。

1. ルーティング設定とオプションを設定します。
2. ルートを計算します。
3. 計算したルートをユーザーに表示します（経路と経路マップのリストおよび概観マップなど）。

例 14-16 では、RoutingSettings オブジェクトを構成し、結果の概観マップと経路マップの解像度（高さと幅）を指定して、ルーティング設定とオプションを設定しています。

例 14-16 ルーティング設定とオプションの設定

```
RoutingSettings rS = new RoutingSettings(true, false);
rS.setSecondaryOption(RoutingOption.overviewMapHeight, "600");
rS.setSecondaryOption(RoutingOption.overviewMapWidth, "800");
rS.setSecondaryOption(RoutingOption.maneuverMapHeight, "600");
rS.setSecondaryOption(RoutingOption.maneuverMapWidth, "800");
```

例 14-17 では、ルートを計算して RoutingResult オブジェクトを戻しています。

例 14-17 ルートの計算

```
RoutingResult rR =
   SpatialManager.getRouter().computeRoute(
      startLoc,
      endLoc,
      null,   // via points
      rS,   // routing options
      Locale.US);
```

例 14-18 では、計算したルートをユーザーに表示し、経路と経路マップのリストおよび概観マップを表示します。（この例では、ルーティング API 固有のコードが太字で示されています。）

例 14-18 ユーザーに対するルートの表示

```
<%!
  public static String translate(String orig)
  {
    return oracle.panama.spatial.XMLEncoder.encodeToSimplifiedXML(orig);
  }
%>
```

```
<%
  oracle.panama.spatial.router.RoutingResult rR = ...
%>
<SimpleResult>
  <SimpleImage src="<%= translate(rR.getOverviewMapURL()[0].toString()) %>"/>
  <SimpleText>
    <%
      oracle.panama.spatial.router.Maneuver mans[] = rR.getManeuvers();
      for(int i = 0; i < mans.length; i++) {
    %>
      <SimpleTextItem>
        <%= mans[i].getNarrative() %>
      </SimpleTextItem>
    <% } %>
  </SimpleText>
</SimpleResult>
```

マッピング

典型的マッピング・アプリケーションの場合、ユーザーは住所を入力してマップを表示します。例 14-19 では、マップする住所のマップ・イメージ URL を取得しています。（Location 型の変数 loc には、以前からジオコーディングされている住所が含まれます。）

例 14-19 マップ・イメージ URL の取得

```
String url =
   SpatialManager.getMapper().getMapURL(
      loc,
      oracle.panama.imagex.ImageFormats.GIF,
      800,   // width
      600,   // height
      false);   // allow turning
```

例 14-19 では、一部のモバイル・デバイスの画面にマップが収まるように、API でイメージの幅と高さを切替えできるかどうかを、最後のパラメータで指定しています。この例では、このオプションは使用禁止になっています。

例 14-19 のように最初のパラメータとして単一の Point オブジェクトを渡すかわりに、Point オブジェクトの配列、または Location 型（住所）か YPBusiness 型のオブジェクトを渡すことができ、これにより Point インタフェースが拡張されます。

ビジネス・ディレクトリ（YP）

典型的ビジネス・ディレクトリ（YP）アプリケーションの場合、ユーザーは国、州および都市を指定してリージョンを入力し、ワインのテイスティングやワイナリに関連するなど、なんらかのカテゴリのビジネスを取得します。ユーザーには国、州および都市の入力を要求し、アプリケーションは正しいカテゴリとすべての関連ビジネスを判別する必要があります。

通常、カテゴリを判別する最初のステップは、SimpleForm オブジェクトを介してユーザーにカテゴリのキーワード（wine など）を要求することです。

次のステップは、例 14-20 のように、キーワードと一致するすべてのカテゴリを判別することです。

例 14-20 キーワードと一致するカテゴリの検索

```
YPFinder ypF = SpatialManager.getYPFinder();
YPCategory cats[] = ypF.getCategoryAtRoot().getCategoriesMatchingName(keyword);
```

例 14-21 に、選択対象のカテゴリを表示する簡単なユーザー・インタフェースを示します。ドロップダウン・メニューが表示され、ユーザーはそこから検索対象と最も一致しているカテゴリを選択します。

例 14-21 カテゴリ選択用のユーザー・インタフェース

```
<SimpleResult>
  <SimpleMenu>
    <%
      oracle.panama.spatial.yp.YPFinder ypF =
        oracle.panama.spatial.SpatialManager.getYPFinder();
      oracle.panama.spatial.yp.YPCategory cats[] =
        ypF.getCategoryAtRoot().getCategoriesMatchingName("auto");
      for(int i = 0; i < cats.length; i++)
      {
        %>
        <SimpleMenuItem
         target="listCategories.jsp?cat=<%= cats[i].getFullyQualifiedName() %>">
            <%= cats[i].getFullyQualifiedName() %>
        </SimpleMenuItem>
       <%
      }
    %>
  </SimpleMenu>
</SimpleResult>
```

選択したカテゴリの完全修飾された名前がアプリケーションにより判別された場合は、例 14-22 のように適切なカテゴリを取得できます。

例 14-22 カテゴリの検索

```
YPCategory cat = YPCategory.fromFullyQualifiedName(categoryNameString);
YPBusiness b[] =
  SpatialManager.getYPFinder().getBusinessesInCity(
    cat,
    country,
    state,
    city,
    Locale.US);
```

例 14-22 の場合、ドロップダウン・メニューは一般オブジェクトではなく String オブジェクトから選択させるため、カテゴリ・オブジェクトから String オブジェクト、String オブジェクトから元のカテゴリ・オブジェクトへと変換する必要があります。

トラフィック

トラフィック・サービス API に基づくアプリケーションを作成する手順は、次のとおりです。

1. 問合せ用の入力オブジェクト（CityInfo、RouteInfo、Point および Location など）を準備します。
2. TrafficReporter を取得して問合せを送ります。
3. TrafficReport を取得して情報を処理します。

以降は、典型的な操作の例を示します。例 14-23 では、都市レベルの問合せを実行しています。

例 14-23 都市レベルの問合せ

```
TrafficReporter reporter = SpatialManager.getTrafficReporter();
CityInfo c = new CityInfo("BOSTON", "MA", "US");
TrafficReport report = null;
try{
  report = reporter.getReportViaCity(c);
}catch(LBSException e){
  System.out.println(e.getLocalizedMessage());
}
```

例 14-24 では、方向を指定せずにルート・レベルの問合せを実行しており、双方向での事象が戻されます。

例 14-24 ルート・レベルの問合せ（双方向での事象）

```
RouteInfo r = new RouteInfo("US 3", null);
try{
  report = reporter.getReportViaRoute(r,c);
}catch(LBSException e){
  System.out.println(e.getLocalizedMessage());
}
```

例 14-25 では、方向（北）を指定してルート・レベルの問合せを実行しています。

例 14-25 方向を指定したルート・レベルの問合せ

```
try{
  report = reporter.getReportViaRoute(r,TrafficReporter.North,c);
}catch(LBSException e){
  System.out.println(e.getLocalizedMessage());
}
```

例 14-26 では、指定した緯度と経度のポイントから半径 10 マイル以内のエリアについて、ルート・レベルの問合せを実行しています。

例 14-26 緯度と経度のポイントと半径を指定したルート・レベルの問合せ

```
p = SpatialManager.createPoint(-71.0607, 42.3659);
try{
  report = reporter.getReportViaLocation(p, 10, TrafficReporter.MILES,
c);
}catch(LBSException e){
  System.out.println(e.getLocalizedMessage());
}
```

例 14-27 では、指定した住所から半径 10 マイル以内のエリアについて、ルート・レベルの問合せを実行しています。

例 14-27 住所を指定したルート・レベルの問合せ

```
Location loc = SpatialManager.createLocation(null, null, "839 Kearny
     Street", null, "San Francisco", "CA", null, null, "US");
try{
  report = reporter.getReportViaAddress(loc, 10, TrafficReporter.MILES);
}catch(LBSException e){
  System.out.println(e.getLocalizedMessage());
}
```

例 14-28 では、トラフィック・レポートを処理して有効な情報を取得しています。

例 14-28 トラフィック・レポートの処理

```
Calendar rTime = report.getReportTime();
TrafficIncident[] incidents = report.getIncidents();
if(incidents != null){
  for(int i=0; i<incidents.length; i++){
    TrafficIncident inc = incidents[i];
    String desc = inc.getDescription();
    String severity = inc.getSeverity();
    String type = inc.getType();
    TrafficRoute route = inc.getIncidentRoute();
    String[] locations = inc.getLocationRange();        //text description
    if(locations.length == 2){                          //a location range
      String exit1 = locations[0];
      String exit2 = locations[1];
    }
    else if(locations.length == 1){
      String exit1 = locations[0];                                //one location
    }
    Point geoLocation = inc.getIncidentLocation();      //lon/lat or
lon/lat+radius
    Calendar[] tr = inc.getTimeRange();
  }
}
```

例 14-29 では、トラフィック・サポートが提供されている都市のリストを戻します。

例 14-29 都市リストの戻り

```
TrafficCityManager manager = reporter.getCityManager();
CityInfo[] cities = null;
try{
  cities = manager.getActiveCities();
}catch(LBSException e){
  System.out.println(e.getLocalizedMessage());
}
```

例 14-30 では、指定した都市（San Francisco, California）でトラフィック・サポートが提供されているルートのリストを戻します。

例 14-30 都市のルート・リストの戻り

```
TrafficCityManager manager = reporter.getCityManager();
CityInfo sf = new CityInfo("SAN FRANCISCO", "CA", "US");
RouteInfo[] routes = null;
try{
  routes = manager.getRoutesInCity(sf);
}catch(LBSException e){
  System.out.println(e.getLocalizedMessage());
}
```

Web サービスの使用

Oracle Application Server のロケーション・サービスは、Geocoder、Mapper、Router または YPFinder のインタフェースの機能を使用する Wireless アプリケーションによって、Web サービスをサポートします。OracleAS Wireless 内で実行するアプリケーションの場合、アプリケーション開発者が特別なコーディングを追加する必要はありません。むしろ、Web サービスは、ジオコーディング、マッピング、ルーティングおよびビジネス・ディレクトリ（YP）をサポートするサービス・プロキシとして統合されています。

外部アプリケーションを開発する場合は、使用する言語が Java であるかどうかに関係なく、Wireless に付属する次の種類のファイルを使用してロケーション・ベースの Web サービスにアクセスできます。

- WSDL ファイル（14-107 ページの「WSDL ファイル」を参照）
- XML ファイル（14-107 ページの「XML ファイル」を参照）
- XSD ファイル（14-108 ページの「XSD ファイル」を参照）

WSDL ファイル

次の WSDL ファイルでは、ジオコーディング、マッピング、ルーティングおよびビジネス・ディレクトリ（イエロー・ページ）サービスの Web サービス・インタフェースを記述します。

- LbsSoapServiceGeocoder.wsdl
- LbsSoapServiceMapper.wsdl
- LbsSoapServiceRouter.wsdl
- LbsSoapServiceYPFinder.wsdl

XML ファイル

次の XML ファイルには、スキーマ・ファイルに対する XML 文書の例が含まれています。

- lbsAddress.xml
- lbsAddressArray.xml
- lbsAddressArray2.xml
- lbsBusiness.xml
- lbsBusinessArray.xml
- lbsCategory.xml
- lbsMap.xml
- lbsMapURL.xml

- lbsMapURLArray2.xml
- lbsPhone.xml
- lbsPhoneArray.xml
- lbsPoint.xml
- lbsPointArray.xml
- lbsRoute.xml
- lbsRouteSettings.xml

XSD ファイル

次の XSD ファイルでは、Web サービス・コールに XML パラメータおよび戻り値を記述します。

- lbsAddress.xsd
- lbsAddressArray.xsd
- lbsAddressArray2.xsd
- lbsBusiness.xsd
- lbsBusinessArray.xsd
- lbsCategory.xsd
- lbsMap.xsd
- lbsMapURL.xsd
- lbsMapURLArray2.xsd
- lbsPhone.xsd
- lbsPhoneArray.xsd
- lbsPoint.xsd
- lbsPointArray.xsd
- lbsRoute.xsd
- lbsRouteSettings.xsd

モバイル・ポジショニングの有効化

モバイル・ポジショニングを、ロケーション・ベースのアプリケーションの個々のユーザーまたはユーザー・グループに使用可能にすることができます。ユーザーのモバイル・ポジショニングとは、そのユーザーのロケーションを特定することです。ユーザーのモバイル・ポジショニングが使用可能になっている場合、OracleAS Wireless では、自動ポジショニングから動的に取得されるかデフォルト・ロケーション・マークから取得されるかに関係なく、そのユーザーの現在のロケーションを使用して、ロケーション・ベースのサービスまたはフォルダの可視性が決定されます。サービスまたはフォルダをシステム・リージョンまたは定義済のカスタム・リージョンに関連付けると、ロケーション依存（14-131 ページの「アプリケーションへのリージョンの関連付け」を参照）として定義できます。ロケーション依存サービスまたはフォルダがユーザーのポータルに表示されるのは、そのユーザーの（自動ポジショニングまたはデフォルト・ロケーション・マークからの）現在のロケーションが、関連付けられているリージョン内にある場合のみです。たとえば、ユーザーの現在のロケーションが Boston にある場合は、Boston のトラフィック情報サービスが表示され、それ以外の場合、そのユーザーにはサービスは表示されません。

モバイル・ポジショニングには、手動と自動の 2 種類があります。

- **手動ポジショニング**が発生するのは、ユーザーに特定のロケーションを割り当てる場合です。ロケーションとして、ユーザーに入力を要求した住所がジオコーディングされた結果、明示的に指定したロケーション・マークまたはユーザーのデフォルト・ロケーションを割り当てることができます。たとえば、ユーザーの自宅のロケーションをモバイル・ポジショニング用に指定すると、アプリケーションではその自宅エリアに関連する情報とオプションを（ユーザーが実際にいる現在の物理的な位置に関係なく）提供できます。
- **自動ポジショニング**（またはロケーション取得）が発生するのは、モバイル・デバイスのロケーションのポジショニング情報に基づいて、ユーザーのロケーションが自動的に判別される場合です。たとえば、配送トラックのドライバやサービス技術者のロケーションは、それぞれのモバイル・デバイスのロケーションに基づいて定期的に判別され、アプリケーションではユーザーに情報や指示を提供するときに、そのロケーション・データを考慮できます。

  自動ポジショニングの場合は、位置の更新頻度とユーザーのプライバシに関連して複数のオプションが用意されています。

この項では、手動ポジショニングと自動ポジショニングの詳細と、各種ポジショニングを使用可能にする方法について説明します。

手動ポジショニング

手動ポジショニングでは、特定のロケーションをモバイル・アプリケーションのユーザーに関連付けます。ロケーションを明示的に指定する方法（ユーザーが住所またはロケーション・マークを入力するなど）と、そのユーザーのデフォルト・ロケーション・マークを設定する方法があります。ロケーション・マークは、通常は緯度と経度の座標に関連付けられている位置で、名前が付いています。たとえば、アプリケーション・ユーザーがロケーション・マーク `MyHome` および `MyOffice`（それぞれユーザーの自宅とオフィスのロケーション）を作成し、それぞれにジオコーディングされた住所を関連付けることができます。このユーザーが `MyHome` をデフォルト・ロケーション・マークとして指定すると、モバイル・アプリケーションではユーザーの自宅住所がユーザーのロケーションとみなされます。

ユーザーがジオコーディングされていないデフォルト・ロケーション・マークを設定すると、ジオコーディング操作が実行されてから、ロケーション・マークがデフォルトとして設定されます。ジオコーディング操作に失敗した場合は、そのロケーション・マークをデフォルトとして設定しないことをお薦めします。これは、デフォルト・ロケーション・マークについては、多くの機能（ロケーション依存のサービスの可視性など）がジオコーディングされた情報に依存しているためです。

ロケーション・マークの詳細は、14-16 ページの「ロケーション・マーク」を参照してください。

手動ポジショニングの有効化

手動ポジショニングをユーザーに使用可能にするには、まず使用するロケーション・マークをセットアップします。API（`LocationMark` クラス）または Personalization Portal の Web インタフェースを使用して、1 つ以上のロケーション・マークを作成し（存在しない場合）、ロケーション・マークの 1 つをそのユーザーのデフォルトとして指定します。

注意： 自動ポジショニング（14-111 ページの「自動ポジショニング」を参照）がオフになっている場合、またはポジショニング・サーバーが一時的に使用不可能になっている場合は、手動ポジショニングが使用され、ユーザーのデフォルト・ロケーション・マークが使用されます。（自動ポジショニングのオンとオフは、OracleAS Wireless システム・マネージャを使用して切り替えることができます。）

Personalization Portal のインタフェースを使用して手動ポジショニングを使用可能にする手順は、次のとおりです。

1. Personalization Portal の Web インタフェースにログインします。
2. 「**LocationMarks**」タブをクリックします。

3. デフォルト・ロケーションにするロケーション・マークが存在しない場合は作成します。（**「作成」**をクリックし、表示されるページで情報を指定します。）
4. デフォルト・ロケーションにするロケーション・マークを選択します。
5. **「デフォルトを設定」**をクリックします。

自動ポジショニング

自動ポジショニングを使用すると、ユーザーのモバイル・デバイスのロケーション・ベースの位置に基づいて、そのユーザーのロケーションを判別できます。ポジショニングのサービス品質（QoS）値を設定して、ロケーションの更新頻度、つまり潜在的な精度を決定できます。

OracleAS Wireless API により、アプリケーションでは現行のセッションを介して、モバイル・ユーザーの現行のロケーションにアクセスできます（`oracle.panama.rt.Session` インタフェースの `getCurrentLocation()` を参照）。システムで自動ポジショニングがオンになっている場合は、モバイル・ポジショニング・システムからユーザーの現在の物理的位置が戻されます。自動ポジショニングがオフになっている場合、またはポジショニング・サーバーが一時的に使用不可能になっている場合は、そのユーザーのデフォルト・ロケーション・マークが戻されます。

プライバシとプライバシ関連情報のセキュリティは、ロケーション取得システムでは重要な問題です。OracleAS Wireless のロケーション・サービスにはプライバシ管理コンポーネントが用意されており、ユーザーは自分のプライバシ設定を表示、編集し、ポジショニング操作自体を使用可能または使用禁止にし、1 人以上のユーザー（モバイル・コミュニティ）に特定の時間枠内でポジショニング情報の取得を許可できます。また、アプリケーション開発者はこれらの機能に公開 API を介してアクセスできます。

自動ポジショニングは、図 14-5 に示すモバイル・ポジショニング・フレームワークにより制御されます。

図 14-5 モバイル・ポジショニング・フレームワーク

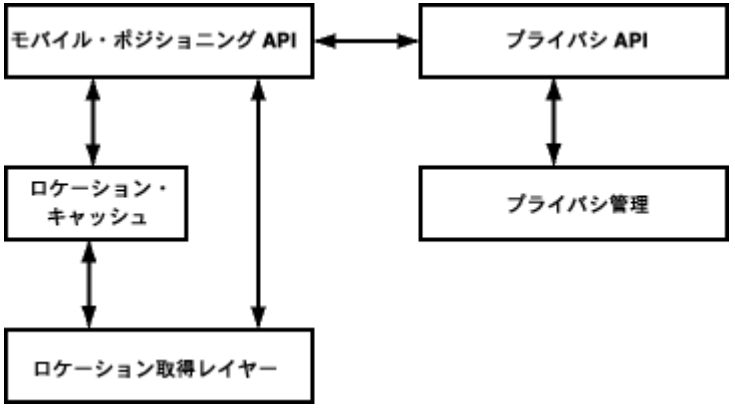


図 14-5 は、次のことを示しています。

- アプリケーション開発者は、モバイル・ポジショニング API とプライバシ API を併用してサービスを提供できます。
- アプリケーションのモバイル・ポジショニング API は、ロケーション・キャッシュ（14-114 ページの「ロケーション・キャッシュ」を参照）およびロケーション取得レイヤーと通信し、ユーザーのロケーションを判別します。キャッシュが使用されるかどうかは、14-114 ページの「ポジショニングのサービス品質」で説明するポジショニングのサービス品質（QoS）値の影響を受けます。
- ロケーション取得レイヤーは、実際の現在位置をロケーション・キャッシュとモバイル・ポジショニング API に渡します。
- プライバシ管理の論理は、モバイル・ポジショニング・フレームワークのうちプライバシに関連する側面を制御します。以降の各項を参照してください。

GPS デバイスを使用したロケーションの提供

モバイル・デバイスは、通常は汎地球測位システム（GPS）を介して提供されるそのモバイル・デバイスの現在のロケーションを、OracleAS Wireless サーバーに送信できます。送信された現在のロケーションには、モバイル・ポジショニング API とプライバシ API を使用して問い合せできます。

モバイル・デバイス上で動作し、デバイスの現在のロケーションを OracleAS Wireless サーバーに転記するクライアント・アプリケーション・プログラムを作成する必要があります。データは、OracleAS Wireless サーバー上で動作する JSP に対して、または Web サービスを介して転記できます。

データは XML フォーマットで、次のスキーマに従う必要があります。

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
elementFormDefault="qualified">
 <xsd:element name="MP_GPS">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element ref="USERNAME"/>
    <xsd:element ref="PASSWORD"/>
    <xsd:element ref="MSID"/>
    <xsd:element ref="TIME" minOccurs="0"/>
    <xsd:element ref="GMT" minOccurs="0"/>
    <xsd:element ref="POS"/>
    <xsd:element ref="ALTITUDE" minOccurs="0"/>
    <xsd:element ref="ALT_UNCERTAINTY" minOccurs="0"/>
    <xsd:element ref="VELOCITY" minOccurs="0"/>
    <xsd:element ref="BEARING" minOccurs="0"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
 <xsd:element name="ALTITUDE" type="xsd:string"/>
 <xsd:element name="ALT_UNCERTAINTY" type="xsd:string"/>
 <xsd:element name="BEARING" type="xsd:string"/>
 <xsd:element name="GMT" type="xsd:string"/>
 <xsd:element name="LAT" type="xsd:string"/>
 <xsd:element name="LONG" type="xsd:string"/>
 <xsd:element name="MSID" type="xsd:string"/>
 <xsd:element name="PASSWORD" type="xsd:string"/>
 <xsd:element name="POS">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element ref="LAT"/>
    <xsd:element ref="LONG"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
 <xsd:element name="TIME" type="xsd:string"/>
 <xsd:element name="USERNAME" type="xsd:string"/>
 <xsd:element name="VELOCITY" type="xsd:string"/>
</xsd:schema>
```

<USERNAME> 要素および <PASSWORD> 要素は、リクエストを許可するために OracleAS Wireless サーバーで使用されます。

<MSID> 要素は、モバイル・デバイスまたはユーザーのモバイル・ステーション ID です。

オプションの <TIME> 要素は、このロケーションが GPS で生成された時刻を示します。この値が欠落している場合は、OracleAS Wireless サーバーでデータを受信した時刻が使用されます。

オプションの <VELOCITY> 要素は、モバイル・デバイスの速度を m/ 秒単位で指定します。

オプションの <BEARING> 要素は、北からの時計回りで方位角（緯度）を指定します。

オプションの <ALTITUDE> 要素は、モバイル・デバイスの海抜高度を m 単位で指定します。

ロケーション・キャッシュ

ロケーション・キャッシュはメモリー内の領域であり、ここにはモバイル・ユーザーの ID、最後に取得されたロケーション情報および情報の収集時刻が一時的に格納されます。ロケーション・キャッシュ内でモバイル・ポジショニング・リクエストが検索され、ロケーションをリクエストされたユーザーのエントリがキャッシュ内にある場合は、キャッシュ・エントリの時刻と現在のリクエスト時刻の差が、ポジショニング・リクエストのポジショニングのサービス品質レベルと比較されます（ポジショニングのサービス品質については、14-114 ページの「ポジショニングのサービス品質」を参照）。

ポジショニング・リクエストがキャッシュ内の情報により満たされる場合、位置の検出、つまりネットワークのラウンドトリップ操作は不要です。

ポジショニングのサービス品質

ポジショニングのサービス品質（QoS）値により、次のことが制御されます。

- ロケーションを判別するためにデバイスの現在の位置とロケーション・キャッシュのどちらをチェックするか。
- ロケーション・キャッシュを調べる場合にアプリケーションで使用する、最後にキャッシュされたロケーション値の最大経過時間（つまり、値がロケーション・キャッシュに書き込まれてからの秒数）。

ポジショニングのサービス品質を指定するには、次の 2 つの方法があります。

- アプリケーションで使用するロケーション・キャッシュ内の位置の最大経過秒数として指定します。ロケーション・キャッシュ内の最新の位置が指定した時刻よりも古い場合は、デバイスの実際の現在位置が取得されてキャッシュに書き込まれ、アプリケーションで使用されます。値 0 を指定すると、ポジショニング・フレームワークでは常に実際のポジショニング結果が示され、ロケーション・キャッシュは検索されません。
- 次の文字列値の 1 つとして指定します。それぞれの値は、ポジショニングの品質レベルを表します。
  - Exact: ポジショニング・フレームワークでは、常に実際のポジショニング結果が示され、ロケーション・キャッシュは検索されません。この動作は、0 秒を指定した場合と同じです。
  - High: 高水準の精度を表します。
  - Medium: 中程度の精度を表します。
  - Low: Medium 値より低水準の精度を表します。

High、Medium および Low 値の場合、ポジショニング・フレームワークでは経験則により経過期間の値（秒数）が判断されます。

ポジショニングのサービス品質レベルにはシステム・デフォルトがあり、ユーザーが設定できます。ポジショニングのサービス品質レベルがポジショニング・リクエストで指定されていない場合は、システム・デフォルトが使用されます。レベルは、モバイル・ポジショニング API（14-120 ページの「モバイル・ポジショニング API」を参照）またはシステム・マネージャを使用して設定できます。

ポジショニングのサービス品質レベルを選択するときのトレードオフは、精度とアプリケーションのパフォーマンスです。値として 0 秒または Exact を指定すると実際の現在位置が取得されることが保証されますが、実際の位置を取得するには、モバイルのポジショニング・リクエストごとにネットワーク上でサービス・プロバイダへのラウンドトリップが必要です。この種のラウンドトリップ操作により、特に多数のユーザーに関するポジショニング・リクエストや同一ユーザーに関する多数のポジショニング・リクエストがある場合には、アプリケーションが低速になる可能性があります。アプリケーションで常に実際の位置を知る必要がある場合にのみ、値として 0 秒または Exact を使用してください。値 Low を使用すると、戻されるロケーションの精度は（ユーザーが移動しない場合を除き）最も低くなりますが、ロケーションがキャッシュから取得される確率が高くなり、ネットワークのラウンドトリップ操作が不要になります。ユーザーの移動距離が小さい場合や、ゆっくり移動している場合、または現在の実際のロケーションを知ることが重要ではない場合は、値 Low を使用するのが最も適切となることがあります。

ポジショニング・プロバイダの指定

自動モバイル・ロケーションは、`Positioner.requestPosition` ファンクションのコールによって問い合せられます（`Positioner` は `oracle.panama.mp` パッケージ内のクラスです）。`Positioner` オブジェクトは、1 つ以上のモバイル・ポジショニング・プロバイダに基づいています。他のロケーション・サービス・プロバイダと同様に、モバイル・ポジショニングを構成するには、名前、バージョン、URL、ユーザー名およびパスワードなどの情報を指定します。

ただし、他のロケーション・サービスとは異なり、モバイル・ポジショニング・サービスの場合は、ポジショニングを処理できるプロバイダが特定の 1 つのみである可能性があります。他のロケーション・ベースのサービスの場合、このようなことはまずありません。たとえば、カリフォルニアのマップをリクエストすると、複数のマッピング・プロバイダがそのマップを提供できます。ただし、特定の電話番号（+4412345678 など）のモバイル位置をリクエストした場合は、その位置を提供できるプロバイダが 1 つのみである可能性が高くなります。一般に、モバイル ID（通常は電話番号）では、Wireless の電話会社が識別されるため、1 つ以上のモバイル・ポジショニング・プロバイダも判別されます。したがって、アプリケーション開発者は、特定のモバイル・ポジショニング・プロバイダに基づいて位置を取得できるようにする必要があります。

アプリケーションの様々なニーズを満たすために、MPManager クラスには複数の getPositioner シグネチャが用意されています。

- getPositioner()
- getPositioner(MPProvider provider)
- getPositioner(MPProvider[] providers)

インターネット・ポータルに様々な電話会社からのサブスクライバがいて、各サブスクライバは使用するプロバイダを実行時にモバイル ID に基づいて動的に決定する必要がある場合があります。このニーズは、モバイル・ポジショニング・プロバイダのセレクタ・フック（oracle.panama.mp.MPProviderSelector インタフェースを介して実装）によりサポートされます。

プロバイダ・**セレクタ・フック**はモバイル ID を取り、ポジショニング・リクエストを処理できる MPProvider オブジェクトの配列を戻します。デフォルトのプロバイダ・セレクタ・フックは oracle.panama.mp.core.ProviderSelectorImpl により提供され、システム内のすべてのプロバイダを戻します。これは、ポジショニング・フレームワークではデフォルトでプロバイダが選択されないことを意味します。セレクタ・フックは、Positioner により positioner.requestPosition のコール時に使用され、getPositioner() のシグネチャに適用できます。Positioner のコール時にすでにプロバイダが指定されている場合は、セレクタ・フックは使用されません。

OracleAS Wireless API により、アプリケーションでは、現行のセッション（Session.getCurrentLocation()）を介してモバイル・ユーザーの現行のロケーションにアクセスできます。デフォルトでは、ユーザーのモバイル ID（ユーザーのプロファイルに格納）を使用して、モバイル・ポジショニング API がコールされ、現在位置が取得されます。ただし、高度なユーザーがポジショニングに異なる値を使用する必要がある場合は、oracle.panama.rt.hook.MobileIDHook インタフェースを実装して、独自のモバイル ID フックを記述できます。モバイル ID フックは、現行ユーザーの情報を取り、ポジショニング用にそのユーザーのモバイル ID を戻します。自動ポジショニングに失敗すると、システムは現在のロケーションとしてユーザーのデフォルト・ロケーション・マークにフェイルオーバーします。

プロバイダ・セレクタ・フックまたはモバイル ID フックを実装する必要はないことに注意してください。デフォルト設定がニーズを満たしている場合は、単にモバイル・ポジショニング・プロバイダを構成し、MPManager.getPositioner().requestPosition() をコールできます。

次にまとめを示します。

- Positioner は、システム内で構成されているすべてのモバイル・ポジショニング・プロバイダに基づいてシステム・デフォルトを使用するか、1 つ以上の特定のプロバイダのみに基づいてカスタマイズできます。
- システム・デフォルトを使用すると、プロバイダ・セレクタ・フックはシステム・デフォルト Positioner を選択する場合にのみ使用されます。セレクタ・フックはモバイル ID を取り、処理できるプロバイダ（1 つ以上）を決定します。バッチ問合せの場合は、バッチ内の最初のモバイル ID により、選択されるプロバイダが決定されます。

- Positioner が複数のプロバイダに基づいており、プロバイダがリクエストを処理できない場合は、フェイルオーバーが提供されます。

プログラムでは、PositionResult を使用する前に、そのエラー・コードが 0（ゼロ）以外であるかどうかをチェックする必要があります。

例 14-31 では、システム・デフォルト・プロバイダとデフォルトのポジショニング・サービス品質を使用して、ユーザーの位置を取得しています。

例 14-31 システム・デフォルト・プロバイダとデフォルト QoS を使用した位置の取得

```
Positioner positioner = MPManager.getPositioner();
PositionResult res = positioner.requestPosition("46708123456790");
Date timeStamp = res.getTimeStamp();
double lon = res.getPositionAreas()[0].getCenterPointLongitude();
double lat = res.getPositionAreas()[0].getCenterPointLatitude();
```

例 14-32 に、ユーザーの位置を指定してポジショニングのサービス品質レベルを指定する 2 つの例を示します。最初の例では品質を文字列で HIGH と指定し、2 番目の例では品質を秒数で指定しています。（ポジショニングのサービス品質を指定する方法は、14-114 ページの「ポジショニングのサービス品質」を参照してください。）

例 14-32 位置の指定による QoS の取得

```
PositionResult res = positioner.requestPosition("46708123456790", ServiceQoS.HIGH_
QUAL);

PositionResult res = positioner.requestPosition("46708123456790", new
ServiceQos(120));
```

例 14-33 では、特定のプロバイダの配列に基づいてユーザーの位置を取得しています。

例 14-33 プロバイダの配列に基づく位置の取得

```
MPProvider[] providers = new MPProvider[2];
providers[0] = MPManager.lookup("CellPoint", "1.2");
providers[1] = MPManager.lookup("Ericsson", "3.0");
Positioner positioner = MPManager.getPositioner(providers);
```

ポジショニング権の付与と取消し

デフォルトでは、ユーザーのロケーション情報にアクセスできるのは、そのユーザーのみです。ユーザーには、他のユーザーのロケーション情報にアクセスする権利は付与されていません。自分のロケーション情報へのアクセスを他のユーザーに許可する必要がある場合は、そのユーザーにポジショニング権を付与する必要があります。ポジショニング権を付与するユーザーは、後で付与した権利を取り消すことができます。

また、ポジショニング権は、特定の期間または定期的な間隔で付与することもできます。多くの場合、ユーザーは、自分のロケーション情報に対するアクセス権を他のユーザーに付与する期間を制限する必要があります。たとえば、この権利を同僚にウィークデイの午前 9:00 から午後 5:00 までは付与し、夜間や週末には付与しない必要があるとします。ユーザーは、この種の時間制限を次のように指定できます。

- 付与する権利の開始日と終了日
- 1 日の開始時刻と終了時刻
- 除外 : 開始日から終了日までのうち、土曜と日曜など、ポジショニング権から除外する日

モバイル・コミュニティ

モバイル・コミュニティとは、ポジショニング権の付与または拒否の対象にできる 1 人以上のユーザーのコレクションです。モバイル・ユーザーを 1 つ以上のコミュニティに割り当てて、ユーザーがコミュニティにポジショニング権を付与したり拒否できます。ユーザーは Personalization Portal を介してコミュニティ情報を表示および管理でき、アプリケーション開発者はこれらの機能に公開 API を介してアクセスできます。

モバイル・コミュニティの概念は、多数のモバイル・アプリケーションの使用例に役立ちます。たとえば、プロジェクト・チームはプロジェクト・コミュニティを作成できます。チームのメンバーは、自分のロケーション情報へのアクセス権を他のメンバーに個別に付与するかわりに、プロジェクト・コミュニティに付与できます。たとえば、フィールド・サービス管理者は、モバイル・ポジショニングとロケーション・ベースのアラートを使用して、サービス担当が付近にいる時期を知り、その担当に連絡してステータスを更新したり、ローカルな問題に対応させることができます。

可視性の概念は、コミュニティとそのメンバーに適用されます。可視性とは、システム・ユーザーがコミュニティまたはメンバーの存在を表示して、なんらかの関連情報を取得できることを指します。可視性は、リクエスト側ユーザーとコミュニティまたはメンバーとの関連に依存する場合があります。たとえば、リクエスト側ユーザーに管理権限が付与されているか、または問題のコミュニティのメンバーであるかなどです。可視性は、プライバシ API のコールを使用して実装されます。詳細は、14-121 ページの「プライバシ API」を参照してください。

ユーザーがコミュニティまたはそのメンバーに関する情報を表示するために行うリクエストの場合は、次の可視性条件が可能です。

- リクエスト側ユーザーは、コミュニティとそのメンバーを参照できます。
- リクエスト側ユーザーは、コミュニティを参照できますが、そのメンバーを参照できません。たとえば、コミュニティは、その有無をすべてのシステム・ユーザーが参照できるようにセットアップされていても、コミュニティ・メンバーに関する情報を使用できるのは管理者のみです。
- リクエスト側ユーザーはコミュニティを参照できないため、そのメンバーも参照できません。

可視性に対する様々なユーザー要件に対応するために、各種のコミュニティがサポートされています。コミュニティの作成時には、次のコミュニティ・タイプを指定できます。

- プライベート : プライベート・コミュニティを参照できるのはコミュニティの作成者のみで、作成者のみが全面的に制御します。コミュニティのメンバーなど、他のユーザーがプライベート・コミュニティの表示や操作を行うことはできません。
- 共有 : 共有コミュニティは、すべてのコミュニティ・メンバーが参照できますが、システムの他のユーザーは参照できません。コミュニティ・メンバーは、他のすべてのコミュニティ・メンバーを参照できます。コミュニティ・メンバーは自分自身をコミュニティから削除できます。
- メンバーの可視性を伴うパブリック : メンバーの可視性を伴うパブリック・コミュニティは、システムのすべてのユーザーが参照できます。システムのすべてのユーザーは、自分自身をコミュニティに追加したり、コミュニティから削除できます。
- パブリック・メンバーが可視性を制御 : メンバーが可視性を制御するパブリック・コミュニティは、システムのすべてのユーザーが参照できます。ただし、各メンバーは自分を他のユーザーに参照させるかどうかを制御できます。
- システム : システム・コミュニティはシステムのすべてのユーザーが参照できますが、メンバーを参照できるのは管理者権限を持つユーザーのみです。管理者権限を持っていないユーザーは、自分自身をシステム・コミュニティから削除できません。

次のコミュニティ操作がサポートされています。

- コミュニティの作成と初期メンバーの追加
- コミュニティの削除
- ユーザーが参照できるすべてのコミュニティのリストの表示
- ユーザーが参照できるすべてのコミュニティ・メンバーの表示
- コミュニティへのユーザーの追加（コミュニティ作成者の場合）
- コミュニティからのユーザーの削除（コミュニティ作成者、または自分自身を共有コミュニティから削除する場合は、すべてのコミュニティ・メンバー）

プライバシ・ディレクティブと自動ポジショニングの有効化 / 無効化

初期のデフォルトのプライバシ設定では、システムにはユーザーをポジショニングする権利がなく、ユーザーの位置をロケーション・キャッシュに一時的に格納して、そのユーザーのロケーション情報をキャッシュ・ログに書き込みます。ただし、管理者は OracleAS Wireless システム・マネージャで次のプライバシ・ディレクティブを使用して、異なるシステム・デフォルト・レベルのプライバシを指定できます。また、ユーザーは、Personalization Portal を介して各自のレベルやプライバシを制御できます。ここでは、プライバシ・ディレクティブを、適用するプライバシの降順で説明します。

- Disable Positioning and Caching: ユーザーのポジショニングは許可されません。システムにはユーザーをポジショニングする権利がなく、ユーザーのロケーションにはアクセスできません。この設定は、最も厳しいプライバシを提供します。
- Enable Positioning、Disable Caching: ユーザーのロケーション情報はキャッシュされません。システムにはユーザーをポジショニングする権利がありますが、そのユーザーのロケーション情報をロケーション・キャッシュに格納できません。この場合、ユーザーのロケーションは、常にポジショニング・サービス・プロバイダに直接アクセスすることで取得されます。

  たとえば、このディレクティブを使用すると、モバイル・ユーザーの移動を追跡できず、ユーザーのオフィスまたはサービス・プロバイダが提供するロケーションとして、任意の時点における位置がレポートされる可能性があります。
- No Log: ユーザーのロケーション情報はロケーション・キャッシュに格納されますが、キャッシュ・ログには書き込まれません。このユーザーのキャッシュ項目は、キャッシュから元に戻されてもログに書き込まれず、単に破棄されます。

  たとえば、No Log ディレクティブを使用すると、モバイル・ユーザーの現在位置は使用可能になりますが、以前の位置はロケーション・キャッシュから破棄されていると使用できない可能性があります。
- Enable Positioning and Caching: システムには、ユーザーのロケーション情報を取得してキャッシュする権利があります。

モバイル・ポジショニング API

モバイル・デバイスのポジショニングは、`Positioner` クラス内の対応する `requestPosition` ファンクションをコールすることで実行されます。この API により、アプリケーション開発者はポジショニングのサービス品質（QoS）レベルを指定できます（これらのレベルについては、14-114 ページの「ポジショニングのサービス品質」を参照）。

プライバシ API

モバイル・アプリケーションの開発者は、ロケーション・サービスのプライバシ API を介してプライバシ機能を管理できます。この項では、プライバシ API の例を使用して説明します。

LocationPrivacyManager クラス LocationPrivacyManager クラスは、ポジショニング権の付与、取消し、有効化と無効化、システムのプライバシ・オプションの設定と取得など、ロケーションのプライバシに関連する操作をすべて処理し、ユーザーに他のユーザーをポジショニングする権利があるかどうかをチェックします。また、このクラスは、LocationPrivacyAuth オブジェクトを取り出す手段を提供します。このオブジェクトには、プライバシ認可項目に関する情報が格納されます。

ユーザーは、grantAuthorization を使用して他のユーザーまたはモバイル・コミュニティに認可を付与できます。disableAuthorization を使用すると、認可を一時的に無効化できます。enableAuthorization を使用すると、無効化した認可をリカバリできます。付与した権利は、revokeAuthorization を使用して永続的に取り消すことができます。checkAuthorization を使用すると、特定の時刻にユーザーが他のユーザーをポジショニングする権利を持っているかどうかをチェックできます。

すべてのプライバシ認可操作はアプリケーション固有であり、その操作が実行されるアプリケーションにのみ影響します。

CommunityManager クラス CommunityManager クラスは、コミュニティの作成と削除、コミュニティ情報の取出しなど、コミュニティ関連の操作を処理します。単一のコミュニティに固有のコミュニティ操作は、Community インタフェースで定義します。

LocationPrivacyAuth インタフェース LocationPrivacyAuth インタフェースは、認可期間、認可が発生するサービス、権利を付与するユーザー、権利を受け取るユーザーなど、ロケーション認可項目に固有の情報を取り出すメソッドを提供します。

Community インタフェース Community インタフェースは、コミュニティ・オブジェクトに関する情報の取出し、コミュニティのメンバーの追加と削除およびコミュニティ属性の設定を行うメソッドを提供します。

AuthPeriod クラス AuthPeriod クラスは、ユーザーが他のユーザーにポジショニング権を付与するときに使用される時間の範囲をメンテナンスします。認可期間は、開始日、終了日、開始時刻、終了時刻および除外で構成されます。また、このクラスは、クラスが表す時間の範囲に特定の時刻が含まれるかどうかをチェックするためのメソッドを提供します。

LocationPrivacyException クラス LocationPrivacyException クラスは、PanamaException のサブクラスです。このクラスは、ロケーションのプライバシ固有の例外を表します。

プライバシ API の例 この項では、ロケーション・サービスのプライバシ API の例を示します。それぞれの例は、*iAS-wireless-home*¥sample¥sampleadapter¥mp¥privacy ディレクトリにあるサンプル・アダプタ SampleCommunityManager.java および SampleFriendFinder.java から抜粋したものです。この 2 つのサンプル・アダプタは、プライバシ API の主機能の例です。

例 14-34 では、指定したタイプのうち、ユーザーが参照できるすべてのコミュニティをリストしています。

例 14-34 指定したタイプのうちユーザーが参照できるコミュニティのリスト

```
CommunityManager commMan = CommunityManager.getInstance();
...
try{
     ResultSetEnumeration comms = commMan.getVisibleCommunities(user,type);
     while (comms.hasNextElement()){
          sfo = XML.makeElement(sfs,"SimpleFormOption");
          oracle.panama.model.Community comm =
(oracle.panama.model.Community)(comms.next());
          sfo.setAttribute("value",String.valueOf(comm.getId()));
          txt = XML.makeText(sfo,comm.getCreator().getName()+":"+ comm.getName() );
          sfo.appendChild(txt);
          sfs.appendChild(sfo);
    }
  }catch(Exception e){ throw new AdapterException(e);    }
```

例 14-35 では、ユーザー入力に基づいてユーザーまたはコミュニティにポジショニング権を付与しています。

例 14-35 ユーザーまたはコミュニティへのポジショニング権の付与

```
CommunityManager commMan = CommunityManager.getInstance();
LocationPrivacyManager priMan = LocationPrivacyManager.getInstance();

...

SimpleDateFormat ddf = new SimpleDateFormat("MM/dd/yyyy");
SimpleDateFormat tdf = new SimpleDateFormat("HH:mm");
Calendar startD,endD,startT,endT=null;
try{
    startD = Calendar.getInstance();
    startD.setTime(ddf.parse(sdate));

    endD = Calendar.getInstance();
    endD.setTime(ddf.parse(edate));
```

```
    startT = Calendar.getInstance();
    startT.setTime(tdf.parse(stime));

    endT = Calendar.getInstance();
    endT.setTime(tdf.parse(etime));

}catch(ParseException e){
      showError(result,sr,"Illegal Date Format","&grantmenu=y");
      return;
}

StringTokenizer st = new StringTokenizer(excl,",");
String exclDate = null;
byte exclusions=0;
while (st.hasMoreTokens()) {
    exclDate=st.nextToken();
    if ("Mon".equals(exclDate))
         exclusions =(byte)(exclusions|AuthPeriod.MONDAY);
    else if ("Tue".equals(exclDate))
         exclusions =(byte)(exclusions | AuthPeriod.TUESDAY);
    else if ("Wed".equals(exclDate))
         exclusions =(byte)(exclusions | AuthPeriod.WEDNESDAY);
    else if ("Thu".equals(exclDate))
         exclusions =(byte)(exclusions | AuthPeriod.THURSDAY);
    else if ("Fri".equals(exclDate))
         exclusions =(byte)(exclusions | AuthPeriod.FRIDAY);
    else if ("Sat".equals(exclDate))
         exclusions =(byte)(exclusions | AuthPeriod.SATURDAY);
    else if ("Sun".equals(exclDate))
         exclusions =(byte)(exclusions | AuthPeriod.SUNDAY);
    else {
        showError(result,sr,"Illegal Exclusions.", "&grantmenu=y");
        return;
    }
 }

 AuthPeriod period = new AuthPeriod(startD,endD, startT,endT, exclusions);
 oracle.panama.model.Community commObj = null;
 User posUserObj = null;
 try{
    if (community!=null && !community.equals("")){
       commObj = commMan.getCommunity(Long.parseLong(community));
       priMan.grantAuthorization(service,owner,commObj,period);
    }
```

```
   else{
      posUserObj = services.lookupUser(positionUser);
      priMan.grantAuthorization(service,owner,posUserObj,period);
  }
}catch(PanamaException e){ throw new AdapterException(e);   }
```

ロケーション・イベント・サーバー

ロケーション・イベント・サーバーは、ロケーション・ベースの条件が発生すると、イベントを生成します。イベントの結果として、ロケーション・ベースのアラートが、モバイル・ユーザーの現行のロケーションに基づいて配信されます。

Wireless アラート・エンジンを使用すると、ユーザーはロケーション・ベースのアラート・サービスをサブスクライブし、ロケーション・ベースの条件を指定できます。ロケーション・ベースの条件が満たされると、アラート・エンジンはロケーション・イベントを受信します。アラートを配信するための他のすべての条件も満たされた場合、アラート・エンジンはサブスクライバにアラートを送信します。次に、ロケーション・ベースのアラートをモバイル・ユーザーに提供できる典型的な使用例を示します。

- 空港に到着した旅行者が、乗ろうとするリムジン・バスが空港から 1 マイル以内にある場合にアラートを受信します。
- プロジェクト・チームの一員が、チームの他のメンバー全員が本社にいる場合にアラートを受信します。
- フィールド・サービス・コーディネータが、新規のサービス・リクエストを受け、それを処理できるエンジニアがサービス・センターから 2 マイル以内にいる場合にアラートを受信します。
- 気象情報の利用者が、予報が雪で、姉が家にいない場合にアラートを受信します（彼の姉は旅行が多く、不在時の植物の世話を彼が頼まれているため）。

潜在的には、より複雑な使用例が考えられます。たとえば、ロケーション関連の条件とロケーションには直接関連しない条件の組合せです。

ロケーション・イベント・サーバーの概要

ロケーション・ベースのアラートとは、ロケーション・ベースの条件のあるアラート・サービスです。

ロケーション・ベースの条件とは、条件有効期限と条件評価モードのような、ロケーション・ベースのアラート基準と関連情報のセットです。この条件が満たされるのは、条件にあるすべての基準が満たされた場合のみです。

ロケーション・ベースのアラート基準とは、ロケーション・ベースの条件の構成要素です。各基準には、ターゲット、リージョンおよびタイプの 3 つの要素があります。ターゲットは、ユーザー、コミュニティまたはモバイル・デバイスです。リージョンは、システム定義ロ

ケーションまたはユーザー定義ロケーションです。タイプは、リージョンに関するターゲットの位置を示す IN または OUT にする必要があります。ロケーション・ベースのアラート基準の例は、次のとおりです。

- チームのすべてのメンバーがシカゴにいる。
- スミス氏はニューヨーク州以外の場所にいる。

ロケーション・イベント・サーバーとは、ポジショニング・サービス・プロバイダからのロケーション関連情報の取出し、ロケーション・ベースの条件の評価、条件が満たされた場合のイベントの生成を行うスタンドアロン・プロセスです。ロケーション・イベント・サーバーは、スケジューリングを行うためにモバイル・ポジショニング・コンポーネントおよびリージョン・モデル・コンポーネントと連携します。つまり、条件評価結果は定期的に更新され、条件が満たされた場合は、条件を生成したクライアントにロケーション・イベントが送信されます。

ロケーション・イベント・クライアントとは、ロケーション・ベースの条件を指定し、ロケーション関連情報をレポートする Wireless アプリケーションまたはシステム・コンポーネントです。各ロケーション・イベント・クライアントには、サーバーとクライアントの間の双方向通信を処理する**ロケーション・イベント・エージェント**があります。ロケーション・イベント・エージェントは、クライアント・インスタンスで作成されたロケーション・ベースの条件を取得し、その条件をサーバー側に登録します。エージェントは、サーバーからロケーション・イベントを受信し、ロケーション・イベント・ハンドラを起動してそのイベントを処理します。また、指定したロケーション条件の評価結果に対するリクエストであるプル問合せをサポートします。

ロケーション・イベントは、条件を生成したクライアントや、1 つ以上の他のクライアント、またはその任意の組合せに送信できます。条件がアクティブ化されたとき、アプリケーションではロケーション・イベントを受信する受信者を決定できます。

複数のロケーション・イベント・サーバーおよびロケーション・イベント・クライアントを構成して使用できます。

ロケーション・イベント・クライアントを実装するために、次の Java クラスが用意されています。

- ロケーション・ベースの条件（LBCondition クラス、126 ページの「ロケーション・ベースの条件オブジェクト（LBCondition）」を参照）
- ロケーション・イベント・エージェント（LBEventAgent クラス、14-127 ページの「ロケーション・イベント・エージェント・オブジェクト（LBEventAgent）」を参照）
- ロケーション・イベント・ハンドラ（LBEventHandler クラス、128 ページの「ロケーション・イベント・ハンドラ・オブジェクト（LBEventHandler）」を参照）

ロケーション・イベント・エージェントの例

例 14-36 では、ロケーション・イベント・エージェントを作成し、ロケーション・ベースの条件をアクティブ化しています。その結果、通学している学校に関連付けられた地域の外に子供が出ると、その両親にアラートが送信されます。別のロケーション・イベント・エージェント（anotherAgent）では、条件が満たされると生成されるイベントをサーバーから受信します。

例 14-36 ロケーション・イベント・エージェント

```
 try{
     LBEventAgent alertAgent = factory.createLBEventAgent("alertAgent",true);
     LBEventHandler handler = new ALBEventHandlerImpl();
     alertAgent.registerLBEventHandler(handler);
     User parent = factory.createUser(USERNAMEPARENT);
     User child = factory.createUser(USERNAMECHILD);
     LocationPrivacyDomain privacyDomain = new LocationPrivacyDomain();
     LBCondition condition = factory.createLBCondition(LBCondition.MODE_ONCE, null,
        parent, privacyDomain);
     condition.addCriteria(String.valueOf(child.getId()),
        LBCondition.TARGETTYPE_USER,LBCondition.TYPE_OUT,SCHOOL_REGION_ID);
     alertAgent.activateCondition(condition,null,null,"anotherAgent",false);
  }catch(LBEventException e){
.......
  }
```

ロケーション・ベースの条件オブジェクト（LBCondition）

ロケーション・ベースの条件（LBCondition）オブジェクトは、ロケーション・ベースの条件を表します。典型的なアラート条件は複数の基準で構成され、それぞれターゲット、基準のタイプ（IN または OUT）およびリージョン（システム・リージョン、カスタム・リージョン、またはユーザー定義リージョン）を定義します。指定した基準間の関係は AND であり、すべての基準が満たされた場合のみ条件が満たされます。

LBCondition オブジェクトは、条件有効期限および条件評価モードも指定します。条件有効期限は、条件が無効になる時期を示します。条件評価モードは、次のいずれかにする必要があります。

- 1 回のみ評価。最初に条件が満たされると、その条件は再度評価されず、条件ステータスは非アクティブになります。
- 有効期限まで評価。条件が満たされると、ロケーション・イベントはロケーション・イベント・クライアントに送信されます。条件は、すでに満たされているかどうか、または満たされる回数に関係なく、有効期限までアクティブのままです。条件が満たされ、次には満たされず、再び満たされた場合は、新規のイベントがユーザーに送信されます。（たとえば、条件が「ユーザー Smith は Boston にいる」で、Smith が Boston に入

り、Boston を出て、再び Boston に入った場合、イベントは Smith が Boston に入るたびに送信されます。）

ロケーション・イベント・エージェント・オブジェクト（LBEventAgent）

ロケーション・イベント・エージェント（LBEventAgent）・オブジェクトは、ロケーション・イベント・クライアントにかわってロケーション・イベント・サーバーと通信します。ロケーション・イベント・エージェント・オブジェクトは、次の操作を実行します。

- ロケーション・ベースの条件をアクティブ化します。
- ロケーション・ベースの特定の条件が満たされているかどうかに関する問合せをサポートします。
- ロケーション・イベント・クライアントでロケーション・イベント・ハンドラを登録してスレッドを開始し、ロケーション・イベントをリスニングできるようにします。ロケーション・イベントを受信すると、ロケーション・イベント・ハンドラが起動してイベントを処理します。
- サーバーからのロケーション・ベースの条件を解除します。

各ロケーション・イベント・エージェントには名前およびメッセージ・チャネルがあります。ロケーション・イベント・クライアントで各ロケーション・イベント・エージェントを作成するとき、他のエージェントがその新規のエージェントと同じ名前を共有できるかどうかを指定できます。同じ名前のロケーション・イベント・エージェントは同じメッセージ・チャネルを共有します。つまり、そのメッセージ・チャネルに送信されたロケーション・イベントは、同じ名前のロケーション・イベント・エージェント間に分散されます。

ロケーション・ベースの条件がアクティブ化されると、条件の情報がロケーション・イベント・サーバーに送信され、サーバーは条件の評価を開始します。evaluationMode パラメータが activateCondition メソッドに渡された場合、そのパラメータの値は LBCondition オブジェクトで定義された評価モードより優先されます。RecipientAgent パラメータが activateCondition メソッドに渡された場合、そのパラメータによって受信者エージェントの名前が指定されます。つまり、条件は 1 つのエージェントで作成でき、その条件が満たされる場合、イベントは別のエージェントに送信できます。

ロケーション・イベント・エージェントの isSatisfied メソッドは、特定の条件が満たされているかどうかをチェックします。条件がアクティブでない場合は、isSatisfied メソッドによって条件評価が開始され、これには少し時間（条件の複雑さに応じて、数秒から数分）がかかる場合があります。checkStatusNoWait メソッドも、特定の条件が満たされているかどうかをチェックしますが、条件がアクティブでない場合、条件評価はアクティブ化されません。

ロケーション・イベント・ハンドラ・オブジェクト（LBEventHandler）

ロケーション・イベント・ハンドラ（LBEventHandler）・オブジェクトは、公開インタフェースです。アプリケーション開発者は、ハンドラのインタフェースを実装し、ロケーション・イベント・エージェントに登録します。実装はスレッド・セーフにする必要があります。ロケーション・イベント・ハンドラは、ロケーション・イベントの処理を行います。

ロケーション・イベント・エージェントがロケーション・イベントを受信すると、そのエージェントに登録されているロケーション・イベント・ハンドラの handleLocationEvent メソッドが起動されます。handleLocationEvent メソッドは、ロケーション・ベースの条件を一意に識別する条件 ID、条件が満たされているかどうか、エラーがあるかどうかを指定するイベント型、およびイベントが生成された時刻を受け取ります。

ロケーション・イベント・サーバーの構成オプション

Wireless システム・マネージャを使用してロケーション・イベント・サーバーを構成する手順は、次のとおりです。

1. Wireless サーバーの**「システム」**タブのページで、**「サイト管理」**をクリックします。
2. クリックして**「コンポーネント構成」**を展開します。
3. **「関連ロケーション」**の下で、**「ロケーション・イベント・サーバー」**をクリックします。

ロケーション・イベント・サーバーでは、次の構成オプションを使用できます。選択したオプションは、ロケーション・イベント・クライアントのアプリケーションの動作とパフォーマンスに影響を与えます。

- 待機なしプル・リクエストのデフォルト有効期間 (秒)

  ロケーション・イベント・エージェントでは、checkStatusNoWait メソッドを使用して、ロケーション・イベント・サーバーから待機なしで結果をプルできます。待機なしプル・リクエストの有効期間によって、プルされた結果の有効期間、および現在有効とみなされるかどうかが判断されます。有効期間内に生成された結果は、有効とみなされます。プルされた結果が有効でない場合、待機なしプル・リクエストは、サーバーによる新規の結果の生成を待機しません。たとえば、有効期間が 600 秒（10 分）で、ユーザーの位置の最新レポートが 11 分前であった場合、そのポジショニング・レポートは有効とはみなされず、checkStatusNoWait メソッドは待機せずに戻ります。

  有効期間が長いほど、ポジショニング・レポートが有効とみなされる可能性は高くなります。ただし、アプリケーションで最新のポジショニング情報が必要とされる場合は、短い有効期間が必要になります。

- プル・リクエストのデフォルト有効期間 (秒)

  ロケーション・イベント・エージェントでは、isSatisfied メソッドを使用して、ロケーション・イベント・サーバーから結果をプルできます。プル・リクエストの有効期間によって、プルされた結果が有効かどうかが判断されます。有効期間内に生成された結果は、有効とみなされます。プルされた結果が有効でない場合、プル・リクエストは、

サーバーが新規の結果を生成するまで待機します。たとえば、有効期間が 600 秒（10 分）で、ユーザーの位置の最新レポートが 11 分前であった場合、そのポジショニング・レポートは有効とはみなされず、isSatisfied メソッドは、ユーザーの位置のレポートが次に受信されるまで待機します。

有効期間が長いほど、ポジショニング・レポートが有効とみなされる可能性は高くなり、isSatisfied メソッドがポジショニング情報を受け入れて戻る可能性も高くなるため、アプリケーションを続行できます。ただし、アプリケーションで最新のポジショニング情報が必要とされる場合は、isSatisfied メソッドが新規情報を待機する間、アプリケーションを遅延させても、短い有効期間が必要になります。

- ロケーション・イベント・リスナーのデフォルト数

  ロケーション・イベント・エージェントには、ロケーション・ベースのイベントをリスニングする複数のリスナーを設定できます。この設定では、1 つのロケーション・イベント・エージェントが使用するリスナーの数を指定します。

  この値は、システムのワークロードに基づいて指定する必要があります。多数のロケーション・ベースの条件が作成および処理される場合は、1 より大きい値（5 や 10 など）が適切と考えられます。ただし、少数のロケーション・ベースの条件が作成および処理される場合は、1 つのロケーション・イベント・リスナーで十分です。アプリケーションでは、このデフォルトを上書きできます。

各ロケーション・イベント・サーバーには、次の値を指定できます。

- ポジショニング・スケジューラの数

  各ロケーション・イベント・サーバーには、ロケーション・ベースの条件を処理する 1 つ以上のポジショニング・スケジューラを設定できます。この設定では、各ロケーション・イベント・サーバーのポジショニング・スケジューラの数を指定します。

  この値は、システムのワークロードに基づいて指定する必要があります。多数のロケーション・ベースの条件が作成および処理される場合は、1 より大きい値（5 や 10 など）が適切と考えられます。ただし、少数のロケーション・ベースの条件が作成および処理される場合は、1 つのポジショニング・スケジューラで十分です。システム管理者はロケーション・イベント・サーバーのパフォーマンスを監視し、それに従ってこの値を調整できます。

リージョン・モデル・ツールの使用

リージョン・モデル・ツールを使用すると、Wireless ポータル・サービスの管理者はリージョンを管理し、サービスまたはフォルダをロケーション依存にすることができます。作成時に、サービスまたはフォルダにシステム・リージョンまたは以前に作成したカスタム・リージョンを関連付けて、ロケーション依存として指定できます。ロケーション依存サービスまたはフォルダがユーザーのポータルに表示されるのは、そのユーザーの（自動モバイル・ポジショニングまたはデフォルト・ロケーション・マークからの）現在のロケーションが、指定したリージョン内にある場合のみです。

リージョンとは、単に地理的エンティティ、つまりロケーションです。リージョンは、小さい場合（番地など）と大きい場合（国など）があります。リージョンは、対象となる住所とロケーション（空港や博物館など）の場合と同様にポイントで表すか、州や国の場合と同様にポリゴンで表すことができます。

リージョン・モデリングを使用したサービスとフォルダの可視性

次のように、特定のリージョンを様々なアプリケーションおよびサービスに対して定義できます。

- 選択した都会エリアのシティ・ガイド。このエリア内のユーザーは、関連するサービスと情報（レストラン・リストや広告など）のみを受信します。
- 特定のランキングを持つ大学、または特定の領域の専門大学。入学希望者と両親は、これらのロケーションに関する情報を受信できます。
- 1 つの都市または複数の州にまたがるエリア内の美術館。美術愛好家は美術館めぐりのプランを立てることができます。

会社が多数の専門サービスを提供しており、ユーザーがリージョンに連結された個々のサービスにサブスクライブして支払えるようにする必要があるとします。たとえば、あるユーザーはアメリカ全体のシティ・ガイドにサブスクライブし、別のユーザーは南東部の州のシティ・ガイドにのみサブスクライブできます。

このシティ・ガイドの例を実装する手順は、次のとおりです。

1. 静的でロケーション依存でないフォルダ City_guide を作成します。
2. City_guide フォルダの下に、ボストン、サンフランシスコおよびカリフォルニアのシティ・ガイド・サービスを作成します。
3. ある都市の住所にデフォルト・ロケーション・マークを設定します。この住所がボストンにある場合、ユーザーにはボストンのシティ・ガイドが表示され、サンフランシスコにある場合はサンフランシスコとカリフォルニアのガイドが表示されます。

別の使用例として、複数のサービスが 1 つのリージョンに関連している可能性があります。その場合は、ロケーション依存のフォルダを作成し、（各サービスをリージョンのロケーション依存として指定するかわりに）そのフォルダに関連サービスを置きます。たとえば、ATM ロケータ、Flight Gate Information、Airport Parking Information および Taxi Finder

の各サービスをリージョン Airport に関連付けており、Printer Finder、Conference Room Scheduler および Cafeteria Menu の各サービスをリージョン Office に関連付けているとします。この場合は、2 つのロケーション依存フォルダ Airport および Office を作成し、それぞれ Airport および Office リージョンに関連付けることができます。

フォルダとリージョン階層

リージョンはフォルダに格納されます。フォルダは階層形式で編成できます（つまり、フォルダにフォルダを入れることができます）。2 つのトップレベル・フォルダ System-Defined Regions および Custom Regions があります。

- **システム定義リージョン**は、定義済エリア、つまり国を含む大陸の階層形式で配置されます。アメリカにはさらに州が含まれ、州には郵便番号、郡および市が含まれます。
- **カスタム・リージョン**は、ユーザーが住所を 1 つ入力するか、他の 1 つ以上のリージョン（システム定義またはカスタム）を選択して作成します。

アプリケーションへのリージョンの関連付け

ロケーション依存にするアプリケーションを指定する場合は、そのサービスを適用するリージョンまたはそのサービスが関連するリージョンを指定する必要があります。リージョンを指定するには、それがシステム定義リージョンまたはカスタム・リージョンとして存在している必要があります。カスタム・リージョンの場合は、それがリージョン・モデル・ツールを使用して作成されている必要があります。

アプリケーションをロケーション依存にし、リージョン・モデル・ツールを使用する手順は、次のとおりです。

1. Wireless サーバーの**「サービス」**タブのページで、**「アプリケーション」**タブをクリックします（選択されていない場合）。
2. ロケーション依存の（またはロケーション依存にする）アプリケーションを選択し、**「編集」**をクリックします。
3. **「追加情報」**をクリックします。
4. **「ロケーションに依存」**を使用可能に（チェック）します。
5. リージョン・モデル・ツールを起動するには、図 14-6 のように、**「リージョン名」**ボックスの横にある**フラッシュライト・アイコン**をクリックします。

図 14-6 リージョン・モデル・ツールを起動するためのページ

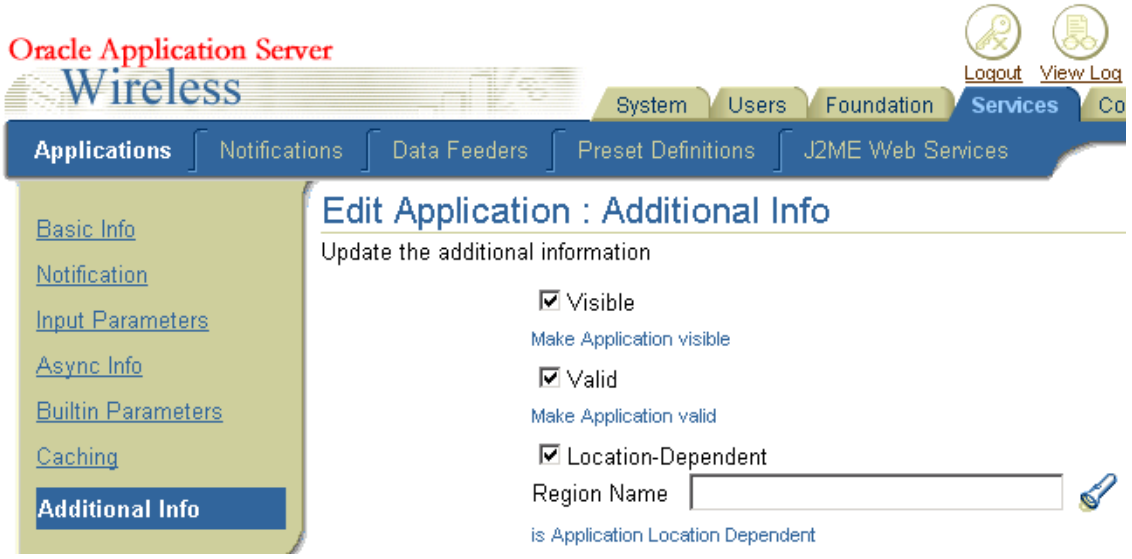
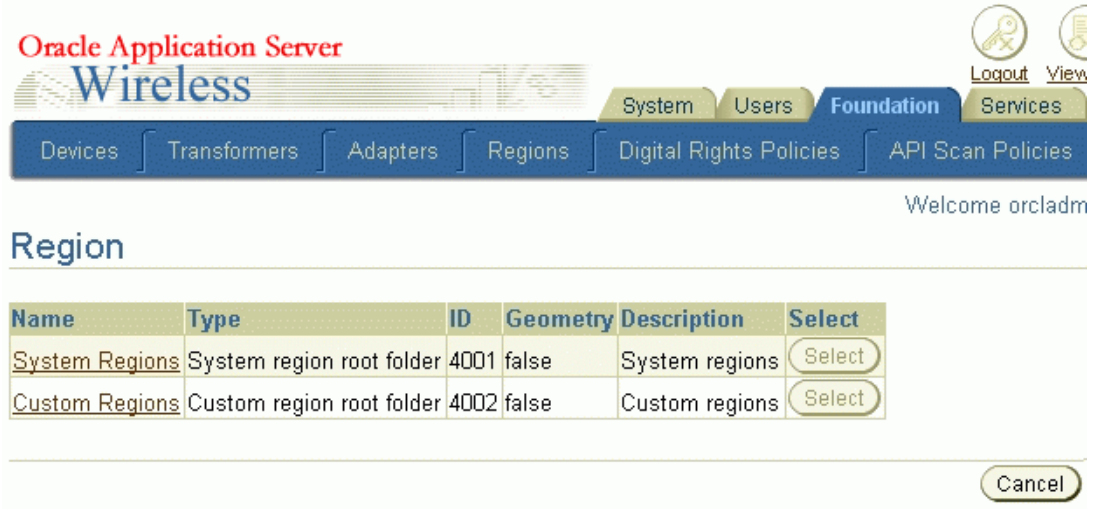


図 14-7 のように、リージョン・モデル・ツールが表示されます。

図 14-7 リージョン・モデル・ツールのインタフェース



Web ブラウザ・ウィンドウには、最初にリージョン階層のトップレベルとシステム・リージョンおよびカスタム・リージョンの 2 つのエントリが表示されます。リージョンを検索して、表示するリージョン、またはコレクションに追加してカスタム・リージョンの作成に使用するリージョンを選択できます。

システム・リージョンまたはカスタム・リージョンの表示の中でリージョンを検索するには、名前に含まれる文字列を入力して名前で検索する方法と、番号を入力して ID（リージョン ID）で検索する方法があります。また、選択対象としてすべてのリージョン、または現行の（現在選択している）リージョンに含まれるリージョンのみを指定し、**「実行」**をクリックします。

操作の実行対象となるリージョンを選択するには、アイコンと名前の左にあるボックスをクリックします。（項目を選択解除するには、ボックスをクリックします。）**「すべて選択」**をクリックすると、現在表示されているすべてのリージョンを選択でき、**「選択解除」**をクリックすると、現在表示されているすべてのリージョンを選択解除できます。

選択した 1 つ以上のリージョンに対して操作を実行するには、表 14-62 に示すコマンド・テキストのリンクまたはボタンをクリックします。

表 14-62 リージョン・モデル・ツールの操作

操作	クリックするリンクまたはボタン
選択したリージョンを表示の下部にあるリージョンのコレクションに追加する	**「コレクションに追加」**
選択したリージョンを示すマップを表示する	**「表示」**
表示の下部にあるリージョンのコレクションからカスタム・リージョンを作成する	**「コレクションから作成」**。一連のページが表示され、リージョン階層内での位置とカスタム・リージョン名を指定できます。
入力した番地からカスタム・リージョンを作成する	**「アドレスから作成」**。一連のページが表示され、住所、リージョン階層内での位置とカスタム・リージョン名を指定できます。
リージョンの編成に使用するフォルダを作成する	**「フォルダの作成」**。一連のページが表示され、リージョン階層内でフォルダを下に作成する位置とフォルダ名を指定できます。
リージョンまたは現行のコレクションの表示内で、前または次のエントリ・セットにジャンプする	**「前」**または**「次」**。

表 14-62 リージョン・モデル・ツールの操作（続き）

操作	クリックするリンクまたはボタン
リージョン階層内で上へ 1 レベル以上ジャンプする	ページ上部にある現行の階層の行で必要なレベルの名前。たとえば、この行（リンクである最後の項目を除くすべての項目）は、「リージョン」→「システム定義リージョン」→「NORTH AMERICA」→「USA」→「カリフォルニア」のように表示されます。
画面に関するヘルプを取得する	「ヘルプ」

リージョン・データのロードと更新

リージョン・モデル・ツールは、アメリカに関する広範囲なデータ・セットおよび多数の国のデータとともにインストールされます。ただし、リージョン・データが格納されている表に行を追加すると、他の国、州、都市などのデータを追加できます。たとえば、STATE 表にインドの州ごとに 1 行を追加できます。操作の意味を理解して慎重に行えば、これらの表内で特定のデータを変更することもできます。たとえば、特定の都市または州の DESCRIPTION 列を編集できます。

リージョン・データの表

リージョン・データは、表 14-63 に示す OracleAS Wireless リポジトリ内のテーブルに格納されます。

表 14-63 リージョン・データの表

表名	格納される情報
CONTINENT	大陸
COUNTRY	国
STATE	州
COUNTY	郡
CITY	都市
POSTALCODE	郵便番号
USERDEFINED	カスタム・リージョン

前述のいずれかの表の定義を表示するには、SQL 文 DESCRIBE を使用します。例 14-37 に、DESCRIBE 文の出力を示します。この出力には、すべての表に関する情報が含まれています。

例 14-37 リージョン・データ表の定義

```
SQL> DESCRIBE continent;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER
 NAME                                               VARCHAR2(100)
 REFCNT                                             NUMBER
 DESCRIPTION                                        VARCHAR2(2000)
 GEOMETRY                                           MDSYS.SDO_GEOMETRY

SQL> DESCRIBE country;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER
 NAME                                               VARCHAR2(300)
 REFCNT                                             NUMBER
 CONT_ID                                            NUMBER
 DESCRIPTION                                        VARCHAR2(2000)
 GEOMETRY                                           MDSYS.SDO_GEOMETRY

SQL> DESCRIBE state;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER
 NAME                                               VARCHAR2(400)
 REFCNT                                             NUMBER
 ABBR                                               VARCHAR2(32)
 CONT_ID                                            NUMBER
 COUNTRY_ID                                         NUMBER
 DESCRIPTION                                        VARCHAR2(2000)
 GEOMETRY                                           MDSYS.SDO_GEOMETRY

SQL> DESCRIBE county;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER
 NAME                                               VARCHAR2(400)
 REFCNT                                             NUMBER
 CONT_ID                                            NUMBER
 COUNTRY_ID                                         NUMBER
 STATE_ID                                           NUMBER
 DESCRIPTION                                        VARCHAR2(2000)
 GEOMETRY                                           MDSYS.SDO_GEOMETRY

SQL> DESCRIBE city;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
```

```
 ID                                        NOT NULL NUMBER
 NAME                                               VARCHAR2(400)
 REFCNT                                             NUMBER
 CONT_ID                                            NUMBER
 COUNTRY_ID                                         NUMBER
 STATE_ID                                           NUMBER
 DESCRIPTION                                        VARCHAR2(2000)
 GEOMETRY                                           MDSYS.SDO_GEOMETRY
 MIN_LON                                            NUMBER
 MIN_LAT                                            NUMBER
 MAX_LON                                            NUMBER
 MAX_LAT                                            NUMBER

SQL> DESCRIBE postalcode;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER
 NAME                                               VARCHAR2(400)
 REFCNT                                             NUMBER
 CONT_ID                                            NUMBER
 COUNTRY_ID                                         NUMBER
 STATE_ID                                           NUMBER
 DESCRIPTION                                        VARCHAR2(2000)
 GEOMETRY                                           MDSYS.SDO_GEOMETRY

SQL> DESCRIBE userdefined;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER
 NAME                                               VARCHAR2(200)
 REFCNT                                             NUMBER
 TYPE                                               NUMBER
 PARENT_FOLDER_ID                                   NUMBER
 DESCRIPTION                                        VARCHAR2(2000)
 GEOMETRY                                           MDSYS.SDO_GEOMETRY
```

リージョン表へのデータの挿入

SQL 文 INSERT を使用すると、リージョン表に新規の行を追加できます。リージョン・データの挿入時には、次の考慮事項が適用されます。

- 新規の行を挿入するたびに、例 14-38 のように `idseq.nextval` を使用して ID 列の値を生成する必要があります。idseq 順序はインストール中に自動的に作成されます。したがって、作成しないでください。
- 行の挿入時には、REFCNT 列を 0（ゼロ）に設定する必要があります。REFCNT 列には、リージョンに関連付けられているサービス数の参照カウントが含まれています。この値は、サービスがリージョンに関連付けられると自動的に増やされ、リージョンとの

関連付けが解除されるか、サービスが削除されると減らされます。REFCNT が 0（ゼロ）以外の値になっているリージョンは削除できません。

- デフォルトのリージョン階層を使用しない国について、郵便番号、都市または郡のデータを挿入する場合は、POSTALCODE、CITY または COUNTY 表で STATE_ID として 0（ゼロ）を指定します。
- GEOMETRY 列の値には、MDSYS.SDO_GEOMETRY 型の有効な Oracle Spatial ジオメトリを指定する必要があります。SDO_GTYPE には 4 桁の値を指定し、SRID（座標系）の値には 8307（WGS-84 経度 / 緯度書式の場合）を指定してください。SRID 値が現在は 8307 でない場合は、リージョン・データ表に挿入する前に、ジオメトリをその書式に変換する必要があります。空間データ型、座標系および座標系の変換の詳細は、『Oracle Spatial ユーザーズ・ガイドおよびリファレンス』を参照してください。

例 14-38 に、CITY 表にマサチューセッツ州コンコードの 1 行を挿入する INSERT 文を示します。この例は、コンコードを表すジオメトリが別の表 MY_CITIES に存在する場合を想定しています。

例 14-38 都市の挿入

```
DECLARE
  city_geom MDSYS.SDO_GEOMETRY;

BEGIN

-- Populate geometry variable with city geometry from another table.
SELECT m.geometry into city_geom FROM my_cities m
  WHERE m.name = 'Concord';

-- Insert into the CITY table.
INSERT INTO CITY VALUES (
   idseq.nextval,
   'Concord',
   0,
   5004, -- continent ID for North America
   5006, -- country ID for USA
   5028, -- state ID for Massachusetts
   'The historic town of Concord',
   city_geom,
   -71.35, -- minimum longitude
   42.46, -- minimum latitude
   -71.34, -- maximum longitude
   42.47); -- maximum latitude
END;
/
```

CITY 表内で、経度と緯度の最小値と最大値は必須であり、トラフィック・サポート・サービスで使用されます。経度と緯度の最小値では境界となる四角形の左下隅が識別され、経度と緯度の最大値では右上隅が識別されます。

Region Modeling API

リージョン・モデル・ツールは Region Modeling API に基づいています。この API は、`oracle.panama.spatial.region` パッケージの `RegionModel` インタフェースを介して実装されます。`RegionModel` インタフェースには、郵便番号、州および国を取得するメソッドと、リージョン間の各種の相互作用を判別するメソッドが含まれています。

外部コンテンツ・プロバイダの統合

OracleAS Wireless は、ジオコーディング、運転方向、ビジネス・ディレクトリ（イエロー・ページ）サービスおよびマッピングなど、多数のロケーション関連サービスへのアクセスをサポートしています。多くの場合、サービスを直接実行せず、外部プロバイダに依存しています。この項では、プロバイダが実装できる機能および OracleAS Wireless と通信するためのインタフェースのオプションについて説明します。

外部プロバイダは、各自が提供するロケーション・ベースのサービスのタイプごとにカスタム・サービス・プロキシを作成することによって、OracleAS Wireless と自社の製品を統合できます。**サービス・プロキシ**は、共通のインタフェースを実装する Java クラスで、各サービス（ジオコーディング、マッピング、ルーティング、トラフィック、イエロー・ページ）に 1 つのインタフェースがあります。サービス・プロキシは、プロバイダのサーバーではなく、OracleAS Wireless と同じシステム上で実行されます。

サービス・プロキシは、プロバイダのサーバーと、Oracle ロケーション・サービスが指定したインタフェースの間で変換する必要があります。また、必要な内部インフラストラクチャを提供するために、オラクル社が提供するクラスを拡張する必要があります。

サービスのタイプ（ジオコーディング、マッピングなど）に応じて、次のリストから適切なインタフェースを実装する必要があります。

```
oracle.panama.spatial.geocoder.Geocoder
oracle.panama.spatial.mapper.Mapper
oracle.panama.spatial.router.Router
oracle.panama.spatial.traffic.TrafficReporter
oracle.panama.spatial.yp.YPFinderSimple
```

インタフェース内には、実装必須、実装可能および実装禁止のファンクションがあります。詳細は、14-140 ページの「実装するファンクション」を参照してください。

サービスのタイプ（ジオコーディング、マッピングなど）に応じて、次のリストから適切なインタフェースを拡張する必要があります。

```
oracle.panama.spatial.core.geocoder.GeocoderImplXMLImpersonator
oracle.panama.spatial.core.mapper.MapperImplXMLImpersonator
oracle.panama.spatial.core.router.RouterImplXMLImpersonator
oracle.panama.spatial.core.traffic.TrafficReporterImplXMLImpersonator
oracle.panama.spatial.core.yp.YPFinderSimpleImplXMLImpersonator
```

たとえば、`oracle.panama.spatial.geocoder.Geocoder` を実装する場合は、次のインタフェースも拡張する必要があります。

```
oracle.panama.spatial.core.geocoder.GeocoderImplXMLImpersonator
```

ファイアウォール内から外部 URL へのアクセス

サービス・プロキシは、OracleAS Wireless サーバー上に配置されますが、このサーバーはファイアウォールの内側にある場合があります。これに対して、コンテンツ・プロバイダは通常、ファイアウォールの外側にあります。この場合、プロキシで OracleAS Wireless ファイアウォール・プロキシ設定（ロケーション・サービス・プロキシ設定とは異なります）を使用する必要があります。次の例は、ファイアウォール・プロキシの設定方法の抜粋です（重要な行は太字で表示されています）。

```
URL u = …

  try
  {
    URLConnection c = u.openConnection();
    ProxyFirewall.setProxyAuthorization(c);
    c.connect();
    BufferedReader
      bReader = new BufferedReader(
        new InputStreamReader(
          c.getInputStream()));
    ...
  }
  catch(...) { ... }
  ...
```

実装するファンクション

サービス・プロキシに使用可能なファンクションには、次の 3 つのカテゴリがあります。

- 実装必須 : 各プロキシの主要な部分として実装が必要なファンクション。
- 実装可能 : オプションで実装可能なファンクション（たとえば、自社の製品を競合品と差別化する機能を提供する場合）。
- 実装禁止 : フレームワークによって実装済で、プロキシでは実装禁止なファンクション。

使用可能な各ファンクションについては、各インタフェースの Javadoc ドキュメントを参照してください。

この項では、各タイプのサービス（ジオコーディング、マッピングなど）の各カテゴリに該当するファンクションを、クラス名を太字にして示します。ただし、サービスのタイプによっては、実装可能カテゴリに該当するファンクションがありません。

ジオコーディング・サービス : 使用可能なファンクション

ジオコーディング・サービス・プロキシでは、次のファンクションの実装が必須です。

```
public Location[] geocodeAddress(Location inp, String matchMode);
```

ジオコーディング・サービス・プロキシでは、次のファンクションの実装が可能です。

```
public Location[][] geocodeAddresses(Location[] inp, String matchMode);
public Location[] reverseGeocodePoint(Point pt);
```

ジオコーディング・サービス・プロキシでは、次のファンクションは実装禁止です。

```
public String xmlGeocode(Document xmlRequest);
```

マッピング・サービス : 使用可能なファンクション

マッピング・サービス・プロキシでは、次のファンクションの実装が必須です。

```
public String getMapURL(Point[] locations, ImageFormats fileType, double minLon,
double maxLon, double minLat, double maxLat, int width, int height, boolean
allowTurning);
```

マッピング・サービス・プロキシでは、次のファンクションは実装禁止です。

```
public String getMapURL(Point[] locations, ImageFormats fileType, int width, int
height, boolean allowTurning);
public String getMapURL(Point location, ImageFormats fileType, int width, int
height, boolean allowTurning);
public String getMapURL(RoutingResult route, boolean allowTurning);
public String getMapURL(Maneuver man, boolean allowTurning);
public String getMapURL(Point location, ImageFormats fileType, double minLon, double
maxLon, double minLat, double maxLat, int width, int height, boolean allowTurning);
```

```
public String[][] getMapURLs(Point[] locations, ImageFormats fileType, int width,
int height, int subdivisionLevel, boolean allowTurning);
public String[][] getMapURLs(Point[] locations, ImageFormats fileType, double
minLon, double maxLon, double minLat, double maxLat, int width, int height, int
subdivisionLevel, boolean allowTurning);
public String[][] getMapURLs(Point location, ImageFormats fileType, int width, int
height, int subdivisionLevel, boolean allowTurning);
public String[][] getMapURLs(Point location, ImageFormats fileType, double minLon,
double maxLon, double minLat, double maxLat, int width, int height, int
subdivisionLevel, boolean allowTurning);
public String[][] getMapURLs(RoutingResult route, int subdivisionLevel, boolean
allowTurning);
public String[][] getMapURLs(Maneuver man, int subdivisionLevel, boolean
allowTurning);
public String xmlMap(Document xmlRequest);
```

ルーティング・サービス : 使用可能なファンクション

ルーティング・サービス・プロキシでは、次のファンクションの実装が必須です。

```
public RoutingResult computeRoute(Point source, Point destination, Point[]
viaPoints, RoutingSettings opt, Locale locale);
```

ルーティング・サービス・プロキシでは、次のファンクションの実装が可能です。

```
public RoutingResult computeRoute(Location source, Location destination, Location[]
viaPoints, RoutingSettings opt, Locale locale);
public Ranking rankByDrivingDistance(Point source, Point[] locations);
```

ルーティング・サービス・プロキシでは、次のファンクションは実装禁止です。

```
public String xmlRoute(Document xmlRequest);
```

トラフィック・サービス : 使用可能なファンクション

トラフィック・サービス・プロキシでは、次のファンクションの実装が必須です。

```
public TrafficReport getReportViaCity(CityInfo city) throws LBSException;
public TrafficReport getReportViaLocation(Point location, double radius, int unit,
CityInfo city) throws LBSException;
public TrafficReport getReportViaRoute(RouteInfo route, CityInfo city) throws
LBSException;
public TrafficReport getReportViaRoute(RouteInfo route, String direction, CityInfo
city) throws LBSException;
```

トラフィック・サービス・プロキシでは、次のファンクションは実装禁止です。

```
public TrafficCityManager getCityManager();
public TrafficReport getReportViaLocation(Point location, double radius, int unit)
```

```
throws LBSException;
public TrafficReport getReportViaAddress(Location address, double radius, int unit)
throws LBSException;
public String xmlTraffic(Document xmlRequest) throws LBSException;
```

ビジネス・ディレクトリ（YP）サービス : 使用可能なファンクション

ビジネス・ディレクトリ（YP）サービス・プロキシでは、次のファンクションの実装が必須です。

```
public YPBusiness[] getBusinessesInCity(String businessName, String country, String
state, String city, Locale locale);
public YPBusiness[] getBusinessesInState(String businessName, String country, String
state, Locale locale);
```

ビジネス・ディレクトリ（YP）サービス・プロキシでは、次のファンクションの実装が可能です。

```
public Boolean anyBusinessesInCity(YPCategory category, String country, String
state, String city);
public Boolean anyBusinessesInState(YPCategory category, String country, String
state);
public Boolean anyBusinessesInPCode(YPCategory category, String country, String
postalCode);
public Boolean anyBusinessesInRadius(YPCategory category, Point location, double
metersRadius);
public YPBusiness[] getBusinessesInRadius(String businessName, Point location,
double metersRadius, Locale locale);
public YPBusiness[] getBusinessesInPCode(String businessName, String country, String
postalCode, Locale locale);
public YPBusiness[] getBusinessesInCity(YPCategory category, String country, String
state, String city, Locale locale);
public YPBusiness[] getBusinessesInState(YPCategory category, String country, String
state, Locale locale);
public YPBusiness[] getBusinessesInRadius(YPCategory category, Point location,
double metersRadius, Locale locale);
public YPBusiness[] getBusinessesInPCode(YPCategory category, String country, String
postalCode, Locale locale);
public YPBusiness[] getNearestNBusinesses(String businessName, Point location, int
n, Locale locale);
public YPBusiness[] getNearestNBusinesses(YPCategory category, Point location, int
n, Locale locale);
```

ビジネス・ディレクトリ（YP）サービス・プロキシでは、次のファンクションは実装禁止です。

```
public Boolean anyBusinessesInSameCity(YPCategory category, Location loc);
public Boolean anyBusinessesInSameState(YPCategory category, Location loc);
public Boolean anyBusinessesInSamePCode(YPCategory category, Location loc);
public YPBusiness[] getBusinessesInSameCity(String businessName, Location loc,
Locale locale);
public YPBusiness[] getBusinessesInSameState(String businessName, Location loc,
Locale locale);
public YPBusiness[] getBusinessesInSamePCode(String businessName, Location loc,
Locale locale);
public YPBusiness[] getBusinessesInSameCity(YPCategory category, Location loc,
Locale locale);
public YPBusiness[] getBusinessesInSameState(YPCategory category, Location loc,
Locale locale);
public YPBusiness[] getBusinessesInSamePCode(YPCategory category, Location loc,
Locale locale);
public YPBusiness[] getBusinessesInSameCity(String businessName, YPCategory
category, Location loc, Locale locale);
public YPBusiness[] getBusinessesInSameState(String businessName, YPCategory
category, Location loc, Locale locale);
public YPBusiness[] getBusinessesInSamePCode(String businessName, YPCategory
category, Location loc, Locale locale);
public YPBusiness[] getBusinessesInCity(String businessName, YPCategory category,
String country, String state, String city, Locale locale);
public YPBusiness[] getBusinessesInState(String businessName, YPCategory category,
String country, String state, Locale locale);
public YPBusiness[] getBusinessesInRadius(String businessName, YPCategory category,
Point location, double metersRadius, Locale locale);
public YPBusiness[] getBusinessesInPCode(String businessName, YPCategory category,
String country, String postalCode, Locale locale);
public YPBusiness[] getNearestNBusinesses(String businessName, YPCategory category,
Point location, int n, Locale locale);
public String xmlYP(Document xmlRequest);
```

モバイル・ポジショニング・プロバイダの統合

この項では、外部モバイル・ポジショニング・プロバイダを OracleAS Wireless に統合するための、サービス・プロキシの実装方法について説明します。モバイル・ポジショニング・プロバイダを統合する前に、次の事項を必ず理解してください。

- モバイル・ポジショニングの概要およびオプション（14-109 ページの「モバイル・ポジショニングの有効化」を参照）
- 外部プロバイダの概要（14-138 ページの「外部コンテンツ・プロバイダの統合」を参照）

モバイル・ポジショニングには、次のステップが含まれています。

1. ロケーションのプライバシ設定を調べ、ポジショニング・リクエストを許可するかどうかを判断します。
2. プロバイダ・セレクタ・フックを使用して、ユーザーのモバイル・ステーション ID を取得します（14-115 ページの「ポジショニング・プロバイダの指定」を参照）。システム・デフォルトでは、ユーザー・プロファイルのモバイル・ステーション ID のフィールドが使用されます。
3. ロケーション・キャッシュを調べます。ロケーション・キャッシュが使用可能で、キャッシュからリクエストが満たせる場合は、キャッシュで見つけたロケーションを戻し、ステップ 4 をスキップします。
4. キャッシュからロケーションが満たされない場合は、1 つ以上のモバイル・ポジショニング・プロキシを起動して、ユーザーの現在のロケーションを取得します。

前述のリストのステップ 4（1 つ以上のモバイル・ポジショニング・プロバイダを起動）には、次の固有の操作が含まれます。

1. システム構成ファイルのモバイル・ポジショニング・プロバイダに関する情報を調べます。この情報には、プロバイダ名、プロキシの実装クラス名、バージョン番号、モバイル・ポジショニング・サーバーの URL、モバイル・ポジショニング・ユーザーの名前とパスワードおよびすべての追加パラメータが含まれます。
2. プロキシのクラスをインスタンス化します。
3. そのクラスの `requestPosition()` method を起動して、モバイル・ステーションのロケーションを取得します。

モバイル・ポジショニング・プロキシの実装

モバイル・ポジショニング・プロバイダを統合するには、oracle.panama.mp.Positioner インタフェースを実装する必要があります。

クラスのコンストラクタは、次の書式にする必要があります。

```
public MyMPImpl (String  providerName,
                 String  providerImpl,
                 String  version,
                 String  url,
                 String  username,
                 String  password,
                 String  parameters);
```

モバイル・ポジショニング・フレームワークは、この情報をシステム構成から読み取り、プロキシ実装を構成するために使用します。クラスはその情報をクラス変数に格納し、後で使用します。

コンストラクタの他に、クラスは次のメソッドを実装する必要があります。

```
public PositionResult    requestPosition ( String   msid);
public PositionResult    requestPosition ( String   msid, PositionQoS qos);
public PositionResult[]  requestPosition ( String[] msids);
public PositionResult[]  requestPosition ( String[] msids, PositionQoS qos);
```

第 1 のメソッドは、モバイル・ステーション ID をパラメータとして取得します。

第 2 のメソッドは、モバイル・ステーション ID および位置の品質（PositionQoS）のパラメータを取得します。PositionQoS パラメータは、モバイル・ステーションのロケーションをキャッシュから供給できる最大許容経過時間を指定します。たとえば、アプリケーションによっては 5 分も経過したユーザーのロケーションを受け入れる可能性もあります。プロキシの実装では、ロケーションがすでに存在しているかどうかを判断するロジックがモバイル・ポジショニング・フレームワークに実装されているため、Oracle Application Server Wireless のロケーション・キャッシュを調べる必要はありません。つまり、システム・キャッシュ内にロケーションが存在し、リクエストを満たしている場合、プロキシは起動しません。プロキシで PositionQoS パラメータを考慮する必要があるのは、実際のモバイル・ポジショニング・プロバイダに同じキャッシュ概念があり、このパラメータを使用できる場合のみです。

第 3 と第 4 のメソッドは、それぞれ第 1 と第 2 のメソッドに似ていますが、複数のモバイル・ステーションのロケーションをリクエストするために使用されます。実際のモバイル・ポジショニング・プロバイダがバルク・リクエストを処理できる場合、プロキシはこの機能を使用して、1 回のコール（たとえば、FOR ループの使用）で複数のロケーションをリクエストできます。プロバイダが複数のリクエストを処理できない場合、プロキシはプロバイダをロケーションごとに 1 回、つまり複数回コールする必要があります。

各メソッドは、PositionResult オブジェクト、またはそのオブジェクトの配列を戻します。PositionResult オブジェクトは、モバイル・ステーションの現在のロケーションを

表します。PositionResult オブジェクトの詳細は、Javadoc ドキュメントの oracle.panama.mp.PositionResult クラスおよび oracle.panama.mp.PositionArea クラスを参照してください。

PositionResult オブジェクトの使用には、次のガイドラインが適用されます。

- モバイル・ポジショニング・プロバイダが、オブジェクトに 1 つ以上の値を戻さない場合は、それらの値を null に設定します。
- PositionResult オブジェクトには PositionArea オブジェクトの配列が含まれます。モバイル・ポジショニング・プロバイダが 1 つの PositionArea オブジェクトのみを戻す場合、その 1 つのオブジェクトも配列に含める必要があります。

oracle.panama.mp.Positioner インタフェースの実装では、例外およびエラーも処理する必要があります（詳細は、14-146 ページの「モバイル・ポジショニングでの例外およびエラーの処理」を参照）。

モバイル・ポジショニングでの例外およびエラーの処理

この項では、次のいずれかの場合に発生する、実行時のモバイル・ポジショニングのエラーおよび例外の処理に関するガイドラインを示します。

- プロバイダのレスポンスにエラー・コードおよびエラー・メッセージが含まれる場合。
- 解析中に例外がスローされた場合。
- 複数のモバイル・ステーションをポジショニングするリクエスト（14-145 ページの「モバイル・ポジショニング・プロキシの実装」で説明した第 3 および第 4 の requestPosition メソッドを使用）で、戻された結果の数とサブスクライバ ID の数が一致しない場合。

単一または複数のサブスクライバ ID に対するリクエストからのエラーまたは例外については、プロバイダからのエラー・コードおよびエラー・メッセージを調べます。

- そのエラーが重大なエラーである場合は、即時に null を戻してフェイルオーバーします。重大なエラーに含まれるのは、認証エラー、XML 解析中のエラー、およびプロキシ実装で発生したのではなく、リクエストの再送信で解決不可能なその他のエラーです。
- エラーが不明サブスクライバ ID を示す場合は、エラー ID UNKNOWNSUBSCRIBER およびエラー・メッセージ UNKNOWNSUBSCRIBER_STR によって、PositionResult オブジェクトを構成して戻します。エラー・メッセージは、PositionResult オブジェクトで getErrorMessage() メソッドを使用して、後で取り出せます。
- その他のエラー・コードについては、プロバイダからのエラー ID およびエラー・メッセージによって、PositionResult オブジェクトを構成して戻します。

15

ユーザー・カスタマイズの有効化

項ごとに様々なトピックを記載しています。各項の内容は、次のとおりです。

- ユーザー作業環境の概要
- 複数のカスタマイズ・プロファイル
- Presets
- ロケーション・マーク
- ユーザー・デバイスの管理
- ユーザーとグループの管理
- Service Management

ユーザー作業環境の概要

OracleAS Wireless には、ユーザー作業環境を管理するための、セキュアで信頼性の高いスケーラブルな機能が用意されています。ユーザー作業環境によって、ナビゲーションをパーソナライズしてモバイル・アプリケーションの効率を高める適応可能なアプリケーションを開発できます。また、この機能によって、コンテキストを認識した複数モーダルのマルチチャネル・アプリケーションを迅速に開発してデプロイできます。その結果、ユーザー操作が拡大し、一連の匿名トランザクションが継続的な 1 対 1 の顧客関係になります。

この章では、ユーザー作業環境を管理するための OracleAS Wireless の機能について説明し、高度なカスタマイズ機能を開発するためにユーザー作業環境を適用する手順をサンプルを使用して説明します。通常、カスタマイズは、ユーザーによるシステムの調整方法、つまりシステムをユーザーの特定のニーズや作業環境に適応させる方法を指します。ユーザー中心のカスタマイズ機能によって、ユーザーはシステムをニーズと作業環境にあわせて調整する方法を制御できます。このシステムでは、一般向けカスタマイズ・アプローチも導入できます。このアプローチでは、ユーザー・プロファイリング（ユーザーを類似のユーザー・グループに関連付けるなど）を適用し、ユーザーのニーズと作業環境を予測し、それに応じてシステムを調整します。

アプリケーションでカスタマイズを実行するには、ユーザーのニーズをそのロールと作業環境に基づいて理解する必要があります。たとえば、顧客、仕入先および従業員に、様々な方法であらかじめ情報を送信すると便利です。ユーザーの作業環境とニーズを十分に把握することによって、アプリケーションではユーザー操作を拡大できます。

自動ユーザー・プロファイリングを使用すると、一般向けカスタマイズ・アプローチを導入できます。ユーザーの使用履歴は、OracleAS Wireless リポジトリ内の `ptg_service_log` 表と `ptg_session_ log` 表で確認できます。この章の一部の例では、OracleAS Wireless Runtime を拡張して一般向けカスタマイズを導入する方法について説明します。OracleAS Wireless には、エンド・ユーザーが PC ブラウザから自分の作業環境を管理できるサンプル Customization Portal が組み込まれています。

OracleAS Wireless のサンプル Customization Portal は、Oracle Cabo UI XML（uix）および UI Bean を使用して開発されています。Customization Portal によって、エンド・ユーザーは、通常使用するユーザー・デバイス、フォルダ、アプリケーション（サービスとも呼ばれます）、ブックマーク、通知イベント（アラートとも呼ばれます）、通知アドレス、ロケーション・マーク、プリセットなどのユーザー作業環境をカスタマイズできます。さらに、ユーザーは、コラボレーション・サービスで使用する、連絡ルールとロケーションのプライバシ・ルールを指定できます。UIX コンポーネントとクラス・ライブラリは、特定の機能別にカテゴリ化されています。この UIX コンポーネントを再利用して、サンプル Customization Portal の特性を変更できます。また、複数の UIX コンポーネントを使用したり組み合せることによって、独自の Customization Portal を開発したり、既存のポータルにカスタマイズ・ウィザードを統合できます。

注意： Customization Portal のカスタマイズに UIX は不要です。任意の Web UI フレームワークを使用できます。

Customization Portal の開発時には、`Runtime API`、`Data Model API` およびサンプルの UIX コンポーネントをガイドラインとして参照してください。この章で説明する概念と例は、エンド・ユーザーの操作性を高めるカスタマイズ機能の設計に役立ちます。

Customization Portal の例では、ユーザーは、フォルダの編成、サービスのサブスクライブまたは取消し、1 つ以上のカスタマイズ・プロファイルでのブックマークとクイックリンクの作成または削除などを実行できます。ユーザーが実行できる操作は、次のとおりです。

- 新規のロケーション・マークの作成とそのジオコーディング。
- 個人情報または作業環境設定を含むプリセットの編集。
- ユーザー・デバイスの新規作成、およびデバイスのメーカーとモデルの指定。デバイス・メーカーとモデル名に基づいて、そのデバイス・リポジトリのデバイス機能が判別されます。
- デバイス・アドレスの指定、および非同期通知を受け取るためのアドレスの検証。ユーザーはデバイスの通知イベントをサブスクライブできます。

OracleAS Wireless に用意されているサンプル Device Customization Portal を使用すると、ユーザーは、Web 電話や PDA などのワイヤレス・デバイスからパーソナル・ポータルを直接カスタマイズできます。Device Customization Portal には、ワイヤレス・デバイスのセットアップ・ボタンからアクセスできます。このポータルは、デバイスの限定された表示機能や入力機能を調整し、ユーザーによるユーザー作業環境設定の変更、フォルダとサービス・リンクの編成、ロケーション・マーク、ブックマーク、クイックリンクおよびカスタマイズ・プロファイルの作成、変更または削除が可能な特殊なモードを提供します。

マルチ・ユーザー・カスタマイズ・プロファイルによって、ワイヤレス・デバイスからのナビゲーションが効率的になりパーソナライズされます。ユーザーは、1 つ以上のカスタマイズ・プロファイルを保持し、PC または Device Customization Portal から管理できます。また、デバイス内で異なるカスタマイズ・プロファイルに切り替えることができます。

OracleAS Wireless には、ユーザーが入力した値を将来の起動に備えてプリセット値として保存するかどうかを選択するオプションが用意されています。また、OracleAS Wireless には、プリセットを表す記号名を入力するオプションもあります。これらの記号名を使用すると、プリセット値のグループが複数ある場合に簡単に選択できます。また、ユーザーは任意のデバイスや PC からプリセットを管理できます。

Presets には、フィールドを一致させるのに必要な数の属性を Web フォームで追加できます。アプリケーション開発者は、Presets カテゴリを作成して Presets の属性を定義できます。各プリセット・カテゴリは、プリセット・カテゴリ名と必要な数のプリセット属性で構成されます。 たとえば、Auto-Fill Address Forms Fields という名前のプリセット・カテゴリを作成し、1 ～ 3 番目の属性として Street、City、ZIP を定義できます。また、ユーザーは、たとえばこのカテゴリに my home、my work、Mom's home という複数の Presets を作成できます。これらの Presets は、運転方向サービスに対するフォームの自動入力に使用できます。

OracleAS Wireless のユーザー作業環境には、連絡ルール、ロケーションのプライバシ・ルールと認可ルールが含まれます。連絡ルールはユーザー作業環境設定の 1 つで、コラボレーション・サービスとメッセージ・サービスで使用されます。連絡ルールは、ユーザーがコールやメッセージを受信する方法を示します。たとえば、ユーザーは会議の連絡ルールを設定する場合、すべての通知を携帯電話で受け取るように設定できます。また、複数の連絡ルールを定義し、それぞれ特定の状況にあわせて適切に設定できます。アクティブにできるのは、一度に 1 つの連絡ルールのみです。アクティブな連絡ルールによって、ユーザーが使用できるデバイス、およびユーザーへの通知方法を制御します。ロケーションのプライバシ・ルールと認可ルールは、ロケーション・ベースのコラボレーション・サービスで使用されます。ユーザーは、これらのロケーション・ルールを使用して、ユーザーのロケーションをロケーション・ベースのアプリケーションの対象にするかどうか、およびその時期を制御できます。また、ユーザーは、これらのルールを使用して、ユーザーのロケーションを問い合せるサービスまたはユーザーを許可できます。

注意： 連絡ルールの詳細は、『OracleAS Wireless 管理者ガイド』を参照してください。

複数のカスタマイズ・プロファイル

OracleAS Wireless では、コンテンツをデバイスとネットワークの機能のみでなく、エンド・ユーザーの作業環境にあわせて調整するユーザー中心の Web サービスを開発できます。通常、デバイス・ポータルにはサービス・メニューが用意されており、複数のフォルダとサブフォルダに編成できます。メニュー方式のデバイス・ポータルは、ワイヤレス・デバイスのナビゲーション効率を最適化するように設計されています。通常、サービス・メニューは静的ですが、ポータルではユーザーのニーズと作業環境に関する情報がより多く記憶されるため、ユーザーに対して新規サービスを提案できます。OracleAS Wireless サーバーでは、エンド・ユーザーはメニュー上のサービスの配置を制御して、ポータルをパーソナライズできます。ポータルはユーザーに新規サービスを提案できますが、各サービスを自分のパーソナライズ・ポータルに組み込む場合と除外する場合を制御するのはユーザーです。管理者は、特定のサービス（プロモーション、優先パートナ、緊急サービスなど）をパーソナライズ・ポータルで再配置または削除する操作を、エンド・ユーザーに対して明示的に禁止できます。

概要

Profile を使用すると、ユーザーはポータルのパーソナライズ・バージョンをデバイス用に複数作成できます。サービス・メニューは、プロファイルごとに異なる場合があります。たとえば、ユーザー用フォルダの 1 つに次のサービスが含まれているとします。

- 電子メール
- ニュース
- 株式
- マップ
- 電話帳
- ショッピング

Home プロファイルでは、フォルダのサービス・メニューを次のようにカスタマイズできます。

- 電話帳
- 電子メール
- ニュース
- 株式

同じフォルダを、Traveling プロファイルでは次のようにカスタマイズできます。

- マップ
- ショッピング
- 電話帳

様々なロール、ロケーションまたはコンテキスト、デバイスとネットワークの特性その他の観点について、複数の Profile を作成できます。

これらのロールごとに Profile を作成すると、サービスの効率とアクセス可能性が向上します。複数の都市を頻繁に訪れる移動型ユーザーの場合は、ロケーションごとにプロファイルを作成できます。たとえば、ユーザーが、文化の中心地（イタリアのローマなど）のカスタマイズ Profile に、劇場、美術館および交通手段の時刻表のサービスを含めるとします。この同じユーザーは、バイカル湖地域について、異なるサービスを組み合せて別の Profile を使用できます。ロケーション認識ポータルでは、ユーザーが異なるロケーションから接続した場合に、そのユーザーのセッション Profile を自動的に設定できます。Profile はロケーション・マークと関連付けることができます。ロケーション・マークについては、「ロケーション・マーク」の項を参照してください。

OracleAS Wireless Runtime controller は、ユーザーの Profile を自動的にプロビジョニングするように拡張できます。これによって、たとえば複数タイプのデバイスからポータルの異なるビューを提供できます。ユーザー用のプロファイルを自動的にプロビジョニングする方法は、次の項の例を参照してください。また、エンド・ユーザーは、PC ブラウザ経由で Customization Portal を介して、任意のコンテキストの Profile を必要な数だけ作成できます。OracleAS Wireless Tools を使用すると、Profile ごとにサービスの配置をカスタマイズできます。

管理者は、共有フォルダのデフォルトのソート規則を指定できます。Profile アーキテクチャでは、エンド・ユーザーはデフォルトのソート規則を変更し、共有フォルダの独自のビューをパーソナライズできます。この場合、次のソート規則から選択できます。

- 指定した順序番号
- 五十音順
- 作成日
- アクセス頻度
- 最新アクセス時間

順序番号、五十音順および作成日を選択すると、フォルダの静的ビューが生成されます。アクセス頻度順または最新アクセス時間順にソートすると、フォルダの動的ビューが生成されます。さらに、管理者は、緊急、プロモーションおよび優先パートナのサービスなど、フォルダ内でエンド・ユーザーが再配置できない一部のアプリケーションの静的または動的な配置を制御できます。また、管理者はエンド・ユーザーが再配置または非表示にできるビューのセグメントを指定できます。

フォルダのビューをセグメントに分割し、あるセグメントは管理者の指定に従ってソートされ、別のセグメントはユーザーの指定に従ってソートされるように指定したりできます。

Profile のアーキテクチャによって、管理者がアプリケーションのパーソナライズ可能な属性を明示的に使用禁止にしていない場合、エンド・ユーザーはプロファイル内のサービスの可視性を指定できます。これによって、エンド・ユーザーは、システムによってフォルダに置かれたアプリケーションへのサブスクライブとサブスクライブの取消しができます。また、システムでは、ロケーション対応フォルダ内のサービスにロケーション・ベースのフィルタ

処理を適用し、ユーザーのモバイル位置に応じて変化するビューの動的性質を高めることができます。

Runtime オブジェクトにアクセスするアプリケーションは、`ServiceContext.getProfile` メソッドから現行の Profile を取得できます。Runtime オブジェクトの詳細は、9-33 ページの「MCS Runtime API」を参照してください。このメソッドは、最初に現行の Request 内で Profile を検索します。現行の Request で Profile が指定されていない場合、このメソッドは Runtime Session 内でセッションの Profile を検索します。セッションの Profile が空の場合は、ユーザーのデフォルト Profile が検索されます。この解消方法にって、Request でセッション Profile をオーバーライドし、セッションでデフォルトのユーザー・プロファイルをオーバーライドできます。Profile が存在しない場合、`ServiceContext.getProfile` は `null` を戻します。アプリケーションは、Profile が未指定の場合にデフォルト動作で反応するように設計する必要があります。

サンプル・アプリケーション

次の例に、ユーザー・デバイスを自動的にプロビジョニングする方法、および各デバイス・タイプに対してユーザーが使用できるデバイスの Profile を示します。`SampleRequestListener` は `serviceBegin()` イベントをリスニングし、**17** 行目と **19** 行目に Request と Session による Profile の指定がない場合は、**22** 行目、**25** 行目および **27** 行目で新規のユーザー・デバイスと Profile をプロビジョニングします。新規 Profile について、**36** 行目でユーザーのホーム・フォルダが、中のサービスが最新アクセス時間順にソートされるように設定されます。**39** 行目でビューをカスタマイズ可能なサービスごとに、**40** 行目でサービスが Profile 内で非表示になるように設定されます。エンド・ユーザーは、使用するサービスが表示されるように後で Profile をカスタマイズできます。この操作は、Profile を初めて作成した後に一度のみ行います。リスナーは、**54** 行目で Request 内で Profile を設定します。

```
import oracle.panama.model.*;
import oracle.panama.rt.Session;
import oracle.panama.rt.Request;
import oracle.panama.rt.event.RequestAdapter;
import oracle.panama.rt.event.RequestEvent;
import oracle.panama.rt.event.AbortServiceException;
import oracle.panama.PanamaException;

public class SampleRequestListener extends RequestAdapter {

    public void serviceBegin(RequestEvent event) throws AbortServiceException {
        Request request = event.getRequest();
        Session session = request.getSession();
        User user = session.getUser();

        Profile profile = request.getProfile();         //[17]

        if (profile == null)
            profile = session.getProfile();               //[19]
```

```
if (profile == null) {
    Device device = request.getDevice();
    String deviceName = device.getName();
    Profile deviceProfile;
    synchronized(user) {
        ModelFactory factory = MetaLocator.getInstance().getModelFactory();
        UserDevice userDevice = user.lookupUserDevice(deviceName);
        boolean deviceProfileCreated = false;
        if (userDevice == null) {
            userDevice = user.createUserDevice();    //[22]
            userDevice.setName(deviceName);
            userDevice.setDisplayName("My user device name for " + deviceName);
            try {
                factory.save();
                deviceProfileCreated = true;
            } catch (PanamaException ex) {
                deviceProfile = null;
            }
        }
        deviceProfile = userDevice.getUserProfile();
        if (deviceProfile == null) {
            deviceProfile = user.lookupProfile(deviceName);         //[25]
            if (deviceProfile == null) {
                deviceProfile = user.createProfile(deviceName);      //[27]
            }
            try {
                factory.save();
                deviceProfileCreated = true;
            } catch (PanamaException ex) {
                deviceProfile = null;
            }
        }

        if (deviceProfileCreated) {
            boolean needCommit = false;
            Folder home = user.getHomeFolder();
            deviceProfile.setSortRule(home, SortRule.SORT_BY_ACCESS_TIME_ASCEND);    //[36]
            Service[] services = home.getAccessibleUserServices(user);
            for (int i = 0; i < services.length; i++) {
                if (services[i].isViewCustomizable()) {           //[39]
                    deviceProfile.setHide(services[i], true);      //[40]
                    needCommit = true;
                }
            }
            try {
                if (needCommit)
                    factory.save();
            } catch (PanamaException ex) {

            }
```

```
            }
        }
                }
    if (deviceProfile != null)
        request.setProfile(deviceProfile);                    //[54]

}
```

Presets

OracleAS Wireless には、開発者とエンド・ユーザーのために、広範囲なカスタマイズを適用してパーソナライズされたポータルを作成するための機能が用意されています。これによって、ポータルと各エンド・ユーザーとの 1 対 1 の関係が強化されます。主要な機能の 1 つは、ユーザーの個人情報、作業環境の設定、頻繁に使用する入力パラメータをサーバー側に格納するための Presets です。アプリケーションは Presets を使用してパーソナライズされたレスポンスを生成できます。

OracleAS Wireless リポジトリには、永続属性が定義されているポータル・ユーザーの概念が含まれています。この基本的な属性には、名前、性別、生年月日、国、言語、ロケールなどがあります。Presets は OracleAS Wireless リポジトリ内の永続オブジェクトであり、特にリポジトリにユーザー・オブジェクト用に新規の永続属性を取り込むなど、リポジトリ・スキーマの拡張に使用できます。

Presets の概念とアーキテクチャ

Presets は OracleAS Wireless リポジトリ内の永続オブジェクトです。このオブジェクトを使用すると、ユーザー・スキーマを拡張し、ユーザーの個人情報をリポジトリに取り込むことができます。アプリケーションの開発者は、プリセット・カテゴリを定義してユーザー・スキーマをアプリケーション固有の方法で拡張できます。たとえば、請求先住所、クレジット・カード引落し口座、銀行口座、手数料引落し口座、株式のポートフォリオ、緊急時の連絡先などを取り込むことができます。このように拡張されたスキーマは、Personal Information Management（PIM）サービスで定義し、排他的にメンテナンスできます。

Presets を使用すると、リポジトリにユーザー作業環境を取り込むこともできます。リポジトリ内の標準的なユーザー・エージェント・タイプとデバイス・モデルによって、デバイスの機能が記述されます。個々のエンド・ユーザーは、ユーザー・エージェントの機能の一部をカスタマイズできます。ユーザー・エージェント・プロファイルの Presets を使用すると、エンド・ユーザーは、サウンドの有効化または無効化、背景色の選択、サービス品質の選択、イメージを使用禁止にしてパケット送信を最小限に抑えるなど、ユーザー・エージェントの機能をカスタマイズできます。ユーザー・エージェント・プロファイルによってコンテンツのフォーマットが制御されますが、一般的なユーザー作業環境プロファイルをアプリケーションの選択とアプリケーションのレスポンスに反映させることができます。たとえば、アプリケーションでは、スポーツ、エンタテインメント、テクノロジ、プライバシ要件などに関するユーザー作業環境プロファイルを使用して、コンテンツをフィルタ処理できます。Presets アーキテクチャでは、新しい Composite Capability/Preference Profile（CC/PP）、ユーザー・エージェント・プロファイル（WAP UAProf）および Platform for Privacy

Preferences（P3P）規格（www.wapforum.org）に基づいて、適応 Web サービスを開発できます。

また、Presets には、アプリケーションに頻繁に使用される入力パラメータも格納できます。アプリケーションでは、Presets の属性をアプリケーションで使用されるフォームに近似となるように定義できます。これらの Presets は、フォームへの自動入力に使用できます。アプリケーションには、ユーザー入力を後で使用できるように Presets として格納できます。Presets 名では入力パラメータ値が一意に識別され、この名前をショートカットとして使用するとデータ入力量を大幅に削減できます。

リポジトリには、様々なカテゴリの Presets があります。各 Presets リレーションにはプリセット属性値のセットが含まれており、属性値の型とリレーションはプリセット・カテゴリで定義されています。ユーザーは、各プリセット・カテゴリ内で 1 つ以上の Presets リレーションを所有できます。各プリセット・カテゴリにはプリセット記述子のコレクションが含まれており、Presets リレーションの属性のメタ情報を提供します。属性のメタデータは、属性の名前、型、サイズ、フォーマットおよび記述などです。たとえば、住所録プリセット・カテゴリの Presets リレーションには、ユーザーの連絡先の名前、住所および電話番号の属性を含めることができます。このようなプリセット・カテゴリは、Personal Information Management（PIM）アプリケーションで定義し、排他的にメンテナンスできます。別のプリセット・カテゴリでは、ユーザーがリストやポートフォリオを注視している企業の銘柄コード、名前および分類を含む Presets リレーションの属性を定義できます。このカテゴリの銘柄コードを、株価サービスの入力パラメータとして使用できます。

プリセット・カテゴリには、リポジトリ内で一意の名前を使用する必要があります。同様に、プリセット記述子には、属しているプリセット・カテゴリ内で一意の名前を使用してください。Presets リレーション名はオプションですが、指定する場合は、同じプリセット・カテゴリ内で同じユーザーが所有している Presets リレーション間で一意の名前にする必要があります。プログラムによって作成されたプリセット・カテゴリは、デフォルトでシステムとしてマークされます。つまり、アプリケーションによって排他的にメンテナンスされます。システム・レベルのプリセット・カテゴリは、Customization Portal 内では参照できず、エンド・ユーザーは直接編集できません。アプリケーションでは、プリセット・カテゴリを非システムに設定できます。これによって、その Presets をエンド・ユーザーが Customization Portal 内で編集できます。

サンプル・アプリケーション

プリセット・カテゴリは、次の例のようにプログラムによって作成できます。また、OracleAS Wireless Tools の「プリセット定義」コントロール・パネルで作成する方法もあります。

例 1: ユーザー・スキーマへの属性の追加

次のコード例に、プリセット・カテゴリ Billing Address を作成してユーザー・スキーマを拡張する方法を示します。このメソッドは、最初に **13** 行目で、Billing Address カテゴリがリポジトリにすでに存在しているかどうかをチェックします。このカテゴリが存在しない場合は、**21** 行目で ModelFactory のメソッド `createPresetCategory(Billing Address)` を使用して作成されます。**23** ～ **27** 行目では、カテゴリの最初の属性 Addressee Name が定義されます。**25** 行目では、最初の属性を 1 行のテキストで構成するように定義されます。これに対して、第 2 の属性 Street Address は、**31** 行目で複数行のテキスト・フィールドとして定義されます。新規のプリセット・カテゴリは、**47** 行目でコミットされます。

```
import oracle.panama.model.ModelFactory;
import oracle.panama.model.PresetCategory;
import oracle.panama.model.PresetDescriptor;
import oracle.panama.ArgumentType;
import oracle.panama.PanamaException;

public void createAddressBook() throws PanamaException {

        ModelFactory factory = MetaLocator.getInstance().getModelFactory();
        ModelServices services = MetaLocator.getInstance().getModelServices();


        PresetCategory category;
        try {
            category = services.lookupPresetCategory("Billing Address");    [13]
        } catch (PanamaRuntimeException ex) {
            category = null;
        }

        if (category != null) {
            return;   // category already exists
        }
         category = factory.createPresetCategory("Billing Address");        [21]

        PresetDescriptor descriptor = category.createPresetDescriptor("Addressee Name");   [23]
        descriptor.setDescription("The name of the addressee");
        descriptor.setPresetType(ArgumentType.SINGLE_LINE);   [25]
        descriptor.setStoredType(Java.sql.Types.VARCHAR);
        descriptor.setSize(new Long(40));[27]
```

```
    descriptor = category.createPresetDescriptor("Street Address");
    descriptor.setDescription("The street address");
    descriptor.setPresetType(ArgumentType.MULTI_LINE);    [31]
    descriptor.setStoredType(Java.sql.Types.VARCHAR);
    descriptor.setSize(new Long(120));

    descriptor = category.createPresetDescriptor("State");
    descriptor.setDescription("The name of the state");
    descriptor.setPresetType(ArgumentType.SINGLE_LINE);
    descriptor.setStoredType(Java.sql.Types.VARCHAR);
    descriptor.setSize(new Long(2));

    descriptor = category.createPresetDescriptor("Zip code");
    descriptor.setDescription("The postal zip code");
    descriptor.setPresetType(ArgumentType.SINGLE_LINE);
    descriptor.setStoredType(Java.sql.Types.NUMERIC);
    descriptor.setSize(new Long(5));

    factory.save();   [47]
}
```

プリセット・カテゴリには、リポジトリ内で一意の名前を使用する必要があります。アプリケーションが同じ名前でプリセット・カテゴリの作成を試行すると、ModelFactory の `createPresetCategory()` メソッドが `oracle.panama.model.NameUniquenessViolationException` をスローします。同様に、プリセット記述子には、プリセット・カテゴリ内で一意の名前を使用する必要があります。アプリケーションが同じ名前でプリセット記述子の作成を試行すると、PresetCategory の `createPresetDescriptor()` メソッドが `oracle.panama.model.NameUniquenessViolationException` をスローします。プリセット・カテゴリとプリセット記述子の名前には大 / 小文字区別があり、空白など、有効な任意の文字を使用できます。

例 2: ユーザーの一意の Presets リレーションの追加

次のコード例に、プリセット・カテゴリ Billing Address を使用してユーザーに永続属性を追加する方法を示します。Billing Address カテゴリが存在しない場合、このメソッドは新規カテゴリを作成します。この例では、Presets 名としてユーザーの一意のオブジェクト ID を使用しています。新規の Presets リレーションは、**13** 行目の参照メソッドで同じ名前を持つ既存の Presets リレーションが見つからない場合にのみ、**16** 行目で作成されます。Presets 名としてオブジェクト ID を使用すると、各ユーザー用に Billing Address の Presets リレーションのインスタンスが確実に 1 つのみ作成されます。Presets リレーションの属性値は、**18** ～ **21** 行目で変更されます。変更された Presets リレーションは、**23** 行目でリポジトリにコミットされます。

```
import oracle.panama.model.*;

public void addBillingAddress(User user, String addressee, String streetAddress,
                              String state, int zipCode) throws PanamaException {
        ModelFactory factory = MetaLocator.getInstance().getModelFactory();
        ModelServices services = MetaLocator.getInstance().getModelServices();

        PresetCategory category;
        try {
            category = services.lookupPresetCategory("Billing Address");
        } catch (PanamaRuntimeException ex) {
            createAddressBook();[9]
            category = services.lookupPresetCategory("Billing Address");
        }

        Presets presets = user.getPresets(category, Long.toString(user.getId()));[13]

        if (presets == null) {
            presets = user.createPresets(category, Long.toString(user.getId()));    [16]
        }

        presets.setPresetValue("Addressee Name", addressee);    [18]
        presets.setPresetValue("Street Address", streetAddress);
        presets.setPresetValue("State", state);
        presets.setPresetValue("Zip code", Integer.toString(zipCode));     [21]


        factory.save();   [23]
    }
```

Presets リレーションには、ユーザーのドメイン内で一意の名前を使用する必要があります。アプリケーションが同じユーザー用に同じ名前で Presets の作成を試行すると、ユーザーの `createPresets()` メソッドが `oracle.panama.model.NameUniquenessViolationException` をスローします。Presets リレーション名には大 / 小文字区別があり、空白など、有効な任意の文字を使用できます。

例 3: ユーザーの Profile 用の一意の Presets リレーションの追加

Profile は、ユーザーごとにパーソナライズされたポータルの複数バージョンをサポートするリポジトリ・オブジェクトです。ユーザーが Business 用と Personal 用に 1 つずつ Profile を持ち、Profile ごとに別個のクレジット・カード引落し口座が必要であるとします。次のコード例に、Credit Card Charge Account カテゴリと、ユーザーの各プロファイルに対して一意の Presets リレーションを作成する方法を示します。プロファイルごとに 1 つのみ Presets リレーションが作成されるように、**43** 行目で User および Profile のオブジェクト ID から一意の Presets 名が作成されます。この例は、Card Type 属性にプリセット・タイプ

ArgumentType.ENUM を使用する方法も示しています。ENUM タイプを使用すると、**28** および **30** 行目のように、その属性に有効なオプションを指定できます。**60** ～ **62** 行目は、永続記憶域に Java.sql.Date タイプを使用する方法を示しています。クレジット・カードの有効期限は、解析してリポジトリに Date 型として格納できるように、**61** 行目で Java.text.DateFormat ユーティリティを使用してフォーマットされます。

```
import oracle.panama.model.*;
import Java.util.Date;
import Java.text.DateFormat;

public void addCreditAccount(User user, Profile profile, String cardNumber,
                             String cardType, int expireMonth, int expireYear)
                             throws PanamaException {
        ModelFactory factory = MetaLocator.getInstance().getModelFactory();
        ModelServices services = MetaLocator.getInstance().getModelServices();

        PresetCategory category;
        try {
            category = services.lookupPresetCategory("Credit Card Charge Account");
        } catch (PanamaRuntimeException ex1) {
            try {
                category = factory.createPresetCategory("Credit Card Charge Account");
            } catch (PanamaException ex2) {
                throw ex2;
            }


            PresetDescriptor descriptor = category.createPresetDescriptor("Account Number");
            descriptor.setDescription("The credit card account number");
            descriptor.setPresetType(ArgumentType.SINGLE_LINE);
            descriptor.setStoredType(Java.sql.Types.VARCHAR);
            descriptor.setSize(new Long(40));

            descriptor = category.createPresetDescriptor("Card Type");
            descriptor.setDescription("The type of credit card");
            descriptor.setPresetType(ArgumentType.ENUM);     [25]
            descriptor.setStoredType(Java.sql.Types.VARCHAR);
            descriptor.setSize(new Long(40));
           String cardTypes[] = { "Master", "Visa", "Discover", "American Express", "Diners Club" };
[28]
            try {
                descriptor.setOptions(cardTypes);      [30]
            } catch (TooManyOptionsException ex3) {
                throw new PanamaException(ex3);
            }

            descriptor = category.createPresetDescriptor("Expiration Date");
```

```
            descriptor.setDescription("The expiration date of the credit card");
            descriptor.setPresetType(ArgumentType.SINGLE_LINE);
            descriptor.setStoredType(Java.sql.Types.DATE);

            factory.save();      [40]
        }

        String presetsName = Long.toString(user.getId()) + "-" + Long.toString(profile.getId()); [43]
        Presets presets = profile.getPresets(category, presetsName);
        if (presets == null) {
            presets = profile.createPresets(category, presetsName);
        }

        presets.setPresetValue("Account Number", cardNumber);
        presets.setPresetValue("Card Type", cardType);
        Date date = new Date(expireYear, expireMonth, 1);   [60]
        String dateStr = DateFormat.getInstance().format(date);    [61]
        presets.setPresetValue("Expiration Date", dateStr);    [62]

        factory.save();     [64]
    }
```

例 4: 現行の Profile での Presets リレーションの選択

リクエスト・リスナーから抜粋した次のコード例に、Credit Card Charge Account の Presets リレーションが `serviceBegin()` イベント通知中にアクセスされる方法を示します。ユーザーに使用可能な有効なクレジット・カード引落し口座がない場合、このルーチンは `AbortServiceException` をスローします。また、**14** および **16** 行目のように、リクエスト・プロファイル、セッション・プロファイルまたはデフォルトのユーザー・プロファイルを順番にチェックします。Presets 名は、User と Profile のオブジェクト ID で構成されます。Credit Card Charge Account の Presets リレーションが見つかると、リスナーは **51** ～ **52** 行目でクレジット・カード情報をサービスにリクエスト・パラメータとして提供します。

```
import oracle.panama.rt.event.RequestEvent;
import oracle.panama.rt.event.AbortServiceException;
import oracle.panama.rt.Session;
import oracle.panama.rt.Request;

public void serviceBegin(RequestEvent event) throws AbortServiceException {
        Request request = event.getRequest();
        PresetCategory category;
        String presetsName;
        ModelServices services = MetaLocator.getInstance().getModelServices();
        String serviceName = request.getServicePath();
        User user;

        Profile profile = request.getProfile();      [14]
```

```
        if (profile == null) {
            profile = request.getSession().getProfile();     [16]
        }
        if (profile != null) {
            user = profile.getUser();
            presetsName = Long.toString(user.getId()) + "-" + Long.toString(profile.getId());
        } else {
            user = request.getSession().getUser();
            presetsName = Long.toString(user.getId());
        }

        try {
            category = services.lookupPresetCategory("Credit Card Charge Account");
        } catch (PanamaRuntimeException ex1) {
            throw new AbortServiceException("This service " + serviceName + " requires a valid charge
account");
        }

        Presets presets = null;
        if (profile == null) {
            presets = user.getPresets(category, presetsName);
        } else {
            presets = profile.getPresets(category, presetsName);
            if (presets == null) {
                presets = user.getPresets(category, presetsName);
            }
        }

        if (presets == null) {

            throw new AbortServiceException("This service " + serviceName + " requires a valid charge
account");
        }

        String creditCardNumber;
        String cardType;
        String expiration;
        try {
            creditCardNumber = presets.getPresetValue("Account Number");
            cardType = presets.getPresetValue("Card Type");
            expiration = presets.getPresetValue("Expiration Date");
        } catch (PanamaException ex) {
            throw new AbortServiceException("This service " + serviceName + " requires a valid charge
account");
        }

        if (! creditAvailable(creditCardNumber, cardType, expiration)) {
```

```
            throw new AbortServiceException("This service " + serviceName + " requires a valid charge
account");
        }

        request.setParameter("Account Number", creditCardNumber);     [51]
        request.setParameter("Card Type", cardType);      [52]
        request.setParameter("Expiration Date", expiration);    [53]
    }
```

前述の例は、アプリケーションで Presets リレーションに予約済のネーミング規則を使用する必要があるが、Presets 名自体はオプションであるという使用例に基づいています。次の例に、複数セットのエントリを使用できるプリセット・カテゴリ Appointments を示します。Presets リレーションの識別は、その属性の 1 つで提供されます。この例では、Presets は名前なしで作成されます。

例 5: 名前指定なしの Presets の作成

次のコード例に、ユーザーに予定イベントを作成させるプリセット・カテゴリ Appointments を示します。属性 Short Title を使用してイベントを識別するため、**65** 行目のようにイベント Presets は名前なしで作成されます。ユーザー用のすべてのイベント Presets は、**97** 行目のようにリポジトリから取り出すことができます。Appointments カテゴリは、**25** 行目で非システムに設定されているため、このカテゴリはエンド・ユーザーが編集できるように Customization Portal に組み込むことができます。この例は、`DateFormat` ユーティリティを使用して、イベントの時刻を **69** 行目で保存し、**105** 行目で取り出す方法を示しています。期限切れになったイベントは、**112** 行目でリポジトリから削除されます。また、この例は、正規表現を使用して Phone Number 属性のフォーマットに制約を適用する方法も示しています。この正規表現には、パブリック・ドメイン `org.apache.regexp.RE` のツールセットとの互換性があります。**59** 行目の正規表現は、次のように US ロケールの電話番号に関するものです。

```
"¥s*[(]?[1-9]¥d{2}[)]?¥s*-?¥s*¥d{3}¥s*-?¥s*¥d{4}"
```

この正規表現にはエスケープ文字がありません。**77** 行目の `setPresetValue()` メソッドは、値が正規表現と一致しないと、Presets 内で `PanamaException` をスローします。`org.apache.regexp.RE` ツールセットと互換性のある正規表現の構文の詳細は、次の項を参照してください。

```
import oracle.panama.model.*;
import oracle.panama.PanamaException;
import oracle.panama.PanamaRuntimeException;
import oracle.panama.ArgumentType;

import Java.util.Vector;
import Java.util.Enumeration;
import Java.util.Date;
import Java.text.DateFormat;
import Java.text.ParseException;
```

```
public class SamplePresets {

    public void addAppointment(User user, String title, String memo, Date time,
                               boolean alarm, String phone) throws PanamaException {
        ModelFactory factory = MetaLocator.getInstance().getModelFactory();
        ModelServices services = MetaLocator.getInstance().getModelServices();

        PresetCategory category;
        try {
            category = services.lookupPresetCategory("Appointments");
        } catch (PanamaRuntimeException ex1) {
            try {
                category = factory.createPresetCategory("Appointments");
                category.setSystem(false);          [25]
            } catch (PanamaException ex2) {
                throw ex2;
            }

            PresetDescriptor descriptor = category.createPresetDescriptor("Short Title");
            descriptor.setDescription("Brief description of the event");
            descriptor.setPresetType(ArgumentType.SINGLE_LINE);

            descriptor.setStoredType(Java.sql.Types.VARCHAR);
            descriptor.setSize(new Long(40));

            descriptor = category.createPresetDescriptor("Memo");
            descriptor.setDescription("Memo for the event");
            descriptor.setPresetType(ArgumentType.MULTI_LINE);
            descriptor.setStoredType(Java.sql.Types.VARCHAR);
            descriptor.setSize(new Long(400));

            descriptor = category.createPresetDescriptor("Time");
            descriptor.setDescription("Time of event");
            descriptor.setPresetType(ArgumentType.SINGLE_LINE);
            descriptor.setStoredType(Java.sql.Types.VARCHAR);
            descriptor.setSize(new Long(40));

            descriptor = category.createPresetDescriptor("Alarm");
            descriptor.setDescription("Enable or disable alarm before event");
            descriptor.setPresetType(ArgumentType.SINGLE_LINE);
            descriptor.setStoredType(Java.sql.Types.VARCHAR);
            descriptor.setSize(new Long(1));

            descriptor = category.createPresetDescriptor("Phone Number");
            descriptor.setDescription("Optional phone number to ring for alarm");
            descriptor.setPresetType(ArgumentType.SINGLE_LINE);
```

```
        descriptor.setStoredType(Java.sql.Types.VARCHAR);
        descriptor.setSize(new Long(40));
        descriptor.setFormat("¥¥s*[(]?[1-9]¥¥d{2}[)]?¥¥s*-?¥¥s*¥¥d{3}¥¥s*-?¥¥s*¥¥d{4}");      [59]
        descriptor.setEmptyOK(true);

        factory.save();
    }

    Presets presets = user.createPresets(category);    [65]

    presets.setPresetValue("Short Title", title);
    presets.setPresetValue("Memo", memo);
    String timeStr = DateFormat.getDateTimeInstance().format(time);   [69]
    presets.setPresetValue("Time", timeStr);
    if (alarm) {
        presets.setPresetValue("Alarm", "Y");
    } else {
        presets.setPresetValue("Alarm", "Y");

    }
   try {
        presets.setPresetValue("Phone Number", phone);         [77]
    } catch (PanamaException ex) {
        // ignore
    }

    factory.save();
}

public Presets[] getAppointments(User user) throws PanamaException {
    ModelFactory factory = MetaLocator.getInstance().getModelFactory();
    ModelServices services = MetaLocator.getInstance().getModelServices();

    PresetCategory category;
    try {
        category = services.lookupPresetCategory("Appointments");
    } catch (PanamaRuntimeException ex1) {
        throw new PanamaException(ex1);
    }

    Date now = new Date(System.currentTimeMillis());
    Vector allPresets = user.getAllPresets(category);       [97]
    Enumeration enum = allPresets.elements();
    Vector pending = new Vector();
    while (enum.hasMoreElements()) {
        Presets event = (Presets) enum.nextElement();
        String timeStr = event.getPresetValue("Time");
```

```
            Date time;
            try {
                time = DateFormat.getDateTimeInstance().parse(timeStr);    [105]
            } catch (ParseException ex) {
                time = null;
            }
            if (time != null && time.after(now)) {
                pending.add(event);
            } else {
                user.deletePresets(category, new Long(event.getId()));     [112]
            }
        }
        factory.save();

        Presets presetsArray[] = new Presets[pending.size()];
        pending.copyInto(presetsArray);
        return presetsArray;

    }

}
```

Presets 属性フォーマットの正規表現の構文

次の表に、フォーマットの定義に使用できる正規表現の構文を示します。

表 15-1 Presets 属性フォーマットの定義で使用する文字

文字	説明
char	同一の 1 文字と一致
¥	エスケープ文字として使用（例 : ¥*、¥¥、¥w）
¥¥	単一の '¥' 文字と一致
¥0nnn	指定の 8 進番号を持つ 1 文字と一致
¥xhh	指定の 8 ビット 16 進値を持つ 1 文字と一致
¥¥uhhhh	指定の 16 ビット 16 進値を持つ 1 文字と一致
¥t	タブ文字と一致
¥n	改行文字と一致
¥r	復帰文字と一致
¥f	改ページ文字と一致

表 15-2 文字クラス

文字	説明
[abc]	単純な文字クラス
[a-zA-Z]	範囲指定による文字クラス（範囲は "-" と "]" で指定）例 : [x-z]
[^abc]	否定による文字クラス、指定の文字を除外する

表 15-3 標準 POSIX 文字クラス

文字	説明
[:alnum:]	英数字
[:alpha:]	アルファベット
[:digit:]	数字
[:upper:]	大文字のアルファベット

表 15-3 標準 POSIX 文字クラス（続き）

文字	説明
[:lower:]	小文字のアルファベット
[:space:]	空白（スペース、タブ、改ページなど）

表 15-4 変数クラス

クラス	説明
.	改行を除く任意の文字と一致
¥w	英数字 1 文字と一致
¥W	英数字以外の 1 文字と一致
¥s	空白文字と一致
¥S	空白以外の 1 文字と一致
¥d	数字 1 文字と一致
¥D	数字以外の 1 文字と一致

表 15-5 境界との一致

構文	説明
^	行頭と一致
$	行末と一致
¥b	ワード境界と一致
¥B	ワード境界以外と一致

表 15-6 最長一致閉包（できるだけ多数の要素との一致）

要素	説明
A*	A と 0 回以上一致（最長一致）
A	A と 1 回以上一致（最長一致）
A?	A と 1 回または 0 回一致（最長一致）

表 15-6 最長一致閉包（できるだけ多数の要素との一致）（続き）

要素	説明
A{n}	A と正確に n 回一致（最長一致）
A{n,}	A と n 回以上一致（最長一致）
A{n,m}	A と n 回以上 m 回以内で一致（最長一致）

表 15-7 最短一致閉包（できるだけ少数の要素との一致）

要素	説明
A*?	A と 0 回以上一致（最短一致）
A?	A と 1 回以上一致（最短一致）
A??	A と 0 回または 1 回一致（最短一致）

表 15-8 論理演算子

演算子	説明
AB	B が続く A と一致（連結）
A|B	A または B と一致（ユニオン）
(A)	"(" と ")" で囲まれた内側の副式と一致

ロケーション・マーク

位置情報依存サービスは、OracleAS Wireless の主要機能です。ユーザーのロケーションを、E911 または GPS ユニット、あるいはロケーション・マークから取得できます。ロケーション・マークは、ユーザー定義ロケーションです。たとえば、エンド・ユーザーがロケーション認識アプリケーションに自宅、職場および本社の住所を入力するとします。次にレストラン検索アプリケーションを使用すると、そのアプリケーションでは現在のロケーションを使用して運転方向を提供できます。セキュリティとプライバシを確保するために、ユーザーは自分のロケーションにアクセスできるアプリケーションを制御できます。

電話など、ある種のモバイル・デバイスは制限を伴うため、長い英数字文字列を入力したり表示するのは困難です。ロケーション・マークにより、簡潔でわかりやすい名前で識別される 1 つの空間情報が格納されます。たとえば、My Home がロケーション・マーク名で、内部的な空間情報が 123 Main Street, Somewhere City, CA, 12345; Lon = -122.42, Lat = 37.58 となる場合があります。

ユーザーは、ロケーション・マークを全面的に制御でき、任意のデバイスまたは PC でロケーションを簡単に選択、作成、削除および変更できます。

また、ロケーション・マークを使用すると、ユーザーは what-if シナリオを試行して、アプリケーションをデフォルト・ロケーションや現在のロケーションとは異なるロケーションにあるかのように動作させることができます。たとえば、エンタテインメント・サービス・アプリケーションのユーザーがボストンにいて、数日後にモンテビデオに旅行するとします。このユーザーがモンテビデオにロケーション・マークを設定すると、モンテビデオ地域に関連する情報とともに表示されます。各ユーザーはパーソナライズされたロケーション・マークを使用できます。これらは Wireless リポジトリに格納されます。

ロケーション・マークは LocationMark クラスを使用して作成します。また、ユーザーがロケーション・マークを作成するには、OracleAS Wireless Customization Portal にログインし、**「ロケーション・マーク」**タブをクリックして**「作成」**をクリックする方法もあります。ジオコーディング、マッピング、ルーティング、トラフィックおよびリージョン・モデリングの各サービスでロケーション・マークを使用する方法の詳細は、第 14 章「ロケーション・サービスの使用」を参照してください。ロケーション・マークは、地点（緯度と経度）または都市の範囲内の空間的な地域としてジオコーディングできます。

次のコード例では、ユーザー JohnBSmith の職場の住所用のロケーション・マークを作成し、ユーザーのロケーション・プロファイル NEDC AREA LOCATION PROFILE に割り当てる方法を示します。ユーザーは、自分のデバイスからこのロケーション・プロファイルに切り替えて、ロケーション認識サービスからのレスポンスを調整できます。

```
public void createLocations() throws Exception {
        User user = services.lookupUser("JohnBSmith");
        Point point = SpatialManager.createPoint(-71.455, 42.7117);
        Location location = SpatialManager.createLocation(point,"", "","1 oracle
drive", "", "nashua", "nh", "03062", "", "", "", "","us");
        LocationMark locMark = factory.createLocationMark("NEDC AREA", user,
```

```
location, 2.0);
        Profile locationProfile = user.createProfile("NEDC AREA LOCATION PROFILE");
        locationProfile.setLocationMark(locMark);
        factory.save();
    }
```

ユーザー・デバイスの管理

複数デバイスを使用するユーザーは、OracleAS Wireless によって各デバイスのモバイル操作を簡単に管理および最適化できます。デバイスの管理には、PC またはモバイル・デバイスを使用できます。また、ユーザーは現行のデフォルト・デバイスを簡単に変更できます。

ユーザー・デバイス・オブジェクトを使用すると、同じエンティティ内の複数のデバイス・アドレスをグループ化できます。これは、同じデバイスに複数のユーザー・エージェントが含まれている可能性がある場合、または同じデバイスで、アドレスや ID はそれぞれ異なるが、すべて同じ物理エンティティから生成されている複数のプロトコルやチャネルが使用されている可能性がある場合に役立ちます。さらに、1 つのユーザー・エージェントの作業環境設定、ロケーション設定およびデバイス設定が、同じデバイス上の別のユーザー・エージェントに影響を与える場合があります。カスタマイズ・プロファイルは、各ユーザー・デバイス・オブジェクトに対して自動的に作成されます。

ユーザーが新規デバイスのプロファイルを作成すると、デバイスごとに次の属性を入力できます。

- デバイス名
- 1 日ごとの通知の許容回数
- 1 つ以上の住所 / 電話番号
- デバイスのタイプ（音声、WAP、PDA など）

ユーザーとグループの管理

OracleAS Wireless には、任意のフォルダまたはアプリケーションへのアクセスを制限したりアクセス権を付与するためのグループとユーザーの管理およびアクセス制御リスト（ロールとも呼ばれる）が用意されています。「ユーザー・マネージャ」ロールが付与されているユーザーは、任意のデバイスまたは PC を介してグループとユーザーを作成、削除および変更できます。

Group オブジェクトと User オブジェクトは、フォルダとアプリケーションに対するアクセス制御を定義するための便利なメカニズムを表します。ユーザーは、1 つ以上のグループのメンバーである場合があります。すべてのフォルダやアプリケーションは個々のユーザーが所有していますが、グループに割り当てることによって、ユーザーのグループ間で共有できます。フォルダまたはアプリケーションをグループに割り当てると、そのグループ内のすべてのユーザーには、そのフォルダまたはアプリケーションへのアクセス権が付与されます。

ユーザーは、所有するフォルダ、および自分のプライベート・フォルダに格納されたアプリケーションを完全に制御できます。また、自分のプライベート・フォルダ内でアプリケーションを作成、削除または変更できます（特にブックマークとクイックリンク）。

Service Management

OracleAS Wireless では、開発者はエンド・ユーザーが実行できるフォルダ管理操作を全面的に制御できます。また、グループやユーザーに Service Management での全面的な柔軟性を与えるか、Service Management の使用を制限できます。

サービスとフォルダは、次の方法で編成できます。

- ユーザー指定の順序番号（任意の順序）
- 五十音順
- 作成日
- アクセス頻度または最新アクセスに基づく動的順序

また、エンド・ユーザーは、ブックマークとクイック・リンクを使用して、モバイル操作をカスタマイズできます。これによって、頻繁にアクセスするサービスをホーム・デッキまたは他の必要なフォルダにリンクできます。

16

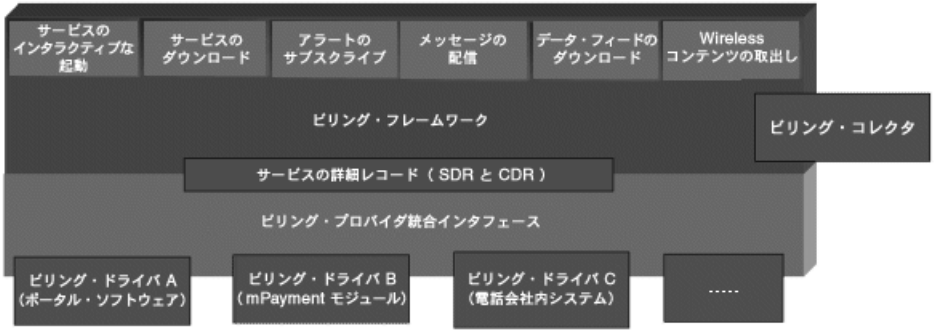
請求

項ごとに様々なトピックを記載しています。各項の内容は、次のとおりです。

概要

OracleAS Wireless の Billing Integration Framework には、拡張可能で柔軟なフレームワークが用意されており、請求可能なサービスのモデル化、請求可能なアクションの取得および請求エンジンとの統合に使用されます。

図 16-1 ビリング・フレームワークのアーキテクチャ



概要

請求可能なアクション : 追跡および認証され、請求システムにレポートされる顧客のアクション。一般的な例には、割増サービスへのアクセス、モバイル・コンテンツのダウンロード、アラート・サブスクリプション、メッセージ配信などがあります。

ビリング・コンテキスト : 実装オブジェクトによって、指定の請求可能なアクションに関連するすべてのデータが提供されるマーカー・インタフェース。OracleAS Wireless には、請求コンテキストとして事前定義されている次のオブジェクトがあります。

- マルチチャネル・サービス・アクセス用 : `oracle.panama.rt.ServiceContext`
- J2ME アプリケーションのダウンロード用 : `oracle.panama.model.UserDownloadStatus`
- モバイル・アラート・サブスクリプション用 : `oracle.panama.mobilealert.ServiceAlertSubscription`
- メッセージ用 :
  - 送信 : `oracle.panama.messaging.transport.impl.SendingBillingContext`
  - 受信 : `oracle.panama.messaging.transport.impl.ReceivingBillingContext`

サービスの詳細レコード（SDR）: 指定の請求可能なアクションのビリング・コンテキストから請求固有の属性すべてを取得する汎用オブジェクト。属性には、必須属性と拡張属性の 2 種類があります。必須属性は、ユーザー情報、サービス情報およびコンポーネント情報です。拡張属性には、コンポーネント固有の情報が含まれます。

ビリング・コレクタ : BillingContext を処理し、対応するサービスの詳細レコードを生成する拡張可能なオブジェクト。

ビリング・ドライバ : OracleAS Wireless によって定義された Java インタフェースで、指定の請求可能なアクションを処理するために外部の請求システムと対話します。請求可能なアクションは、通常、次の 2 つのステップで処理されます。

- アクションの開始前に、ドライバの preService API が請求フレームワークによってコールされます。例 : ユーザーのバランスとクレジット、リソース予約、レーティングおよび不正の検出に基づいて使用量を承認する一般的な例で、このドライバは、ユーザーの請求可能なアクションのリクエストを否認するようにビリング・フレームワークに対して指示できます。
- アクションが開始されると、ドライバの postService API がビリング・フレームワークによってコールされ、請求トランザクションが完了します。

Billing Integration Framework の使用

請求可能なアクションと請求ステムの対話

デフォルトの請求可能なアクション

OracleAS Wireless では、次の請求可能なアクションが定義されています。

- マルチチャネル・サービス・リクエスト : モデル・サービスはゼロ以外の値に設定されたコスト属性を使用して定義されます。
  - サービス認証の直後とサービス起動の直前に、ドライバの preService API をコールします。この API コールの結果コードでエラーが示された場合、そのリクエストは拒否されます。
  - サービス・リクエストの処理後とリスナー通知の前に、ドライバの postService API をコールします。
- J2ME アプリケーションのダウンロード :
  - プロビジョニング・サーバーがダウンロード・リクエストを生成する直前に、ドライバの preService API をコールします。この API コールの結果コードでエラーが示された場合、そのリクエストは拒否されます。

- プロビジョニング・サーバーが、正常にインストールされているデバイスから通知を受信した後およびリスナー通知の前に、ドライバの postService API をコールします。

- アラート・サブスクリプション:
  - ユーザーがモバイル・アラート・サービスをサブスクライブする直前に、ドライバの preService API をコールします。この API コールの結果コードでエラーが示された場合、そのリクエストは拒否されます。
  - アラート・サブスクリプションが作成された後に、ドライバの postService API をコールします。
- メッセージ:
  - 送信
    * メッセージ・フレームワークがメッセージの配信リクエストを受け入れる前に、ドライバの preService API をコールします。この API コールの結果コードでエラーが示された場合、そのリクエストは拒否されます。
    * メッセージ・フレームワークがメッセージの配信リクエストを受け入れた後に、ドライバの postService API をコールします。
  - 受信
    * 登録済のターゲット・アプリケーションにメッセージがルーティングされる前に、ドライバの preService API をコールします。この API コールの結果コードでエラーが示された場合、そのメッセージは破棄されます。
    * 登録済のターゲット・アプリケーションにメッセージがルーティングされた後に、ドライバの postService API をコールします。

カスタムの請求可能なアクション

OracleAS Wireless の Billing Integration Framework を使用すると、カスタムの請求可能なアクションを、ユーザーのニーズにあわせて導入できます。次に、請求可能な新規アクションを導入するステップを示します。

- ランタイム・フックや JSP など、請求可能な新規アクションを配置する場所を識別します。
- この請求可能な新規アクションに関連したすべてのデータへのアクセスを提供するオブジェクトを定義します。このオブジェクトは、マーカー・インタフェースの BillingContext を実装しているかぎり、どの Java クラスでも構いません。前に述べた事前定義の BillingContext オブジェクトも必要に応じて使用できます。
- ビリング・コレクタ・オブジェクトをカスタマイズし、BillingCollectorImpl を拡張するか、ビリング・コレクタの独自の実装を定義することで、新規の BillingContext オブジェクトを処理します。ビリング・コレクタの実装については、次の項で詳細に説明します。

ビリング・コレクタ・クラスは、既製のコレクタ実装から拡張（サブクラス化）することも、新規に作成することもできます。作成した場合、実装クラスは次の方法で設定します。「Enterprise Manager」→「Wireless サーバー」:「サイト管理」→「ビリング・フレームワーク」→「ビリング・コレクタ・クラス名」。

次のコード部分を使用して、preService コールの請求可能な新規アクションをトリガーします。

```
// Suppose your BillingContext Object is foo
BillingController controller =  BillingController.getInstance();
if (controller.isBillingEnabled() )
 {
       try {
             BillingResult result = controller.preService(foo);
            if ( result != null ){
               ServiceDetailRecord sdr = result.getServiceDetailRecord();
               if (result.getResultCode() != BillingResult.FAILED) { //Succeed
                 // Add your logic here
               }
              else { // Failed
                    // Add your logic here to handle preService failure
                    //BillingException e = new BillingException(failure message);
                    // e.setResult(result);
                    // throw e;

               }
          }
      } catch (BillingException be) {
              //Handle Billing Execption here
        }
```

次のコード部分を使用して、postService コールの請求可能な新規アクションをトリガーします。

```
// Suppose your BillingContext Object is foo
BillingController controller =  BillingController.getInstance();
if (controller.isBillingEnabled() )
 {
       try {
             BillingResult result = controller.postService(foo);
            if ( result != null ){
               ServiceDetailRecord sdr = result.getServiceDetailRecord();
               if (result.getResultCode() != BillingResult.FAILED) { //Succeed
                 // Add your logic here
               }
              else { // Failed
                    // Add your logic here to handle postService failure
               }
```

```
    }
} catch (BillingException be) {
        //Handle Billing Execption here
   }
```

BillingLoader ユーティリティ

BillingLoader ユーティリティは、請求トランザクションのレコードをダウンロード、パージおよびアップロードするためのバッチ・ユーティリティです。BillingLoader ユーティリティの詳細は、『Oracle Application Server Wireless 管理者ガイド』を参照してください。

ビリング・コレクタとサービスの詳細レコード

各コンポーネントに関する請求可能な操作は、イベント前およびイベント後の 2 つの部分に分割されます。SDR は、イベント前およびイベント後の両方について生成されます。各サービス・コールの前または後の SDR の作成には、ビリング・データ・コレクタ・クラスが使用されます。

サービス前 SDR が請求システムに渡されると、その請求システムは、一意の請求参照 ID を設定し、その SDR を請求結果の一部として戻します。

戻されたサービス前請求参照 ID は、OracleAS Wireless によって抽出されます。この ID は、同じ請求可能操作のサービス後 SDR に設定されるため、外部の請求システムは、この請求参照 ID に基づいて、すべての請求可能操作のイベント前とイベント後の間の状態を保持できます。

データベースに記録されるのは、サービス前の後で戻す SDR にトランザクション ID 属性を設定することで、ドライバがトランザクションを開始しないかぎり、サービス後請求イベントの SDR のみです。トランザクション ID が設定されている場合は、サービス前の SDR も記録されます。サービス前とサービス後の SDR は、*PRE_SERVICE* または *POST_SERVICE* のいずれかを示す `LOG_TYPE` 列によって識別できます。

SDR ID は、表内で主キーとして使用される内部 ID です。特定の SDR は、SDR ID に基づいて参照できます。

通常は、ドライバがトランザクションを開始（この場合はサービス前 SDR が記録されます）していないかぎり、各サービス後イベントに対する SDR ID は 1 つのみです。

ビリング・コレクタのデフォルト実装

BillingCollector 実装オブジェクトは、指定の請求可能アクションの BillingContext を処理します。この実装オブジェクトは、ビリング・ドライバで処理する適切なサービスの詳細レコードを生成します。OracleAS Wireless には、次の BillingContext オブジェクトを処理する `BillingCollectorImpl` という名前のデフォルトの BillingCollector 実装が含まれています。

- マルチチャネル・サービス・アクセス用 : `oracle.panama.rt.ServiceContext`
  - 拡張属性名 :
    - SERVICE_URL: 起動されるサービスの URL。
    - SERVICE_TYPE: サービスのタイプ。つまり、フォルダまたはリンク（FOLD、LINK）。
    - DEVICE_NAME: デバイス名。
    - INVOKER: このランタイム・サービスの実行者。値リストには、HTTP、ASYNC、ALERT、PROVISIONING、AGENT が含まれています。この属性をドライバで使用すると、このサービスのルート実行者をトレースできます。この実行者属性と値リストは、BillingDataCollector インタフェースで定義します。
    - ASK_IN_MSGID: このオグジェクトは、`INVOKER` 属性値が ASYNC の場合に設定されます。この値は、非同期アプリケーションの着信メッセージ ID を示します。この値は、非同期アクションにメッセージ・リクエストを関連付けるために使用できます。
- J2ME アプリケーションのダウンロード用 : `oracle.panama.model.UserDownloadStatusAPPLICATION_NAME`
  - 拡張属性名 :
    - CONTENT_NAME: アプリケーション名。
    - CONTENT_VERSION: コンテンツ・バージョン。
    - USER_DEVICE_NAME（ある場合）: ユーザー・デバイス名。
    - MIME_TYPE: ダウンロードする MIME タイプ。
    - CURRENT_NUMBER_OF_DOWNLOADS: 現在のダウンロードより前に行われたダウンロードの回数。
    - CONTENT_SIZE: コンテンツのサイズ。

- モバイル・アラート・サブスクリプション用 :
  `oracle.panama.mobilealert.ServiceAlertSubscription`
  - 拡張属性名 :
    * OPERATION_TYPE: アラート・メッセージ SDR か、またはアラート・サブスクリプション SDR かを示します。モバイル・アラートの値は ALERT_SUBSCRIPTION です。
    * SERVICE_URL: 起動されるサービスの URL。アラート・エンジンがサービスを起動してコンテンツを生成し、この属性がそのサービスの URL となります。
    * ALERT_DELIVERY_ADDR: アラートの配信先のエンド・ユーザー・アドレス。カンマで区切られた配信可能なアドレスのリストです。
- メッセージ用 :
  - 送信 :
    `oracle.panama.messaging.transport.impl.SendingBillingContext`
  - 拡張属性名 :
    * TYPE: アクションが送信か、または受信かを示します。送信の場合は、S に設定されます。
    * FROM: 送信元アドレス（NULL でも構いません）。
    * TO: 送信先アドレス。
    * DELIVERY_TYPE: 配信チャネル・タイプ。
    * SERVICE_NAME（オプション）: クライアント名（プッシュ・サーバー、非同期エージェントなど）。
    * DRIVER_NAME: ドライバ名（送信の場合のみ）。
    * REPLY_TO: 返信先アドレス。
  - 受信 :
    `oracle.panama.messaging.transport.impl.ReceivingBillingContext`
  - 拡張属性名 :
    * TYPE: アクションが送信か、または受信かを示します。受信の場合は、R に設定されます。
    * FROM: 送信元アドレス（NULL でも構いません）。
    * TO: 送信先アドレス。
    * DELIVERY_TYPE: 配信チャネル・タイプ。

* SERVICE_NAME（オプション）: クライアント名（プッシュ・サーバー、非同期エージェントなど）。
* DRIVER_NAME: ドライバ名（送信の場合のみ）。
* REPLY_TO: 返信先アドレス。

サービスの詳細レコード ID と請求参照 ID

サービスの詳細レコード ID は、ビリング・コレクタによって定義されます。デフォルトの書式は [Component Name: Random Key] です。請求参照 ID は、請求システムから取得されます。請求の前処理では、通常、認証 ID として定義されます。

デフォルトのビリング・コレクタの拡張

OracleAS Wireless の Billing Integration Framework を使用すると、デフォルトのビリング・コレクタ実装を様々な理由で拡張できます。

- デフォルトの BillingContext のデフォルト実装よりも多いデータを取得するためのコードは、次のとおりです。

```
public class MyBillingCollector extends
oracle.wireless.billing.BillingDataCollectorImpl
{
   ...
    public ServiceDetailRecord createServiceDetailRecord(BillingContext context)
   {
       ServiceDetailRecord sdr = null;
        if (context instanceof oracle.panama.rt.ServiceContext ){
            sdr = super.createServiceDetailRecord(context);
           if ( sdr != null){
             // Add your additional data capture logic here
            // For example:
           // oracle.panama.rt.ServiceContext serviceContext =
(oracle.panama.rt.ServiceContext)context;
           // sdr.setExtendedData("SERVICE_PATHURL",
context.getService().getURLPathParameter());
         }
       }
       else{
          sdr = super.createServiceDetailRecord(context);
      }
      return sdr;
   }
  ...
}
```

- カスタムの BillingContext からデータを取得するためのコードは、次のとおりです。

```
public class MyBillingCollector extends
oracle.wireless.billing.BillingDataCollectorImpl
{
   ...
    public ServiceDetailRecord createServiceDetailRecord(BillingContext context)
   {
       ServiceDetailRecord sdr = null;
        if (context instanceof MyBillingContext ){
             // MyBillingContext mycontext = (MyBillingContext)context;
            // get the user name from your context
            // Java.sql.Timestamp accessTime = null;
            // BillingManager manager = BillingManager.getInstance();
            //sdr = manager.createServiceDetailRecord(<User Name>, <Your service
Name>, accessTime, <your billing component name>);
           // sdr.setExtendedData("MY_DATA", <Get data from your context object>);
           ...
         }
       }
       else{
          sdr = super.createServiceDetailRecord(context);
      }
      return sdr;
   }
  ...
}
```

コンテキスト・データに基づいて請求可能アクションを無視するには、ビジネス・ロジックに基づいて、`createServiceDetailRecord` コールから NULL オブジェクトを戻します。

Oracle Enterprise Manager を使用して、拡張したビリング・コレクタを指定する方法は、次のとおりです。

「Enterprise Manager」→「Wireless サーバー」:「サイト管理」→「ビリング・フレームワーク」→「ビリング・コレクタ・クラス名」。

複数の部分からなるリクエストでのドランザクション・コンテキストのメンテナンス

一般的な Wireless リクエストは、複数のエンティティ（複数の部分からなるリクエスト）にまたがっています。たとえば、非同期リクエストはメッセージング・サーバーで作成されて、非同期リスナーに転送されます。このリスナーは、ランタイム・サービスを起動してから、メッセージング・サーバーを介して結果をユーザーに戻します。請求規則では、トランザクションの各段階での請求が求められますが、価格設定は、リクエストのコンテキストに従って変化する可能性があります。このような場合をサポートするには、請求規則を実施する場合に、リクエストのパスの履歴（ビリング・ドライバ [またはデータ・コレクタ] で使用できるリソース）を保持する必要があります。

請求トランザクションの作成と割当て

ビリング・ドライバ（つまり、データ・コレクタ）によって、特定のサービスの詳細レコードをトランザクションの一部として割り当てることができます。たとえば、ビリング・ドライバのサービス前コール実装によって、SDR に追加する次の行を、新しく作成するトランザクションに含めることができます。

```
          BillingTransaction Trans =
BillingTransactionManager.getInstance().createTransaction();
         Trans.getId(); // if its XYZ say
          sdr.setTransaction(Trans);
```

これによって、新規トランザクションが作成され、そのトランザクションにサービスの詳細レコードが追加されます。後続のサービスの詳細レコードは、トランザクション ID を保持することで、同じトランザクションに追加できます。次に、後続のサービスの詳細レコードのための擬似コードを示します。

```
           BillingTransaction Trans =
BillingTransactionManager.getInstance().lookupTransaction (XYZ);
         Trans.getId(); // if its XYZ say
         Trans.addSdr(sdr); // OR sdr.setTransaction(Trans);
```

このトランザクションに関する過去のサービスの詳細レコードは、BillingTransaction 公開 API（`BillingTransaction` の `getServiceDetailRecords()` を参照）を使用して検索できます。

サービスの詳細レコードに対するロギング規則

トランザクションの一部であるサービスの詳細レコードは、BillingDriver 実装からコール `BillingResult` が戻されると、データベースに暗黙的に記録されます。BillingController は、サービスの詳細レコードを調べて、この操作を自動的に処理します。

このトランザクションの SDR 部分は、検索できます（BillingTransaction の getServiceDetailRecords() を参照）。SDR がトランザクションの一部でない場合、そのロギングはロギング・フレームワークまで延期され、バックグラウンドで処理されます。

単一スレッドの複数の部分からなるリクエストでのトランザクション状態のメンテナンス

複数の部分からなるリクエストのすべての後続リクエストが、同じ Java スレッドのコンテキスト内で処理されている場合、そのトランザクション情報は、スレッド内に、スレッド・ローカル・オブジェクトとして格納され、後続のリクエスト用に後で参照できます。BillingDataCollector のデフォルト実装（oracle.wirless.billing.BillingDataCollectorImpl）によって提供されるのは、その内容のとおりです。設定と取得のためのトランザクション API は、BillingController オブジェクトを介して使用可能です。

```
  /**
   * Returns the current billing transaction
   * @return BillingTransaction the current billing transaction
   */
public BillingTransaction getCurrentTransaction();

/** Sets the transaction for the current transaction
 * @param transaction the current transaction
 */
public void setCurrentTransaction(BillingTransaction transaction);
```

ビリング・ドライバ

OracleAS Wireless のビリング統合を外部の請求システムと統合するには、システム・インテグレータが、BillingDriver インタフェースを実装する Java 実装クラスを提供する必要があります。ドライバを設定する方法は、次のとおりです。

「Enterprise Manager」→「Wireless サーバー」:「サイト管理」→「ビリング・フレームワーク」→「ビリング・プロバイダ・ドライバ」の「ドライバ・クラス名」。

ビリング・ドライバ実装の望ましい動作は、次のとおりです。

- init: この API には、外部の請求システムに接続するためのロジックが含まれている必要があります。
- preService: この API には、次の請求操作のいずれか、またはすべてを処理するためのロジックが含まれている必要があります。
  - 認証
  - レーティング
  - リソース予約

  - 不正の検出
  - その他
- postService: この API には、請求トランザクションを完了するためのロジックが含まれている必要があります。
- cancelService: この API には、請求トランザクションを取り消すためのロジックが含まれている必要があります。
- destroy: この API には、請求システムとの接続を切断するためのロジックが含まれている必要があります。

> **注意：** このリリースでは、1 回に統合できる請求システムは 1 つのみです。ただし、ドライバ実装では、異なる請求システムとインタフェースするプロキシを必要に応じて使用できます。

Billing Integration Framework の使用例

請求の前処理

次に、請求の前処理の実装例を示します。

- デフォルトまたはカスタムのアクションを使用して請求可能なアクションを定義します。
- ドライバの `preService` コールを使用して、請求の認証またはリソースの予約 (あるいはその両方）を行い、請求システムによって生成された認証 ID を戻します。
- ドライバの `postService` コールを使用し、`preService` コールから取得した認証を渡して、請求トランザクションを完了します。

請求の後処理

次に、請求の後処理の実装例を示します。

- デフォルトまたはカスタムのアクションを使用して請求可能なアクションを定義します。
- ドライバの `preService` コールを使用して、請求の認証または不正の検出（あるいはその両方）を行います。
- ドライバの `postService` コールを使用して、請求関連のデータを請求システムに送信します。

A

サポートされる XHTML モジュール

この付録では、サポートされる XHTML モジュールとその機能について説明します。各項の内容は、次のとおりです。

Structure モジュール

XHTML の Structure モジュールは、次の要素を定義します。

- <html>: OracleAS Wireless によって、追加の *profile* 属性が定義されます。この *profile* 属性は、特定の XHTML プロファイルに準拠している文書を指定します。プロファイルの値は URI リストを受け入れ、URI 値の http://xmlns.oracle.com/ias/dtds/xhtml+xforms/0.9.0/1.0 が含まれている必要があります。
- <head>
- <title>
- <body>

Text モジュール

XHTML の Text モジュールは、次の要素を定義します。

- 、<p>
- 、
- <address>、<blockquote>、<h1>、<h2>、<h3>、<h4>、<h5>、<h6>、<pre>
- <abbr>、<acronym>、<cite>、<code>、<dfn>、、<kbd>、<q>、<samp>、、<var>

HyperText モジュール

XHTML の HyperText モジュールは、次の要素を定義します。

- <a>
  - *href* 属性によって、属性値テンプレートが（XSLT で使用されているように）サポートされます。このため、<a href="{instance('data')/links/url}"> などの URL を XForms インスタンスから取得するためのアンカーが可能となります。
  - type 属性は、示されたリソース（href）のコンテンツ・タイプ（MIME タイプ / メディア・タイプ）が、起動時に生成されるかどうかを示すために使用されます。*type* 属性に、OracleAS Wireless でサポートされていない MIME タイプの値が含まれている場合、このリソースは外部リソースとして処理されます。OracleAS Wireless は、この URL リソースの仮想ブラウザまたはプロキシとして機能しません。このようなリソース（リンク）にユーザーがナビゲートすると、デバイスはそのリソースからコンテンツをフェッチします。
  - *rel* 属性は、現在の文書とターゲット・リソース（href）との関係を示します。この *rel* 属性には、値がスペースで区切られている場合などに、複数の関係を保持できま

す。既知の rel タイプのリストは、
http://www.w3.org/TR/html4/types.html#type-links から取得できます。
OracleAS Wireless で認識される *rel* の値は、次のとおりです。

* `next`: 参照した href リソースは、アプリケーション内の次の文書に相当する可能性があることを示します。ブラウザはこの値を使用して、その文書をプリフェッチできます。Next 値は、これらのリソースのプリフェッチ・インジケータとして使用されます（プリフェッチ・インジケータがない場合に、リソースがデバイスによってフェッチされるのは、ユーザーによってアクティブ化された場合のみです）。

* `maxage`: OracleAS Wireless によって追加された値。この値は、デバイスまたはブラウザでは、指定時間を経過していないキャッシュの文書が使用できることを示します。時間はダッシュ（-）をセパレータとして使用した maxage の値で指定します。次に例を示します。`rel="maxage-5"`（5 秒間を意味します）。maxage の時間単位は秒です。この値がサポートされるのは、VoiceXML ゲートウエイを使用する音声インタフェース上のみです。

* `maxstale`: OracleAS Wireless によって追加された値。この値は、ブラウザでは、有効期限に指定の制限時間を加えた時間を経過していないキャッシュの文書が使用できることを示します。時間はダッシュ（-）をセパレータとして使用した maxstale の値で指定します。次に例を示します。`rel="maxstale-5"`（ブラウザでは失効した文書を受け入れることができ、その文書は 5 秒間失効しないことを意味します）。maxstale の時間単位は秒です。この値がサポートされるのは、VoiceXML ゲートウエイを使用する音声インタフェース上のみです。

* `connecttimeout`: OracleAS Wireless によって追加された値。この値は、アンカーによって `"tel:.."` などの電話サービスが起動された場合に、接続タイムアウトが発生するまでの最小時間をブラウザに示します。制限時間は、ダッシュ（-）をセパレータとして使用した connect-timeout の値で指定します。次に例を示します。`rel="connecttimeout-5ms"`（最短 5 ミリ秒後に接続が継続されていなかった場合は、ブラウザによるタイムアウト発行が可能であることを意味します）。connect-timeout の時間単位は s または ms のいずれかです（単位の指定が必要です）。

* `caching`: 可能な値は fast および safe です。この値がサポートされるのは、VoiceXML ゲートウエイを使用する音声インタフェース上のみです。

rel 属性の使用例

```
<a href="mypage.jsp" rel="Next Maxage-0 Maxstale-5 Fetchtimeout-2">Link</a>
```

href がサポートするプロトコルは、次のとおりです。

- HTTP（http:// または https://）: 業界標準の HTTP プロトコル。
- OMP（omp://）: アプリケーションでは、仮想 URL を使用して、OracleAS Wireless に定義されているサービスにリンクできます。
- Mail To（mailto:）: アプリケーションでメール・アプリケーションを起動できます（ターゲット・デバイスでサポートされている場合）。
- Tel To（tel:）: アプリケーションで電話アプリケーションを起動できます（ターゲット・デバイスでサポートされている場合）。

リンクの構文（音声インタフェース用）を埋め込むには、<object> モジュールを使用します。

List モジュール

XHTML の List モジュールは、次の要素を定義します。

- ol、ul、li
- dl、dt、dd
- nl: ナビゲーション・リスト

ナビゲーション・リスト（nl）は、XHTML 2.0 仕様（http://www.w3.org/tr/xhtml20）の一部です。ナビゲーション・リストは、ナビゲーション・メニューを定義するために使用されます。各ナビゲーション・リストは、各メニュー項目の <name> 要素と <li> 要素をサポートします。ナビゲーション・リストはネスト可能で、 要素には、別のナビゲーション・リストを含めることができます。

ナビゲーション・リスト内の各メニュー項目は、 要素によって定義され、この <li> 要素には、*href* 属性（および *type*、*rel* などのリンク属性と <a> 要素によって定義された別の属性）があります。 要素のコンテンツは、メニュー内のメニュー項目として表示され、選択すると、*href* 属性によって定義されたリンクがその後に続きます。

ナビゲーション・リスト項目の <li> は、*href*、*rel* および *type*（コンテンツ・タイプ）の各属性をサポートします（*rel* 属性と *type* 属性の詳細は、前述の <a> を参照）。ナビゲーション・リストに音声構文を定義するには、 要素のコンテンツ・モデル内で <object> モジュールを使用する必要があります。

ナビゲーション・リストのネスト例

```
<nl>
   <name>Contents</name>
   <li href="/link1">Item 1</li>
   <li>
     <nl>
       <name>Nested Menu List</name>
       <li href="sub1">Sub List Item 1</li>
       <li href="sub2">Sub List Item 2</li>
     </nl>
   </li>
   <li href="#conformance>Conformance</li>
 </nl>
```

ネストしたナビゲーション・リストをネストした場合（ 内に <nl> がある）、表示されるのは、そのネストしたナビゲーション・リストのみです（同じ <li> を持つ他の要素はすべて無視されます）。

Presentation モジュール

OracleAS Wireless は、XHTML の Presentation モジュールに定義されている次の要素をサポートしています。

- <hr>

Object モジュール

XHTML の Object モジュールは、次の要素を定義します。

- <object>: OracleAS Wireless では、*declare*、*name*、*usemap* および *standby* の各属性をサポートしていません。data 属性は、<a> 要素で定義されているとおりに、属性値テンプレートの使用をサポートします。object 要素は、外部コンテンツを埋め込むために使用されます。この外部コンテンツには、イメージやオーディオの他に、アプレットやフラッシュ（ターゲット・デバイスがこのようなコンテンツをサポートしている場合）などの他の埋込みコンテンツがあります。objects タグには、HTML マークアップを含めることができます。<object> のセマンティクスでは、デバイスのユーザー・エージェントがオブジェクトの *type* を認識しない場合やユーザー・エージェントがその <object> をレンダリングできない場合に、埋込みマークアップ（その <object> 内の）のレンダリングが要求されます。オブジェクトはネスト可能で、同じセマンティクスがネストされているすべての <object> に適用されます。

次の例で、<object> タグ内のマークアップは、ターゲット・ユーザー・エージェントがイメージをサポートしていない場合にレンダリングされます。

```
<object data="myimage.jpg" type="image/jpeg">
    <strong>Images</strong>  are <em>not</em> supported by your device
</object>
```

イメージの埋込み

文書にイメージを埋め込むには、<object> 要素を使用する必要があります。<object> の data 属性は、イメージ・リソースがある URI を指し示し、type 属性は、そのイメージのメディア・タイプ（MIME タイプ）を指定します。

様々なデバイスの複数イメージ・フォーマットは、ネストされている <object> を使用することでサポートできます。次の例で、OracleAS Wireless は、現行のデバイス・リクエストについて、適正なイメージをレンダリングしています。

```
gif image, if gif not supported then bmp else wbmp
```

```
  <object data="http://../myimage.gif" type="image/gif">
     <!-- render "bmp" if "gif" not supported -->
     <object data="http://../myimage.bmp" type="image/bmp">
        <!-- render "wbmp" if "bmp" not supported -->
        <object data="http://../myimage.wbmp" type="image/wbmp">
            Your UA does not support any image format
         </object>
      </object>
  </object>
```

デバイスによってサポートされているイメージがない場合、OracleAS Wireless は、トップレベルの <object> に指定されているイメージを、そのデバイスがサポートしているフォーマットに調整しようとします。この調整プロセスは、デバイス / メディアのプロパティに基づいたイメージのフォーマットとサイズに影響を与えます。開発者は、この調整プロセスに影響を与える追加のパラメータを指定できます。

指定できるオプション・パラメータは、*adapt* および *notsize* です。<param> を、値の属性が false に設定されている *adapt* 名で指定すると、<object> に対するイメージ調整プロセスが停止します。<param> を、値の属性が *notsize* に設定されている *adapt* 名で指定すると、<object> に対するサイズ調整が停止し、イメージのフォーマットに関する調整のみが行われます。

注意： 元のイメージが画面（またはそのオブジェクトが発生する関連のフレーム）に十分納まるほど小さい場合、デフォルトではサイズは変更されません。

次に、イメージの自動調整をすべて停止する例を示します。

```
<!-- Turn of auto adaptation, using <param> element -->
<object id=myid2" data="images/oralogo.gif" type="image/gif">
   <param name="adapt" value="false" />
</object>
```

次に、自動調整をイメージのフォーマットのみに制限する例を示します。

```
<!-- Turn of auto size adaptation, using <param> element -->
<object id=myid2" data="images/oralogo.gif" type="image/gif">
   <param name="adapt" value="notsize" />
 </object>
```

> **注意：** デバイスによってサポートされているイメージがない場合、OracleAS Wireless は、デバイスでサポートされているフォーマット、幅および高さに基づいて、そのイメージを調整しようとします。<object> の幅と高さは、絶対的な数値ではなく、パーセントで指定（CSS の幅と高さを使用）することをお薦めします。これによって、効率的で一貫性のあるイメージ調整が可能となります。

オーディオの埋込み

文書にオーディオ・コンテンツを埋め込むには、<object> 要素を使用する必要があります。<object> の data 属性は、オーディオ・リソースがある URI を指し示し、type 属性は、そのイメージのメディア・タイプを指定します。オーディオ・コンテンツは、ビジュアル・デバイスとオーラル（音声）デバイスの両方で使用できます。特定のフォーマットが音声ゲートウェイやビジュアル・ブラウザでサポートされていない場合は、埋込みオーディオ・マークアップを使用し、TTS を使用してテキストを再生します。

```
<object id="myid" data="myaudio.gif" type="audio/wav">
   <div>
        <p>Audio is not supported by your UA<p>
        <p>This will played using TTS</p>
   </div>
</object>
```

> **注意：** メディア・タイプ（*type* 属性）は、オーディオ・ファイルの正確なフォーマットを識別しません。フォーマットの識別には、<param> 要素が使用されます。オーディオ・フォーマットを識別するには、name="format" を指定した <param> 要素およびそのオーディオ・フォーマットが含まれた値を使用します。また、オーディオを埋め込む場合は、<param> 要素を使用して、*format*、*fetchhint*、*fetchtimeout*、*maxage*、*maxstale* および *cache* の各パラメータを指定できます。

音声と DTMF 構文の埋込み

音声構文のリソースを埋め込むには、<object> 要素を使用する必要があります。<object> の data 属性は、構文のリソースを指し示し、type 属性は、構文のフォーマットを指定します。

次に、音声構文をリンク・アンカー（<a>）に関連付ける例を示します。

```
<a href="mypage.jsp"> My Home Page
  <object data="grammar.dat" type="application/srgs+xml"/>
</a>
```

複数のオブジェクトを使用すると、1 つ以上の構文を関連付けることができます。

```
<a href="mypage.jsp"> My Home Page
  <object data="grammar.dat" type="application/srgs+xml"/>
  <object data="#Grammar_in_head_1""
type="vnd.oracle.srgs+xmlapplication/srgs+xml"/>
</a>
```

<object> 要素では、インライン構文も使用できます。インライン構文を埋め込むために、OracleAS Wireless では、<grammar> 要素を HTML 要素の head セクションで使用できるため、<object> 要素は、フラグメント URI（#id）を使用してインライン構文を指し示すことができます。この <grammar> 要素（およびその名前空間）は、W3C Speech Recognition Grammar Specification（http://www.w3.org/TR/speech-grammar/）による定義に従っています。

```
<head>
  <grammar xmlns="http://www.w3.org/2001/06/grammar"
           id="grammar_in_head_1">
      ........
  </grammar>
  <grammar xmlns="http://www.w3.org/2001/06/grammar"
           id="grammar_in_head_2">
      ........
  </grammar>
</head>
```

注意： <object> を使用して構文を埋め込む場合は、<param> 要素を使用して、*scope*、*mode*、*root*、*version*、*weight*、*fetchhint*、*fetchtimeout*、*maxage*、*maxstale*、*caching* の各構文パラメータを指定できます。

音声認識の詳細は、A-11 ページの「Speech Recognition Grammar モジュール」を参照してください。

<param> の使用

<object> に関するパラメータは、<param> 要素を使用して指定できます。たとえば、<param> 要素を使用すると、プリフェッチをサポートし、オーラル・インタフェース（オーディオや構文リソースなど）に関するキャッシュされたソースを使用できます。次に、<param> 要素の使用例をいくつか示します。

- プリフェッチのサポートには、<param name="fetchhint" value="prefetch"/> を使用します。
- キャッシュされたリソースを使用可能にするには、maxage または maxstale（あるいはその両方）を使用します。この値は、秒単位で示されます。

```
<param name="maxage" value="5"/>
<param name="maxstale" value="5"/>
```

Basic Tables モジュール

XHTML の Basic Tables モジュールは、次の要素を定義します。

- <table>
- <caption>
- <tr>
- <tr>、<th>

ネストした表は使用できません（XHTML の Basic Tables ではこの制約が定義されています）。表は、表データを示す場合にのみ使用することをお薦めします。表を表現の目的で使用しないでください、使用すると、デバイスの独立モデルが破壊されます。レイアウトには CSS プロパティを使用します。

OracleAS Wireless は、*rowspan* 属性や *colspan* 属性をサポートしていません。

Meta Information モジュール

XHTML の Meta Information モジュールは、次の要素を定義します。

- <meta>: 作成者が HTTP プロトコル・セマンティクスの埋込みに <meta> 要素を使用した場合でも、文書に関するメタ情報の指定が必要です。meta 要素を、http の相当する要素（refresh、expires、cache-control など）に使用すると、すべてのユーザー・エージェントまたはデバイスで一貫性のある結果が得られない可能性があります。文書をデバイスに依存しないように作成するには、http の相当する要素として <meta> 要素を使用しないでください。meta 要素の値の例には、author、copyright、description、keywords、maintainer、robots、bookmark などがあります。

OracleAS Wireless では、次のような特別の <meta> 要素をサポートしています。

- name=" __ASYNC_NO_RESPONSE__" を指定した <meta>

  この meta タグは、レスポンスをデバイスに転送しないことを OracleAS Wireless サーバーに対して指示します。この指示が有効なのは、ターゲット・デバイスがメッセージ・ベースのデバイスである場合のみです。プロトコル・ベースのデバイスでのメッセージでは、リクエスト・メッセージへのレスポンスを要求しない場合があります。アプリケーションでは HTTP プロトコルが使用されているため、有効なレスポンス・ストリームの生成が強制されます。この <meta> 値によって、アプリケーションではデバイスに転送されないレスポンス・ストリームを生成できます。

Style Sheet モジュール

XHTML の Style Sheet モジュールは、次の要素を定義します。

- <style>: XHTML 文書内に CSS のスタイル規則を埋め込むために使用されます。*type* 属性には、text/css メディア・タイプ（MIME タイプ）を使用する必要があります。

Style Attribute モジュール

Style Attribute モジュールは、*style* 属性を定義します。この *style* 属性は、指定の要素に CSS のスタイル規則を指定します。

Link モジュール

XHTML の Link モジュールは、次の要素を定義します。

- <link>: この要素によって、XHTML 文書プロセッサで使用できる外部ソース（ブラウザなど）への参照が可能になります。OracleAS Wireless は、この <link> 要素を使用した外部の CSS スタイルシートの参照をサポートしています。外部の CSS スタイルシートを参照するには、作成者は、*type* 属性（type="text/css"）および *rel* 属性（rel="stylesheet"）を指定する必要があります（これらの属性が欠落していたり、異なる値が含まれている場合、これらの外部参照は、OracleAS Wireless によって無視されます）。

OracleAS Wireless の MXML Media Attribute モジュール

Media Attribute モジュールは、*media* 属性を定義します。media 属性は、MXML 名前空間の mxml:media 属性（mxml は、MXML 名前空間に対する名前空間接頭辞）に定義します。

mxml:media によって、メディア（デバイスとその機能）の表示条件が要素に割り当てられます。この属性は、現行の要素が表示またはレンダリングされるターゲット・メディア（デバイスとその機能）を指定するコアとなる属性です。mxml:media 属性は、http://www.w3.org/TR/css3-mediaqueries/ に定義されているメディア問合せ構文をサポートしています。サポートされているメディアとその機能のリストについは、メディア固有のコンテンツの埋込みに関する項を参照してください。

注意： mxml:media 属性によって影響を受けるのはレンダリングのみです。この属性によって、要素が文書から削除されることはありません。したがって、イベント・ハンドラやオブザーバを宣言する要素に指定した mxml:media は、文書の一部として残り、mxml:media 属性の値にかかわらずイベント実装に登録されます。mxml:media は、現行のメディア（およびその機能）の内容が mxml:media 属性に指定されている内容と一致しない場合に、現行の要素に 'style="display:none"'CSS プロパティを指定するショートカットと考える必要があります。

Speech Recognition Grammar モジュール

XHTML の Speech Recognition Grammar モジュールは、次の要素を定義します。

- <grammar>: 構文要素のコンテンツ・モデルと名前空間は、Speech Recognition Grammar Specification Version 1.0（http://www.w3.org/TR/speech-grammar/）によって定義されています。この音声認識構文は XML で定義されており、すべての要素が音声認識構文の名前空間に定義されています。この XML のルート要素は <grammar> です。この <grammar> 要素は、XHTML モジュール化仕様の定義に従い、モジュールとして XHTML スキーマに追加されました。

OracleAS Wireless では、音声デバイス用の inline 構文をサポートするために、<grammar> 要素が、XHTML モジュールとして追加されました。<grammar> 要素は、XHTML 文書の head セクションで指定できます。複数の構文を宣言できます。また、<grammar> 要素は、XForms UI コントロールの <extension> 要素内で指定できます。

<a>、<li>（<nl> 内）、UI コントロールなどの要素は、<object> 要素を使用して、<head> セクション内の複数の <grammar> を参照できます。head セクションに定義されている構文は、文書のフラグメント ID（#id）を使用し、<object> によって参照できます。次に、XHTML 文書に含まれている、このような構文の例を示します。この例では、構文をリンク（<a>）要素に関連付けています。

```
<html xmlns="http://www.w3.org/1999/html"
          xmnls:grammar="http://www.w3.org/2001/06/grammar"
                            .....>
       <head>
         <meta name=".." content="..."/>
         <title>...</title

         <grammar id="g1" xmlns="http://www.w3.org/2001/06/grammar"
                  xml:lang="en" version="1.0" mode="voice">
            <rule id="flight">
               ...
              <ruleref uri="#city"/>
            </rule>

            <rule id="city">
                <one-of>
             <item>New York</item>
             <item>Los Angeles</item>
             <item>Chicago</item>
               ...
                </one-of>
            </rule>
               ...
          </grammar>

          <grammar id="g2" xmlns="http://www.w3.org/2001/06/grammar" ....>
                 another grammar
           </grammar>
         </head>
         <body>
              ....
           <a href="hello.jsp"> Hello
             <object data="#id_of_grammar_element_in_head_section"
                   type="application/vnd.oracle.srgs+xml">
               <param name="scope" value="dialog"/>
                    ..
                  <!-- all other attributes supported by <grammar> element
                     as defined in VoiceXML-->
               <param name="....." value="....">
             </object>
               ....
           </a>
         </body>
      </html>
```

B

メディア・タイプとその機能

この付録では、メディア・タイプとその機能について説明します。各項の内容は、次のとおりです。

OracleAS Wireless の CSS メディア問合せと MXML メディア属性構文

CSS メディア問合せ構文は、次の 2 つの部分で構成されています。

1. メディア・タイプ : 特定のメディアを CSS2 仕様の定義に従って指定します。メディア・タイプの値の例には、screen、handheld、tty、tv、aural などがあります。
2. メディア機能 : ターゲット・メディアの特定機能（色やサウンドの機能など）を指定します。

CSS Media Queries では、スタイルの問合せ構文でメディアのタイプと機能をともに使用できます。次に例を示します。

```
media="handheld and (color)"
```

この問合せ構文では、not 条件および only 条件（指定がない場合のデフォルトは only）を指定することもできます。次に例を示します。

```
mxml:media="not all and (color)"
```

メディア問合せ構文は、http://www.w3.org/TR/css3-mediaqueries の CSS3 Media Queries に定義されています。

OracleAS Wireless でサポートされるメディア・タイプ

OracleAS Wireless では、次のメディア・タイプが（`mxml:media` 属性で）サポートされています。

- all: すべてのメディア・タイプ。
- screen: コンピュータ画面、連続、ビジュアル、対話型と静的画面の両方。
- handheld: 携帯情報端末、連続とページ化の両方、対話型と静的画面。
- tty: 表示機能に制限があるポータブル・デバイス、連続（ページ化されていない）で固定ピッチの文字グリッド。
- aural: オーラル / 音声インタフェース用。

OracleAS Wireless でサポートされるメディア機能

CSS3 Media Queries 仕様に指定されているメディア機能

一連のメディア機能は、CSS3 Media Queries 仕様で指定されています。OracleAS Wireless でサポートされるメディア機能は次の機能のみです。

- メディア機能 : color

  説明 : ターゲット・デバイスでサポートされている色コンポーネント当たりのビット数を指定します。

  最小プロパティ : min-color

  最大プロパティ : max-color

  値 : `<integer>`

  適用 : ビジュアル・メディア

- メディア機能 : monochrome

  説明 : ターゲット（モノクローム）デバイスでサポートされているピクセル当たりのビット数を指定します。

  最小プロパティ : min-monochrome

  最大プロパティ : max-monochrome

  値 : `<integer>`

  適用 : ビジュアル・メディア

- メディア機能 : device-width

  説明 : ターゲット・デバイス画面の幅の要件を指定します。

  最小プロパティ : minidevice-width

  最大プロパティ : max-device-width

  値 : <length>（長さの単位は、in、cm、mm、pt、pc、px、em がサポートされます）

  適用 : ビジュアル・メディア

- メディア機能 : device-height

  説明 : ターゲット・デバイスの画面の高さの要件を指定します。

  最小プロパティ : min-device-height

  最大プロパティ : max-device-height

  値 : <length>（長さの単位は、in、cm、mm、pt、pc、px、em がサポートされます）

  適用 : ビジュアル・メディア

- メディア機能 : device-aspect-ratio

  説明 : ターゲット・デバイスのアスペクト比の要件を指定します。

  最小プロパティ : min-device-aspect-ratio（"min-device-aspect-ratio:1" は、縦長を意味します）

  最大プロパティ : max-device-aspect-ratio（"max-device-aspect-ratio:1" は、横長を意味します）

  値 : <ratio>（<integer>/<integer>）

  適用 : ビジュアル・メディア

- メディア機能 : grid

  説明 : ターゲット・デバイスのビットマップに対するグリッドの要件を指定します。

  最小プロパティ : なし

  最大プロパティ : なし

  値 : <integer>（1 の値は、デバイスの表示タイプがグリッドベースであることを、0（ゼロ）の値は、表示タイプがビットマップであることを示します）

  適用 : ビジュアル・メディア

メディア機能の拡張セット

次に、OracleAS Wireless で定義されたメディア機能の拡張セットを示します。

- メディア機能 : paged

  説明 : 連続メディアに対するページ・メディアを指定します。

  最小プロパティ : なし

  最大プロパティ : なし

  値 : <integer>。1 の値はページ・デバイス・メディア（WML 電話機やプリンタなど）を、0（ゼロ）の値はデスクトップ画面などの連続デバイス・メディアを示します。

  適用 : ビジュアル・メディア

OracleAS Wireless で定義された機能

デバイス / ソフトウェアの UA 機能

- メディア機能 : text-input

  説明 : デバイスがテキスト入力に対応しているかどうかを指定します。デバイスには、少なくとも電話のキーパッドがあることが前提です。

  最小プロパティ : なし

  最大プロパティ : なし

  値 : <integer>。1 の値は、デバイスがテキスト入力をサポートしていることを、0（ゼロ）の値は、デバイスが電話のキーパッドのみをサポートしていることを示します。

  適用 : すべてのメディア

- メディア機能 : keyboard

  説明 : キーボード・サポートのタイプを指定します。

  最小プロパティ : なし

  最大プロパティ : なし

  値 : PhoneKeypad | Qwerty | Disambiguating

  適用 : すべてのメディア

- メディア機能 :tables

  説明 : ターゲット・メディアがレイアウト（表、グリッド）のレンダリングをサポートしているかどうかを指定します。

  最小プロパティ : なし

  最大プロパティ : なし

  値 : <integer>。1 の値は、デバイスがレイアウトのレンダリングをサポートしていることを、0（ゼロ）の値は、レイアウトのレンダリングを固有にサポートしていないことを示します。

  適用 : ビジュアル・メディア

- メディア機能 : speech-grammar

  説明 : ターゲット・プラットフォームが音声構文をサポートしているかどうかを指定します。

  最小プロパティ : なし

  最大プロパティ : なし

値 : <integer>。1 の値は、デバイスが音声構文をサポートしていることを、0（ゼロ）の値は、デバイスには音声構文の機能がないことを示します。

適用 : すべてのメディア

- メディア機能 : text-to-speech

  説明 : ページをオーラル・モードでレンダリングする際のテキスト音声合成の機能が、ターゲット・プラットフォームにあるかどうかを指定します。

  最小プロパティ : なし

  最大プロパティ : なし

  値 : <integer>。1 の値は、デバイスがテキスト音声合成をサポートしていることを、0（ゼロ）の値は、デバイスにはテキスト音声合成の機能がないことを示します。

  適用 : オーラル・メディア

- メディア機能 : record-speech

  説明 : ターゲット・プラットフォームが音声録音をサポートしているかどうかを指定します。

  最小プロパティ : なし

  最大プロパティ : なし

  値 : <integer>。1 の値は、デバイスで音声録音が可能なことを、0（ゼロ）の値は、デバイスには音声録音の機能がないことを示します。

  適用 : すべてのメディア

ネットワークの機能と特性

- メディア機能 : async

  説明 : ターゲット・デバイスのネットワーク・モードを指定します。

  最小プロパティ : なし

  最大プロパティ : なし

  値 : <integer>。1 の値は、デバイスが現在メッセージ・モードで動作中であることを示します。このモードでは、リクエストとレスポンスがリモート・リソースへのブロック・コールとはなりません。0（ゼロ）の値は、デバイスが同期モードで動作中であることを意味します。このモードでは、各リクエストにレスポンスがあります。

  適用 : すべてのメディア

- メディア機能 : voice-call

  説明 : デバイス・エージェントから起動するボイス呼出しがネットワークで可能かどうかを指定します。

最小プロパティ : なし

最大プロパティ : なし

値 : <integer>。1 の値は、ボイス呼出しの起動が可能であることを、0（ゼロ）の値は、ボイス呼出しの起動が不可能であることを示します。

適用 : すべてのメディア

- メディア機能 : call-control

  説明 : デバイスのエージェントから起動するボイス呼出しをネットワークで呼出し制御できるかどうかを指定します。呼出し制御によって転送などの機能が有効となります。

  最小プロパティ : なし

  最大プロパティ : なし

  値 : <integer>。1 の値は、デバイスが呼出し制御をサポートしていることを、0（ゼロ）の値は、呼出し制御をサポートしていないことを示します。

  適用 : すべてのメディア

- メディア機能 : email

  説明 : デバイスのエージェントから起動する電子メール・メッセージがネットワークで可能かどうかを指定します。

  最小プロパティ : なし

  最大プロパティ : なし

  値 : <integer>。1 の値は、電子メールの起動が可能であることを、0（ゼロ）の値は、電子メールの起動が不可能であることを示します。

  適用 : すべてのメディア

- メディア機能 : content-length

  説明 : ネットワークでサポートされているコンテンツの最大長と最小長を指定します。

  最小プロパティ : あり

  最大プロパティ : あり

  値 : <Message Length>（サイズは、"content-length: 10" のようにバイト数で指定します）

  適用 : すべてのメディア

メディア問合せの例

表 B-1 メディア問合せの例

デバイス	問合せ
デスクトップ・ブラウザ	@media screen
携帯情報端末（すべての PDA、WML/HDML 電話機、インダストリアル・デバイス、XHTML 電話機用ブラウザ）	@media handheld
電話機の HDML/WML ブラウザ	@media handheld および（tables: 0）
カラー PDA（pocketpc）、カラー XHTML 電話機	@media handheld および（color）
Palm 製品（Palm VII と Palm V）- モノクローム・デバイス	@media handheld および（monochrome）
双方向ポケットベル /SMS デバイス	@media all および（async）
音声デバイス	@media aural

注意： @media handheld は、複数の PDA と HDML/WML 電話機にマップされます。メディアの指定がない場合は、常にすべてのメディアが対象です。

C

XForms 仕様のサポート

この付録では、XForms 仕様のサポートについて説明します。各項の内容は、次のとおりです。

- XForms の文書構造
- XForms の処理モデル
- データ型
- モデル項目プロパティとスキーマ制約
- XForms の XPath 式
- XForms の UI コントロール
- XForms アクション

注意： OracleAS Wireless は、W3C XForms 1.0 Candidate Recommendation（http://www.w3.org/TR/2002/CR-xforms-20021112/）で指定されている機能のサブセットもサポートしています。

XForms の文書構造

表 C-1 XForms の文書構造

XForms 仕様	サポートの有無	コメント
XForms 名前空間	有	
共通属性	有	xsd:ID が XHTML とともにコンテナとしてサポートされます。
リンク属性	一部サポート	instance 要素でのみサポートされます。
単一ノード・バインド属性	有	XForms 要素の label、help、hint、alert または message 要素ではサポートされません（注意：この機能は、label、help、hint、alert または message 要素内で <output> コントロールを使用して実現できます）。
ノード集合バインド属性	有	
model 要素	有	複数のモデルがサポートされます。属性 schema はサポートされません。属性 functions はサポートされません。
instance 要素	有	複数のインスタンスがサポートされます。
submission 要素	有	xforms:submission 内で method="put" はサポートされません。
bind 要素	有	属性 p3ptype はサポートされません。
xsd:schema 要素	有	
mustUnderstand	無	
Extension モジュール	有	

XForms の処理モデル

表 C-2 初期化イベント

XForms 仕様	サポートの有無　コメント
xforms-model-construct	有
xforms-model-initialize	有
xforms-initialize-done	有
xforms-ui-initialize	有
xforms-form-control-initialize	有
xforms-model-destruct	無

表 C-3 対話イベント

XForms 仕様	サポートの有無	コメント
xforms-previous	無	
xforms-next	無	
xforms-focus	無	
xforms-help	有	ユーザー・エージェントがヘルプ・イベントをサポートしている場合にサポートされます。
xforms-hint	有	ユーザー・エージェントがヒント・イベントをサポートしている場合にサポートされます。
xforms-refresh	有	
xforms-revalidate	有	
xforms-recalculate	有	
xforms-reset	有	
xforms-submit	有	

表 C-4 通知イベント

XForms 仕様	サポートの有無	コメント
DOMActivate	有	
xforms-value-changing	無	
xforms-value-changed	有	
xforms-select	無	
xforms-deselect	無	
xforms-scroll-first	無	
xforms-scroll-last	無	
xforms-insert	無	
xforms-delete	無	
xforms-valid	有	
xforms-invalid	有	
DOMFocusIn	無	
DOMFocusOut	無	
xforms-readonly	有	
xforms-readwrite	有	
xforms-required	有	
xforms-optional	有	
xforms-enabled	有	
xforms-disabled	有	
xforms-submit-done	有	
xforms-submit-error	有	

表 C-5 エラー指示

XForms 仕様	サポートの有無	コメント
xforms-bind-exception	無	
xforms-link-exception	無	
xforms-link-error	無	
xforms-compute-exception	無	

データ型

表 C-6 基本的なデータ型

XForms 仕様	サポートの有無	コメント
dateTime	有	
time	有	
date	有	
gYearMonth	有	
gYear	有	
gMonthDay	有	
gDay	有	
gMonth	有	
string	有	
boolean	有	
base64Binary	無	
decimal	有	
anyURI	有	
integer	有	
nonPositiveInteger	有	
negativeInteger	有	
long	有	
int	有	

表 C-6 基本的なデータ型（続き）

XForms 仕様	サポートの有無	コメント
short	有	
byte	有	
nonNegativeInteger	有	
unsignedLong	有	
unsignedInt	有	
unsignedShort	有	
unsignedByte	有	
positiveInteger	有	

表 C-7 XForms のデータ型

XForms 仕様	サポートの有無	コメント
xforms:listItem	有	
xforms:listItems	有	
xforms:dayTimeDuration	有	
xforms:yearMonthDuration	有	

表 C-8 拡張および派生データ型

XForms 仕様	サポートの有無	コメント
normalizedString	無	
hexBinary	無	
float	無	
double	無	
QName	無	
NOTATION	無	
token	無	
language	無	

表 C-8 拡張および派生データ型（続き）

XForms 仕様	サポートの有無	コメント
Name	無	
NCName	無	
ID	無	
IDREF	無	
IDREFS	無	
ENTITY	無	
ENTITIES	無	
NMTOKEN	無	
NMTOKENS	無	

モデル項目プロパティとスキーマ制約

表 C-9 モデル項目プロパティ

XForms 仕様	サポートの有無	コメント
type	有	現行の実装でサポートされているスキーマのデータ型に制限されます。
readonly	有	
required	有	
relevant	有	
calculate	有	
constraint	有	
maxOccurs	有	
minOccurs	有	
p3ptype	無	

表 C-10 スキーマ制約

XForms 仕様	サポートの有無	コメント
スキーマ対応付け	無	
xsi:type	有	
Bind での type 属性	有	
デフォルトの xsi:string	有	

XForms の XPath 式

表 C-11 XPath のデータ型、DOM アクセス、評価コンテキスト

XForms 仕様	サポートの有無	コメント
XPath 1.0 データ型（boolean、string、number、node-set）	有	
インスタンス・データへの DOM アクセス	無	
評価コンテキスト	有	

表 C-12 バインド式

XForms 仕様	サポートの有無	コメント
モデル・バインド式	有	モデル項目プロパティ（bind 要素）で使用する式。
モデル・バインド式での動的な依存関係	無	XForms 仕様では、モデル・バインド式での動的な依存関係の使用を許可していません。
UI バインド式	有	
UI バインド式での動的な依存関係	有	XForms は、UI コントロールのバインド属性での動的なバインド式をサポートしています。OracleAS Wireless は、音声レンダリング・モード以外では、UI コントロールでの動的なバインド式をサポートしています。

表 C-13 XPath のノード集合関数

XForms 仕様	サポートの有無	コメント
count()	有	
id()[1]	有	
last()[1]	有	
position()[1]	有	
name()[1]	有	
local-name()[1]	有	
namespace-uri()[1]	有	
instance()[1]	有	

表 C-14 XPath の文字列関数

XForms 仕様	サポートの有無	コメント
concat()	有	
contains()	有	
normalize-space()[1]	有	
starts-with()	有	
string()	有	
string-length()	有	
substring()	有	
substring-after()	有	
substring-before()	有	
translate()	有	
property()	無	

表 C-15 XPath の数値関数

XForms 仕様	サポートの有無	コメント
ceiling()	有	
floor()	有	
number()	有	
round()	有	
sum()	有	
avg()	有	XForms 仕様で定義
min()	有	XForms 仕様で定義
max()	有	XForms 仕様で定義
count-non-empty()[1]	有	
index()	有	XForms 仕様で定義

表 C-16 XPath のブール関数

XForms 仕様	サポートの有無	コメント
boolean()	有	
false()	有	
lang()[1]	有	
not()	有	
true()	有	
boolean-from-string()	有	XForms 仕様で定義
if()	有	XForms 仕様で定義

[1] これらの XPath 関数を音声（オーラル）モードで使用する場合は注意が必要です。音声（オーラル）モードでは、フォームの相互作用性を向上させるために、スクリプト・モデルを使用して特定のアクションをクライアント側に伝えます。これらの XPath 関数を音声（オーラル）モードで使用する場合、作成者は、サーバー側のサポートを必要とするアクションを使用して、再計算または再検証がサーバーで実行されるようにする必要があります。

表 C-17 XForms/XPath の日時関数

XForms 仕様	サポートの有無	コメント
now()	有	XForms 仕様で定義
days-from-date()	有	XForms 仕様で定義
seconds-from-dateTime()	有	XForms 仕様で定義
seconds()	有	XForms 仕様で定義
months()	有	XForms 仕様で定義

表 C-18 XPath の拡張関数

拡張仕様	サポートの有無	コメント
current()	有	XSLT 仕様で定義されているとおり、current() 関数は、repeat 要素内での反復に便利です。現行のノードを repeat 要素の nodeset 属性に戻します。**注意 :** ref/nodeset 属性（*group*、*setvalue* など）を使用してコンテキスト・ノードを変更した場合、current() は現行の繰返し対象ノードを戻しません。

XForms の UI コントロール

表 C-19 基本的な UI コントロール

XForms 仕様	サポートの有無	コメント
input	有	navindex、accesskey、appearance の各属性はサポートされません。 inputmode 属性はサポートされません。 id、style、class の各属性が追加サポートされます。 slot、modal の各属性が追加サポートされます（オーラル / 音声構文の場合）。 スタイル・ヒントのサイズ、mxml:media の各属性が追加サポートされます。
secret	有	navindex、accesskey、appearance、incremental の各属性はサポートされません。 inputmode 属性はサポートされません。 id、style、class の各属性が追加サポートされます。 slot、modal の各属性が追加サポートされます（オーラル / 音声構文の場合）。 スタイル・ヒントのサイズ、mxml:media の各属性が追加サポートされます。
textarea	有	navindex、accesskey、appearance、incremental の各属性はサポートされません。 inputmode 属性はサポートされません。 id、style、class の各属性が追加サポートされます。 slot、modal の各属性が追加サポートされます（オーラル / 音声構文の場合）。 スタイル・ヒントの行、列、mxml:media の各属性が追加サポートされます。
output	有	appearance 属性はサポートされません。 id、style、class の各属性が追加拡張されます。 デバイス・スタイルのヒント、mxml:media が追加サポートされます。
upload	無	
range	無	

表 C-19 基本的な UI コントロール（続き）

XForms 仕様	サポートの有無	コメント
trigger	有	navindex、accesskey、appearance の各属性はサポートされません。id、style、class の各属性が追加拡張されます。 slot、modal の各属性が追加サポートされます（オーラル / 音声構文の場合）。 デバイス・スタイルのヒント、mxml:media が追加サポートされます。
submit	有	navindex、accesskey、appearance の各属性はサポートされません。id、style、class の各属性が追加拡張されます。 slot、modal の各属性が追加サポートされます（オーラル / 音声構文の場合）。 デバイス・スタイルのヒント、mxml:media が追加サポートされます。
select	有	navindex、accesskey、appearance の各属性はサポートされません。 selection 属性はサポートされません。selection は常にクローズ状態です。 UI 表示をチェックボックスとして選択します。 id、style、class の各属性が追加サポートされます。 slot、modal の各属性が追加サポートされます（オーラル / 音声構文の場合）。 スタイル・ヒントのサイズ、mxml:media の各属性が追加サポートされます。
select1	有	navindex、accesskey、appearance の各属性はサポートされません。 selection 属性はサポートされません。selection は常にクローズ状態です。 UI 表示をラジオ・ボタンとして選択します。 id、style、class の各属性が追加サポートされます。 slot、modal の各属性が追加サポートされます（オーラル / 音声構文の場合）。 スタイル・ヒントのサイズ、mxml:media の各属性が追加サポートされます。
choices	有	

表 C-19 基本的な UI コントロール（続き）

XForms 仕様	サポートの有無	コメント
item	有	item に関する help、hint、alert、actions はサポートされません。
filename	無	
mediatype	無	
value	有	PCDATA のみが子ノードとして value 要素でサポートされます。
label	有	単一ノード・バインド属性はサポートされません（注意 : label 内で <output> を使用して、インスタンスから値を埋め込むことができます)。 リンク属性はサポートされません。
help	有	単一ノード・バインド属性はサポートされません（注意 : help 内で <output> を使用して、インスタンスから値を埋め込むことができます)。 リンク属性はサポートされません。
hint	有	単一ノード・バインド属性はサポートされません（注意 : hint 内で <output> を使用して、インスタンスから値を埋め込むことができます)。 リンク属性はサポートされません。
alert	有	単一ノード・バインド属性はサポートされません（注意 : notification 内で <output> を使用して、インスタンスから値を埋め込むことができます)。 リンク属性はサポートされません。
itemset	有	itemset に関する help、hint、alert、actions はサポートされません。
copy	無	無
extension	有	<extension> 要素は、UI コントロールの <grammar> の宣言をサポートします。 <extension> 要素は <label> 要素をサポートします。これによって、UI コントロールに複数のラベルを設定できます。これらの複数の <label> は、拡張プロンプト・インタフェースとして音声インタフェースで再生されます。

表 C-20 拡張 UI コントロール

XForms 仕様	サポートの有無	コメント
group	有	navindex、accesskey、appearance の各属性はサポートされません。
repeat	有	navindex、accesskey、appearance の各属性はサポートされません。 ネストされた repeat はサポートされません。
repeat の属性	無	
switch/case	有	navindex、accesskey、appearance の各属性はサポートされません。

表 C-21 拡張 UI コントロール

拡張仕様	サポートの有無	コメント
mxml:uiobject	有	MXML 名前空間に定義される拡張 UI コントロールです。uiobject 要素を使用すると、XForms アプリケーションは他の言語で定義されたユーザー・インタフェースを起動できるため、各言語で定義された既存のアプリケーションを再利用できます。 uiobject によって、XForms アプリケーションは uiparam 要素を使用して外部ユーザー・インタフェースとの間でパラメータ（フォーム・データ）の受渡しができます。uiobject のコンテンツ・タイプは、その type 属性で指定されます。外部ユーザー・インタフェースの URI は、<uiobject> の data 属性で指定される一方、フォーム・データのパラメータを外部ユーザー・インタフェースに渡す方法は、method 属性と enctype 属性によって決まります。XForms の <group> については、<uiobject> の単一ノード・バインド属性によって、<uiparam> の子要素に対する XPath コンテキストが設定されます。

表 C-21 拡張 UI コントロール（続き）

拡張仕様	サポートの有無	コメント
mxml:uiparam	有	MXML 名前空間に定義され、<uiobject> の子ノードとして発生する拡張要素です。<uiparam> を使用すると、XForms インスタンスと <uiobject> との間でインスタンス・データの受渡しができます。<uiparam> には、インスタンス・データを <uiobject> への入力値（valuetype="in" の場合）、<uiobject> からの出力値（valuetype="out" の場合）または <uiobject> の data 属性で指定された URI への送信パラメータ（valuetype="submit" の場合）のいずれかにマップする単一ノード・バインド属性が設定されます。<uiparam> の name 属性は、インスタンス・データに関連付けられた外部ユーザー・インタフェースのパラメータ名を識別します。
mxml:uieventmap	有	MXML 名前空間に定義され、<uiobject> の子ノードとして発生する拡張要素です。<uieventmap> を使用すると、外部ユーザー・インタフェースによってスローされたカスタム・イベントを XForms イベントにマップできます。

注意： このリリースでは、<uiobject> は、VoiceXML で定義された UI インタフェースの起動のみサポートします。VoiceXML で実装されたインタフェースにリンクする <uiobject> では、そのインタフェースのタイプが text/x-vxml に設定されている必要があります。

XForms アクション

表 C-22 基本的な UI コントロール

XForms 仕様	サポートの有無	コメント
action	有	更新が遅れているため、サポートされません。
dispatch	無	
rebuild	無	
recalculate	有	
refresh	有	
setfocus	有	スクリプトまたは XForms プラグインをサポートするデバイスでのみサポートされます。このリリースでは、音声（オーラル）デバイスおよびラップトップ・デバイスでサポートされます。
load	有	show="new" はサポートされません。show="new" の動作は show="replace" と同じになります。
setvalue	有	
send	有	
reset	有	
message	有	`src` 属性はサポートされません。 単一ノード・バインド属性（bind/ref 属性）はサポートされません（注意 : help 内で <output> を使用して、インスタンスから値を埋め込むことができます）。

表 C-23 拡張アクション

拡張仕様	サポートの有無	コメント
mxml:disconnect	有	MXML 名前空間に定義される拡張アクションです。OracleAS Wireless では、通常の電話機と VoiceXML ゲートウェイを使用した音声インタフェースをサポートします。このアクションは、通話の切断を実行します。
mxml:handler	有	MXML 名前空間に定義される拡張アクションです。このアクションは、UI コントロールで n 番目に発生したイベントに応じた XForms アクションをサポートします。

D

OracleAS Wireless による CSS のサポート

この付録では、CSS のサポートについて説明します。

OracleAS Wireless による CSS のサポート

表 D-1 CSS のセレクタ

CSS 仕様	サポートの有無	コメント
*	有	
E	有	任意の E 要素（つまり、タイプ E の任意の要素）と一致します。
E F	有	E 要素の子孫である任意の F 要素と一致します。
E > F	有	E 要素の子である任意の F 要素と一致します。
E.warning	有	class="warning"（Class セレクタ）を持つ任意の E 要素と一致します。
E#myid	有	"myid" と等価の id を持つ任意の E 要素と一致します。
#myid	有	"myid" と等価の id を持つ任意の要素と一致します。

注意： このリリースでは、OracleAS Wireless は、CSS セレクタで名前空間をサポートしません。要素セレクタは、任意の名前空間にある（または、名前空間の存在に関係なく）、すべての要素と一致します。XForms と XHTML では、競合する要素名を指定できません。

表 D-2 CSS の at-rules（@ 規則）

CSS 仕様	サポートの有無	コメント
@media（CSS3 メディア問合せ構文）	有	サポートされている値は、"all"、"screen"、"handheld"、"tty"、"aural" です。
@import	有	有

表 D-3 CSS の背景プロパティ

CSS 仕様	サポートの有無	サポートされる値	初期値 / 継承 / 適用 / メディア
background-color	有	<color> | transparent | inherit	初期値 : transparent 継承 : しない 適用 : すべての要素 メディア : ビジュアル・メディア
background-image	有	<uri> | none | inherit	初期値 : none 継承 : しない 適用 : すべての要素 メディア : ビジュアル・メディア
background-position	無		
background-repeat	無		
background-attachment	無		
background	無		

表 D-4 CSS のボーダー・プロパティ

CSS 仕様	サポートの有無	サポートされる値	初期値 / 継承 / 適用 / メディア
border-top-width、 border-right-width、 border-bottom-width、 border-left-width	有	<border-width> | inherit <border-width> => [thin | medium | thick | <length>]	初期値 :medium 継承 : しない 適用 : すべての要素 メディア : ビジュアル・メディア
border-top-color、 border-right-color、 border-bottom-color、 border-left-color	有	<color> | inherit	初期値 : <color プロパティの値 > 継承 : しない 適用 : すべての要素 メディア : ビジュアル・メディア
border-top-style、 border-right-style、 border-bottom-style、 border-left-style	有	<border-style> | inherit <border-style> => [solid | none]	初期値 : none 継承 : しない 適用 : すべての要素 メディア : ビジュアル・メディア

注意： ボーダー・プロパティは、HTML ベースのブラウザでのみサポートされます。また、HTML32 ベースのブラウザの場合、これらのプロパティは表要素とグリッド・レイアウト要素でのみサポートされます。さらに、HTML32 ベースのブラウザでは、四辺（左、右、上、下）のコントロールを個別にサポートしません。HTML32 では、'border-left-*' プロパティを使用して、すべての辺を対象にプロパティをオンまたはオフにできます。

表 D-5 CSS のボックス・プロパティ

CSS 仕様	サポートの有無	サポートされる値	初期値 / 継承 / 適用 / メディア
margin-top、 margin-right、 margin-bottom、 margin-left	有	<margin-width> | inherit 注意 : デバイスがサポートしている場合にサポートされます。HTML32 ベースのブラウザでは、表要素のみがサポートされます。	初期値 : 0 継承 : しない 適用 : すべての要素 メディア : ビジュアル・メディア
padding-top、 padding-right、 padding-bottom、 padding-left	有	<padding-width> | inherit 注意 : デバイスがサポートしている場合にサポートされます。HTML32 ベースのブラウザでは、表要素のみがサポートされます。	初期値 : 0 継承 : しない 適用 : すべての要素 メディア : ビジュアル・メディア

注意： ボックス・プロパティは、HTML ベースのブラウザでのみサポートされます。HTML32 ベースのブラウザの場合、これらのプロパティは表要素とグリッド・レイアウト要素でのみサポートされます。さらに、HTML32 ベースのブラウザでは、四辺（左、右、上、下）のコントロールを個別にサポートしません。HTML32 では、'border-left-*' プロパティを使用して、すべての辺を対象にプロパティをオンまたはオフにできます。

表 D-6 CSS の色プロパティ

CSS 仕様	サポートの有無	サポートされる値	初期値 / 継承 / 適用 / メディア
color	有	<color>	初期値 : #000000 継承 : する 適用 : すべての要素 メディア : ビジュアル・メディア

表 D-7 CSS のフォント・プロパティ

CSS 仕様	サポートの有無	サポートされる値	初期値 / 継承 / 適用 / メディア
font-family	有	[[<family-name> | <generic-family>],]* [<family-name> | <generic-family>] | inherit	継承 : する 適用 : すべての要素 メディア : ビジュアル・メディア
font-size	有	<absolute_size> | inherit <absolute_size> => [xx-small | x-small | small | medium | large | x-large | xx-large]	初期値 :medium 継承 : する 適用 : すべての要素 メディア : ビジュアル・メディア
font-style	有	normal | italic | oblique | inherit	初期値 : normal 継承 : する 適用 : すべての要素 メディア : ビジュアル・メディア
font-weight	有	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit	初期値 : normal 継承 : する 適用 : すべての要素 メディア : ビジュアル・メディア

表 D-8 CSS のレイアウト・プロパティ

CSS 仕様	サポートの有無	サポートされる値	初期値 / 継承 / 適用 / メディア
display	有	inline | block | list-item | none |grid | grid-label | grid-cell | grid-cell-label	初期値 : inline 継承 : する 適用 : すべての要素 メディア : すべてのメディア (**注意 :** "display: none" のみオーラル・メディアに適用可能で、その他の値はビジュアル・メディアに適用されます)。
height	有	<length> | <percentage> | auto | inherit	初期値 : auto 継承 : する 適用 : すべての要素 メディア : ビジュアル・メディア
width	有	<length> | <percentage> | auto | inherit	初期値 : auto 継承 : する 適用 : すべての要素 メディア : ビジュアル・メディア
float	無		
clear	無		
visibility	無		

注意 : height プロパティと width プロパティは、HTML ベースのブラウザでのみサポートされます。HTML32 ベースのブラウザ（Pocket PC、Palm）の場合、これらのプロパティはイメージ、表およびグリッド・レイアウト要素でのみサポートされます。マルチチャネル・アプリケーションでは、通常の場合も特殊な場合も、width と height にパーセントを使用するのが一般的です。

表 D-9 CSS のリスト・プロパティ

CSS 仕様	サポートの有無	サポートされる値	初期値 / 継承 / 適用 / メディア
list-style	有	list-style-type | list-style-position | list-style-image | inherit	継承 : する 適用 : display: list-item を持つ要素 メディア : ビジュアル・メディア
list-style-image	有	<uri> | none | inherit	初期値 : none 継承 : する 適用 : display: list-item を持つ要素 メディア : ビジュアル・メディア
list-style-type	有	disc | circle | square | decimal | lower-roman | upper-roman | lower-alpha | upper-alpha | none | inherit	初期値 : disc 継承 : する 適用 : display: list-item を持つ要素 メディア : ビジュアル・メディア
list-style-position	無		

表 D-10 CSS のテキスト・プロパティ

CSS 仕様	サポートの有無	サポートされる値	初期値 / 継承 / 適用 / メディア
text-align	有	left | right | center | justify | inherit	継承 : する 適用 : すべての要素 メディア : ビジュアル・メディア
text-decoration	有	none | underline | inherit	初期値 : none 継承 : する 適用 : display: すべての要素 メディア : ビジュアル・メディア
text-transform	有	capitalize | uppercase | lowercase | none | inherit	初期値 : none 継承 : する 適用 : すべての要素 メディア : ビジュアル・メディア
text-indent	有	<length> | <percentage> | inherit	初期値 : 0 継承 : する 適用 : すべての要素 メディア : ビジュアル・メディア

表 D-10 CSS のテキスト・プロパティ（続き）

CSS 仕様	サポートの有無	サポートされる値	初期値 / 継承 / 適用 / メディア
vertical-align	有	top | middle | bottom | sub | super	初期値 : baseline 継承 : する 適用 : すべての要素 メディア : ビジュアル・メディア
white-space	有	normal | pre | nowrap | inherit	初期値 : normal 継承 : する 適用 : すべての要素 メディア : ビジュアル・メディア

注意 :

text-align プロパティと vertical-align プロパティは、HTML ベースのブラウザでのみサポートされます。HTML32 ベースのブラウザ（Pocket PC、Palm）の場合は、次の追加制約が適用されます。

`'vertical-align: top | middle| bottom'` は、`div`、`p` および表セルなどのブロック・レベル要素でのみサポートされます。

`'vertical-align: sub | super'` は、`span` や `strong` などのインライン要素でのみサポートされます。

`'text-align: left | right | center'` は、`div`、`p` および表セルなどのブロック・レベル要素でのみサポートされます。

`'text-indent'` は、XHTML MP に準拠したブラウザでのみサポートされます。

`'white-space'` プロパティは、折返しモード（`white-space: normal`）と折返し禁止モードの制御に使用されます。

表 D-11 CSS の改ページ・プロパティ

CSS 仕様	サポートの有無	サポートされる値	初期値 / 継承 / 適用 / メディア
page-break-before	有	auto | always | avoid | inherit	初期値 : auto 継承 : しない 適用 : すべての要素 メディア : ビジュアルおよびページ・メディア

表 D-11 CSS の改ページ・プロパティ（続き）

CSS 仕様	サポートの有無	サポートされる値	初期値 / 継承 / 適用 / メディア
page-break-after	有	auto | always | avoid | inherit	初期値 : auto 継承 : しない 適用 : すべての要素 メディア : ビジュアルおよびページ・メディア
page-break-inside	有	auto | avoid | inherit	初期値 : auto 継承 : しない 適用 : すべての要素 メディア : ビジュアルおよびページ・メディア

注意 : 改ページ・プロパティは、ページ・デバイス（WML/HDML をサポートするブラウザ）で単一のページを複数のカードに分割するために使用します。これらのプロパティは、デッキ（ページ）サイズ制限が存在するデバイスで、サーバーがデッキ（ページ）・オーバーフロー・サポートを処理する方法を制御します。

表 D-12 UI レイアウトに関する Oracle CSS 拡張機能

CSS 仕様	サポートの有無	サポートされる値	初期値 / 継承 / 適用 / メディア
display	有	grid | grid-cell (OracleAS Wireless では、'display' プロパティの拡張値をサポートします)	初期値 : inline 継承 : する 適用 : すべての要素 メディア : ビジュアル・メディア
_orcl-grid-cells	有	<number>	初期値 : 1 継承 : しない 適用 : display: grid を持つ要素 メディア : ビジュアル・メディア
_orcl-grid-cellspan	有	<number | all	初期値 : 1 継承 : しない 適用 : display: grid-cell を持つ要素 メディア : ビジュアル・メディア
_orcl-label-side	有	left | right | top | bottom	初期値 :left 継承 : しない 適用 : display: grid-cell および display: inline を持つ要素 メディア : ビジュアル・メディア

注意 : `'display: inline'` の場合、`_orcl-label-side` プロパティがサポートするのは `'left | right'` のみです。

表 D-13 リスト・コントロール・レイアウトに関する Oracle CSS 拡張機能

CSS 仕様	サポートの有無	サポートされる値	初期値 / 継承 / 適用 / メディア
_orcl-list-orient	有	horizontal | vertical	継承 : する 適用 : リスト・コントロール (select/select1) メディア : ビジュアル・メディア

表 D-14 繰返しレイアウトに関する Oracle CSS 拡張機能

CSS 仕様	サポートの有無	サポートされる値	初期値 / 継承 / 適用 / メディア
_orcl-repeat-labels	有	none | once | always	初期値 : once 継承 : する 適用 : repeat 要素 メディア : ビジュアル・メディア

注意： _orcl-repeat-labels プロパティを適用できるのは、repeat 要素のみです。このプロパティは、繰返し構造を持つラベルまたは UI コントロールの表示を制御します。

表 D-15 その他の Oracle CSS 拡張機能

CSS 仕様	サポートの有無	サポートされる値	初期値 / 継承 / 適用 / メディア
_orcl-table-col-separator	有	<string>	初期値 : "," 継承 : する 適用 : 表および表の行 メディア : "table" メディア機能のないビジュアル・メディア

注意： `_orcl-table-col-separator` プロパティは、表がターゲット・ブラウザで固有にサポートされていない場合に使用します。

表 D-16 CSS のオーラル・プロパティ

CSS 仕様	サポートの有無	サポートされる値	初期値 / 継承 / 適用 / メディア
volume	有	default | silent | soft | medium | loud | inherit	初期値 :default 継承 : する 適用 : すべての要素 メディア : オーラル・メディア
speak	有	normal | spell-out | date | time | currency | duration | telephone | net | address| inherit | measurement | name	初期値 : normal 継承 : する 適用 : すべての要素 メディア : オーラル・メディア
speak-numeral	有	digits | continuous | ordinal | inherit	初期値 : continuous 継承 : する 適用 : すべての要素 メディア : オーラル・メディア
speak-header	有	once | always | inherit	初期値 : once 継承 : する 適用 : 表ヘッダーを持つ要素 メディア : オーラル・メディア
pause-after	有	<time> | none | small | medium | large | inherit	初期値 : none 継承 : しない 適用 : すべての要素 メディア : オーラル・メディア
pause-before	有	<time> | none | small | medium | large | inherit	初期値 : none 継承 : する 適用 : すべての要素 メディア : オーラル・メディア
speech-rate	有	slow | medium | fast | default | inherit	初期値 :default 継承 : しない 適用 : すべての要素 メディア : オーラル・メディア
pitch	有	low | medium | high | default | inherit	初期値 :default 継承 : する 適用 : すべての要素 メディア : オーラル・メディア

表 D-16 CSS のオーラル・プロパティ（続き）

CSS 仕様	サポートの有無	サポートされる値	初期値 / 継承 / 適用 / メディア
pitch-range	有	low | medium | high | default | inherit	初期値 :default 継承 : する 適用 : すべての要素 メディア : オーラル・メディア
stress	有	none | reduced | moderate | strong | inherit	初期値 : none 継承 : する 適用 : すべての要素 メディア : オーラル・メディア

表 D-17 オーラルに関する Oracle CSS 拡張機能

CSS 仕様	サポートの有無	サポートされる値	初期値 / 継承 / 適用 / メディア
_orcl-bargein	有	true | false | inherit	初期値 :true 継承 : する 適用 : すべての要素 メディア : オーラル・メディア
_orcl_sayas_format	有	none | date(dmy | mdy | ymd | ym | my | md | y) | time(hm | hms) | duration(hm | hms | ms | h | m | s) | net (email | url)	初期値 : none 継承 : する 適用 : すべての要素 メディア : オーラル・メディア
_orcl-duration	有	<time>	継承 : する 適用 : "speak: duration" を持つすべての要素 メディア : オーラル・メディア
_orcl-pitch-contour	有	<string> 例 : "(0%,+20)(10%,+30%)(40%,+10)"	継承 : する 適用 : "speak: duration" を持つすべての要素 メディア : オーラル・メディア
_orcl-aural-props	有	vxml-gateway-property-name(property-value) [,vxml-gateway-property-name(property-value)]*	継承 : する 適用 : すべての要素 メディア : オーラル・メディア

E

CSS のレイアウト・プロパティの使用

この付録では、CSS の機能の使用方法について説明します。各項の内容は、次のとおりです。

- OracleAS Wireless の CSS レイアウト拡張機能 : 新規プロパティと値
- グリッド・レイアウト・モデル
- XForms グループのデフォルト・スタイル

OracleAS Wireless の CSS レイアウト拡張機能 : 新規プロパティと値

OracleAS Wireless では、XHTML+XForms 文書を視覚的にレンダリングする際に、フォーム・コントロールをレイアウトするために、CSS の拡張プロパティが定義されています。

グリッド・レイアウト・モデルをサポートするために、OracleAS Wireless は、次の値を使用して CSS の 'display' の値空間を拡張します。これらの値は、CSS 仕様で定義されている値に追加されています。

```
'display' : grid | grid-cell
```

さらに、OracleAS Wireless では、次の CSS プロパティが追加定義されています。

- '_orcl-grid-cells'

  値 : <number>

  初期値 : 1

  適用 : 'display: grid' を持つすべての要素

  継承 : しない

  パーセント : N/A

  メディア : ビジュアル

- '_orcl-grid-cellspan'

  値 : <number> | all

  初期値 : 1

  適用 : 'display: grid-cell' を持つすべての要素

  継承 : しない

  パーセント : N/A

  メディア : ビジュアル

- '_orcl-label-side'

  値 : left | right | top | bottom

  初期値 :left

  適用 : 'display: grid-cell' および 'display: inline' を持つ label 要素

  継承 : しない

  パーセント : N/A

  メディア : ビジュアル

グリッド・レイアウト・モデル

グリッド・コントロールは、複数行にレイアウトされたセルのグリッドを包含している、生成されたボックスです。CSS プロパティ 'display: grid' を持つ要素は、グリッド・コントロールを生成します。グリッド・コントロール内の各セルは、グリッドセルと呼ばれます。行内のグリッドセルの数は、CSS プロパティ '_orcl-grid-cells' によって決まります。グリッド・コントロールの子要素はそれぞれ、CSS の 'display' プロパティが 'grid-cell'（style="display: grid-cell"）に設定されている場合に、そのコンテンツをグリッドセル内にレイアウトします。文書の順序によって、特定の要素が占有する適切なグリッドセルが決まります。CSS プロパティ '_orcl-grid-cellspan' に応じて、要素は複数のグリッドセルにまたがる場合があります。

グリッド・レイアウトの例：

```
<xforms:group style="display: grid; _orcl-grid-cells: 2">
  <p style="display: grid-cell">
      Content of 'grid-cell' 1
  </p>
  <p style="display: grid-cell">
      Content of 'grid-cell' 2
  </p>
  <p style="display: grid-cell">
      Content of 'grid-cell' 3
  </p>
<xforms:group>
```

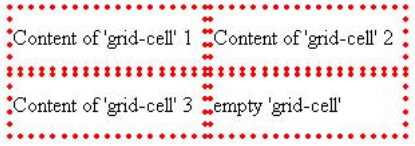
グリッドセル・レイアウトとセル範囲

grid 内に 'display: grid-cell' を持つ要素は、複数のセルにまたがることができ、またがるセル数は、プロパティ '_orcl-grid-cellspan' によって制御されます。

グリッド・レイアウトとセル範囲の例：

```
<xforms:group style="display: grid; _orcl-grid-cells: 2">
  <p style="display: grid-cell">
      Content of 'grid-cell' 1
  </p>
  <p style="display: grid-cell">
      Content of 'grid-cell' 12
  </p>
  <p style="display: grid-cell;_orcl-grid-cellspan: all">
      Content of 'grid-cell' 3
  </p>
<xforms:group>
```

図 E-1 グリッド・レイアウトとセル範囲の結果



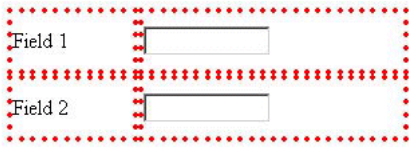
グリッドセルとグリッドセル・ラベル

グリッド・コントロールの子要素には、CSS の display の値 'grid-cell' を設定することも、別のグリッド・レイアウト・コントロールを設定することもできます。グリッドセルには、必要に応じて、CSS プロパティ 'display: grid-cell' で識別される要素を設定できます。'display: grid-cell' を持つ要素は、グリッド構造内で別々のグリッドセルを占有します。その例として、<xforms:label> を持つ UI コントロール <xforms:input> があります。このコントロールは、その UI コントロール自体が占有するセルとは異なる、別のグリッドセルを占有します。

'grid-cell' と 'grid-cell-label' の例:

```
<xforms:group style="display: grid; _orcl-grid-cells: 2">
  <xforms:input style="display: grid-cell">
     <xforms:label style="display: grid-cell">
        Field 1
     </xforms:label>
  </xforms:input>
  <xforms:input style="display: grid-cell">
     <xforms:label style="display: grid-cell">
        Field 2
     </xforms:label>
  </xforms:input>
<xforms:group>
```

図 E-2 'grid-cell' と 'grid-cell-label' の結果



UI コントロール自体とは異なるセルに <label> を配置する必要はありません。ラベルを UI コントロールと同じセルに配置するには、CSS の 'display' プロパティを 'inline'（'display: inline'）に設定します。

インライン・ラベルの例 :

```
<xforms:group style="display: grid; _orcl-grid-cells: 2">
  <xforms:input style="display: grid-cell">
     <xforms:label style="display: inline">
        Field 1
     </xforms:label>
  </xforms:input>
  <xforms:input style="display: grid-cell">
     <xforms:label style="display: inline">
        Field 2
     </xforms:label>
  </xforms:input>
<xforms:group>
```

図 E-3 インライン・ラベルの結果

グリッドセル内のインライン・コンテンツ

グリッド・コントロールのすべての子要素が、それぞれのコンテンツを別々のグリッドセルに配置するわけではありません。'display' プロパティが inline に設定された子要素は、そのコンテンツを先行する要素によって作成されたセルにレイアウトします。

グリッド・レイアウトにおける 'display: inline' の例 :

```
<xforms:group style="display: grid; _orcl-grid-cells: 2">
  <xforms:input style="display: grid-cell">
     <xforms:label style="display: grid-cell-label">
        Field 1
     </xforms:label>
  </xforms:input>
  <xforms:input style="display: grid-cell">
     <xforms:label style="display: grid-cell-label">
        Field 2
     </xforms:label>
  </xforms:input>
  <p style="display: inline">
     Some text
  </p>
<xforms:group>
```

図 E-4 グリッド・レイアウトにおける `'display: inline'` **の結果**

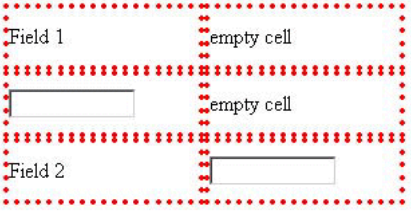
グリッドセル・ラベルのラベル位置

OracleAS Wireless では、CSS の新規プロパティ `'_orcl-label-side'` が定義されています。`_orcl-label-side` は、グリッドセルのコンテンツに対するラベルの位置を制御します。`_orcl-label-side` に指定可能な値は、`left | right | top | bottom` です。

グリッドセルのラベル位置の例：

```
<xforms:group style="display: grid; _orcl-grid-cells: 2">
  <xforms:input style="display: grid-cell">
     <xforms:label
         style="display: grid-cell; _orcl-label-side: top">
        Field 1
     </xforms:label>
  </xforms:input>
  <xforms:input style="display: grid-cell; ">
     <xforms:label
        style="display: grid-cell; _orcl-label-side: left">
        Field 2
     </xforms:label>
  </xforms:input>
<xforms:group>
```

図 E-5 グリッドセルのラベル位置の結果



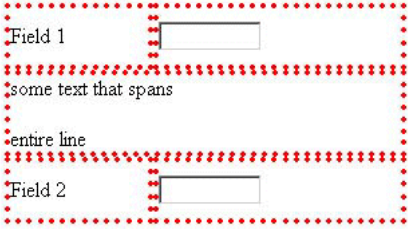
XForms グループのデフォルト・スタイル

OracleAS Wireless では、グリッド・レイアウト・モデルに基づいて、XForms グループ要素のデフォルトのスタイルシートが定義されています。デフォルトのスタイルシートでは、XForms グループは、グループのラベルを上（_orcl-label-side: top）に設定したグリッド・レイアウト（display: grid）としてレンダリングされます。XForms グループのセル数は、デフォルトで 2（_orcl-grid-cells: 2）になります。また、デフォルトでは、グループ内のすべてのフォーム・コントロールとそのラベルは、それぞれ 1 つのグリッドセルを占有し、フォーム・コントロールのラベルは、左側に配置されます。

デフォルトの CSS スタイルシートを使用した例：

```
<xforms:group>
     <xforms:label>Grid label</xforms:label>
     <xforms:input>
         <xforms:label>
           Field 1
        </xforms:label>
     </xforms:input>
     <div> some text that spans
          <p> entire line </p>
     </div>
     <xforms:input>
         <xforms:label>
           Field 2
         </xforms:label>
     </xforms:input>
   <xforms:group>
```

図 E-6 デフォルトの CSS スタイルシートを使用した結果



F

Oracle XML Grammar Subset

この付録では、Oracle XML Grammar Subset について説明します。

Oracle XML Grammar Subset

Oracle Grammar Subset（OGS）は、W3C Speech Recognition Grammar Format（SRGS）の XML Form のサブセットです。このサブセットの主な目的は次のとおりです。

- OGS の構文を自己完結型にすることで、他の構文への参照を解決しなくても他のフォーマットに変換できるようにします。
- すべてのゲートウェイでサポートされているとは限らない複数エントリ・ポイントや再帰的規則の参照などの機能を除外します。
- 二重引用符の区切り記号付きのトークンなどの構文を除外します。
- 重み、繰返し確率など、Speech Recognition Grammar Specification（SRGS）でセマンティクスが明確に定義されていない機能を除外します。
- セマンティクス解析の使用を制限します。
- XML Form 構文および実装間でセマンティクスが変化する可能性のある（曖昧な構文などの）セマンティクス解析に関する特定の機能、および未定義の識別子への参照を除外します。

OGS の構文はスタンドアロンの XML 文書ではないため、XML ヘッダーや DOCTYPE を持たず、ルート `<grammar>` 要素とコンテンツのみで構成されています。このような `<grammar>` 要素とコンテンツは、W3C Speech Recognition Grammar Format の DTD に従った適切なものである場合、および次の追加制限を満たす場合にのみ、サブセット内に存在します。

- `<grammar>` 要素の *root* 属性が定義されており、その属性は構文内のいずれか 1 つの `<rule>` への参照であること（したがって、構文には最低 1 つの `<rule>` が定義されていること）。`<grammar>` の *root* 属性によって参照される規則を「ルート規則」と呼びます。
- 定義済のすべての *scope* 属性に値 private が設定されていること。
- すべての `<ruleref>` 要素に *uri* 属性が設定され、その値は # で始まること。つまり、それらの要素が同じ構文内の他の規則への参照であること。`<ruleref>` に *type*、*alias* または *special* のいずれの属性も設定されていないこと。
- 規則の参照にループが存在しないこと。つまり、
  - 規則に、それ自体を参照する `<ruleref>` が含まれていないこと。
  - *ri*（*i = 1, ..., n-1* のすべての場合）に、*ri+1* を参照する `<ruleref>` が含まれ、*rn* に、*r1* を参照する `<ruleref>` が含まれるような規則の順序 *r1, ..., rn* が存在しないこと。
- 二重引用符の区切り記号付きのトークンが存在せず、複数のワードで構成されるトークンはすべて、`<token>` タグで区切られていること。
- 構文内で発生する実体参照は、*&*、*<*、*>*、*"* および *'* のみであり、これらは 10 進数または 16 進数コードを使用して表現されていること。

- <example>、<alias>、<meta>、<metadata>、<lexicon> のいずれの要素も存在しないこと。
- 要素に *weight*、*repeat-prob* のいずれの属性も設定されていないこと。
- Speech Recognition Grammar Specification で定義されているとおり、構文が曖昧でないこと。
- 規則で認識する発声には、その規則内で発生する最大 1 つの <ruleref> が含まれ、特定の *uri* 属性値が指定されていること（つまり、同じ規則を参照する複数の <ruleref> が含まれる発声を認識可能な規則が存在しないこと）。
- すべての <tag> 要素には、W3C Semantic Interpretation for Speech Recognition（SISR）で説明されている書式のサブセットによるテキストが含まれ、次の追加制限を満たしていること。
  - 各 <tag> のコンテンツは次のいずれかであること。
    - 二重引用符の区切り記号付きの文字列要素。
    - 構文規則参照属性。つまり、$ の後に <rule> の *id* 属性が続くこと。
    - 構文規則参照属性のプロパティ。つまり、構文規則参照属性の後にピリオド（.）とプロパティ名が続くこと。
    - Semantic Interpretation 仕様で定義されている AssignmentList（セミコロンで区切られた割当てリスト）。各割当ての左辺は、Semantic Interpretation 仕様の定義どおり識別子であり、右辺は、前述の 3 つのタイプの表現（二重引用符の区切り記号付きの文字列要素、構文規則参照属性、構文規則参照属性のプロパティ）のいずれか 1 つになります。

      たとえば、次の行はすべて、<tag> のコンテンツとして認められます。

      ```
      "sausage"
      $gender
      $animal.species
      what = "animal"
      where = "New York City";
      ; when = "now"
      who = "me"; why = "because"
      question = "how" ; how = "by hook or by crook";
      ;;
      food = $order
      diameter = $order.size
      what = "pizza"; size = $order.diameter
      ```

  - ルート規則でない規則は、次のいずれかのカテゴリに分類されること。
    - 規則に <tag> が含まれない。このような規則を「void 規則」と呼びます。

  * 規則によって認識された各発声に、その規則内の <tag> が 1 つのみ含まれる。この <tag> は、発声に含まれている規則の最後の部分で、<tag> のコンテンツは、二重引用符の区切り記号付きの文字列、構文規則参照属性または構文規則参照属性のプロパティのいずれかである。このような規則を「文字列規則」と呼びます。
  * 規則によって認識された各発声に、その規則内の <tag> が 1 つのみ含まれる。この <tag> は、発声に含まれている規則の最後の部分で、<tag> のコンテンツは、AssignmentList である。このような規則を「構造規則」と呼びます。

- 構造規則内の各 <tag> が、完全に同じ識別子のセットに割り当てられていること（つまり、完全に同じ識別子が、各 <tag> のコンテンツ内にある割当ての左辺に表示されること）。構造規則内で <tag> によって割り当てられた識別子のセットを「規則のシグネチャ」と呼びます。
- <tag> のコンテンツに構文規則参照属性のプロパティが含まれている場合、参照される規則は構造規則であり、そのプロパティは参照される規則のシグネチャ内に存在すること。たとえば、<tag> に次のコンテンツが設定されているとします。

```
$y.z
```

または

```
x = $y.z
```

  id *y* を持つ規則は構造規則であり、規則 *y* 内のすべての <tag> は、識別子 z への割当てを含む必要があります。

- <tag> のコンテンツに、プロパティが指定されていない構文規則参照属性が含まれる場合、参照される規則は文字列規則であること。たとえば、<tag> に次のコンテンツが設定されているとします。

```
$y
```

または

```
x = $y
```

  id *y* を持つ規則は文字列規則であることが必要です。

- ルート規則の各 <tag> には、AssignmentList が含まれること（構造規則という用語は、ルート規則以外の規則にのみ適用されるため、ルート規則は、構造規則に関する前述の制限を受けません）。
- 発声が規則（ルート規則またはそれ以外）によって認識された場合、その発声に含まれている規則の <tag> のコレクションには、同じ識別子に対する複数の割当てがないこと（識別子を割り当てる式が同じ場合でも）。
- 規則（ルート規則またはそれ以外）によって認識された発声に、その <rule> の <tag> が含まれており、さらに、その <tag> に構文規則参照属性 *$id*（スタンドアロンまたはプロパティの一部）が含まれている場合、発声には <tag> と同じ規則の <ruleref> が含まれ、その発声の <tag> の前に、uri= *#id* を伴うこと。

G

JSP タグ・ライブラリ

Mobile Studio のページは、J2EE JSP テクノロジを使用して作成されているため、カスタム・タグ・ライブラリの機能を利用できます。この付録では、Mobile Studio のページをカスタマイズするときに使用できるカスタム・タグについて説明します。

> **注意：** JSP タグ・ライブラリを使用するのは、Mobile Studio をカスタマイズする場合のみです。Wireless アプリケーションの開発には必要ありません。

この付録では、次の JSP タグのリストを記載します。

- <om:is />
- <om:not>
- <om:get />
- <om:bean />
- <om:test />
- <om:equals />
- <om:indexIs />
- <om:indexEquals />
- <om:index />
- <om:res />
- <om:enc />
- <om:exist />
- <om:notexist />
- <om:if />

- <om:elseif>
- <om:else />
- <om:then />
- <om:iterate />
- <om:switch />
- <om:case />
- <om:default />

表 G-1 <om:is />

taglib での仕様	使用方法
<tag> <name>is</name> <attribute> <name>attr</name> <required>true</required> <rtexprvalue>true</rtexprvalue> </attribute> <attribute> <name>value</name> <required>true</required> <rtexprvalue>true</rtexprvalue> </attribute> <attribute> <name>name</name> <required>false</required> <rtexprvalue>true</rtexprvalue> </attribute> <attribute> <name>prefix</name> <required>false</required> <rtexprvalue>true</rtexprvalue> </attribute> <tagclass>oracle.panama.studio.taglib.IsTag</tagclass> <bodycontent>JSP</bodycontent> <info>Evaluate the body if the value found from the bean matches the given value</info> </tag>	<om:is attr= *attrName* value= *attrValue* (name= *beanName*) (prefix= *prefixName*) >.*</om:is>* **注意 :** カッコ内の属性はオプションです。 *<om:is>* は本体タグです。つまり、条件が true に評価されると、本体のコンテンツが処理され、必要に応じて出力されます。 *attr* は、値をテストする Bean の属性です。 *value* は、テストの対象となる Bean の属性の値です。 *name*（オプション）は、コンテキスト内の Bean の名前です。 *prefix*（オプション）。Bean に属性を要求すると、その属性の名前に接頭辞が追加されます（たとえば、接頭辞が *is* で、属性が *empty* の場合、Bean を起動するメソッドは Java のネーミング規則に従って isEmpty() となります）。この属性を指定しない場合は、最初の *get* がテストされてから、*is* が接頭辞として扱われます。属性をまったく指定しないと、例外がスローされます。

表 G-2 <om:not>

taglib での仕様	使用方法
<tag> <name>not</name> <attribute> <name>attr</name> <required>true</required> <rtexprvalue>true</rtexprvalue> </attribute> <attribute> <name>value</name> <required>true</required> <rtexprvalue>true</rtexprvalue> </attribute> <attribute> <name>name</name> <required>false</required> <rtexprvalue>true</rtexprvalue> </attribute> <attribute> <name>prefix</name> <required>false</required> <rtexprvalue>true</rtexprvalue> </attribute> <tagclass>oracle.panama.studio.taglib.NotTag</tagclass> <bodycontent>JSP</bodycontent> <info>Evaluate the body if the value found from the bean does not match the given value.</info> </tag>	<om:not attr= *attrName* value= *attrValue* (name= *beanName*) (prefix= *prefixName*) >.</*om:not*> **注意 :** カッコ内の属性はオプションです。 <*om:not*> は本体タグです。つまり、条件が true に評価されると、本体のコンテンツが処理され、必要に応じて出力されます。 *attr* は、値をテストする Bean の属性です。 *value* は、テストの対象となる Bean の属性の値です。 *name*（オプション）は、コンテキスト内の Bean の名前です。 *prefix*（オプション）。Bean に属性を要求すると、その属性の名前に接頭辞が追加されます（たとえば、接頭辞が *is* で、属性が *empty* の場合、Bean を起動するメソッドは Java のネーミング規則に従って `isEmpty()` となります）。この属性を指定しない場合は、最初の *get* がテストされてから、*is* が接頭辞として扱われます。属性をまったく指定しないと、例外がスローされます。

表 G-3 <om:get />

taglib での仕様	使用方法
<tag> <name>get</name> <attribute> <name>attr</name> <required>true</required> <rtexprvalue>true</rtexprvalue> </attribute> <attribute> <name>name</name> <required>false</required> <rtexprvalue>true</rtexprvalue> </attribute> <tagclass>oracle.panama.studio.taglib.GetTag</tagclass> <bodycontent>empty</bodycontent> <info>Gets the value of a bean attribute using reflection.</info> </tag>	<om:get attr= *attrName* (name= *beanName*)/> *<om:get>* は単純タグです。つまり、このタグの本体にはコンテンツを挿入できません。このタグは、Bean から属性を取得し、見つかった場合はそれを出力します。Bean または属性が見つからない場合は何もしません。 *attr* は、値をテストする Bean の属性です。 *name*（オプション）は、コンテキスト内の Bean の名前です。

表 G-4 <om:bean />

taglib での仕様	使用方法
<tag> <name>bean</name> <attribute> <name>name</name> <required>true</required> <rtexprvalue>true</rtexprvalue> </attribute> <tagclass>oracle.panama.studio.taglib.BeanTag</tagclass> <bodycontent>JSP</bodycontent> <info>Sets up the context for the bean.</info> </tag>	<om:bean name= *beanName*/> *<om:bean>* は単純タグです。つまり、このタグの本体にはコンテンツを挿入できません。このタグは、指定された名前で Bean をコンテキスト（JSP ページのコンテキストと同じ）に入れます。 *attr* は、テストの対象となる Bean の属性です。 *name*（オプション）は、コンテキスト内の Bean の名前です。

表 G-5 <om:test />

taglib での仕様	使用方法
<tag> <name>test</name> <attribute> <name>attr</name> <required>true</required> <rtexprvalue>true</rtexprvalue> </attribute> <attribute> <name>value</name> <required>true</required> <rtexprvalue>true</rtexprvalue> </attribute> <attribute> <name>prefix</name> <required>false</required> <rtexprvalue>true</rtexprvalue> </attribute> <tagclass>oracle.panama.studio.taglib.TestTag</tagclass> <bodycontent>JSP</bodycontent> <info>Test if the value of a bean attribute is the same as given using reflection.</info> </tag>	<om:test attr= *attrName* value= *attrValue* (prefix = *prefix*)>.</om:test> *<om:test>* は本体タグです。つまり、本体のコンテンツが評価され、テスト条件が true に評価されると、必要に応じて出力されます。 *attr* は、テストの対象となる Bean の属性です。 *value* は、テストの対象となる Bean の属性の値です。 *prefix*（オプション）は、コンテキスト内で Bean に対してメソッドを起動するときに使用する接頭辞です。

表 G-6 <om:equals />

taglib での仕様	使用方法
<tag> <name>equals</name> <attribute> <name>attr</name> <required>true</required> <rtexprvalue>true</rtexprvalue> </attribute> <attribute> <name>name</name> <required>true</required> <rtexprvalue>true</rtexprvalue> </attribute> <attribute> <name>prefix</name> <required>false</required> <rtexprvalue>true</rtexprvalue> </attribute> <tagclass>oracle.panama.studio.taglib.EqualsTag</tagclass> <bodycontent>JSP</bodycontent> <info>Test if the value of a bean attribute is the same as given using reflection.</info> </tag>	<om:equals attr= *attrName* name= *attrName* (prefix = *prefix*)>.</om:equals> <om:equals> は本体タグです。つまり、本体のコンテンツが評価され、テスト条件が true に評価されると、必要に応じて出力されます。 *attr* は、テストの対象となる Bean の属性です。 *name* は、値をテストするコンテキスト内の属性の名前です。 *prefix*（オプション）は、コンテキスト内で Bean に対してメソッドを起動するときに使用する接頭辞です。

表 G-7 <om:indexIs />

taglib での仕様	使用方法
<tag> <name>indexIs</name> <tagclass>oracle.panama.studio.taglib.IndexIsTag</tagclass> <bodycontent>JSP</bodycontent> <info>Test if the index inside iteration is the same as given by the user.</info> <attribute> <name>value</name> <required>true</required> <rtexprvalue>true</rtexprvalue> </attribute> </tag>	<om:indexIs value= *value* >.</om:indexIs> *<om:indexIs>* は本体タグです。つまり、本体のコンテンツが評価され、テスト条件が true に評価されると、必要に応じて出力されます。 *value* は、テストの対象となる索引の値です。

表 G-8 <om:indexEquals />

taglib での仕様	使用方法
<tag> <name>indexEquals</name> <tagclass>oracle.panama.studio.taglib.IndexEqualsTag</tagclass> <bodycontent>JSP</bodycontent> <info>Test if the index inside iteration is the same as the value found.</info> <attribute> <name>name</name> <required>true</required> <rtexprvalue>true</rtexprvalue> </attribute> </tag>	<om:indexEquals name= *attrName* >.</om:indexEquals> *<om:indexEquals>* は本体タグです。つまり、本体のコンテンツが評価され、テスト条件が true に評価されると、必要に応じて出力されます。 *name* は、値をテストするコンテキスト内の属性の名前です。

表 G-9 <om:index />

taglib での仕様	使用方法
<tag> <name>index</name> <tagclass>oracle.panama.studio.taglib.IndexTag</tagclass> <bodycontent>empty</bodycontent> <info>Gets the current index during iteration.</info> </tag>	<om:index /> <om:index> は単純タグです。つまり、このタグにはコンテンツを挿入できません。単に、<om:iterate> の一番内側で使用されている現行の Bean の索引を出力します。

表 G-10 <om:res />

taglib での仕様	使用方法
<tag> <name>res</name> <attribute> <name>name</name> <required>true</required> <rtexprvalue>true</rtexprvalue> </attribute> <tagclass>oracle.panama.studio.taglib.ResourceTag</tagclass> <bodycontent>empty</bodycontent> <info>A generic "get-resource" tag.</info> </tag>	<om:res name= *resName*>.</om:res> <om:res> は単純タグです。つまり、このタグの本体にはコンテンツを挿入できません。 *name* は、値を出力する（存在する場合）コンテキスト内の属性の名前です。 コンテキストは HTTP リクエスト・パラメータ、JSP ページ・コンテキストまたは現行の HTTP セッションとして定義されます。

表 G-11 <om:enc />

taglib での仕様	使用方法
<tag> <name>enc</name> <attribute> <name>name</name> <required>true</required> <rtexprvalue>true</rtexprvalue> </attribute> <tagclass>oracle.panama.studio.taglib.EncodeTag</tagclass> <bodycontent>empty</bodycontent> <info>A generic "encode-resource" tag.</info> </tag>	

表 G-12 <om:exist />

taglib での仕様	使用方法
<tag> <name>exist</name> <tagclass>oracle.panama.studio.taglib.ExistTag</tagclass> <bodycontent>JSP</bodycontent> <info>A logical exists tag.</info> <attribute> <name>name</name> <required>true</required> <rtexprvalue>true</rtexprvalue> </attribute> </tag>	<om:exist name= *attrName* >.</om:exist> <om:exist> は本体タグです。つまり、本体のコンテンツが評価され、テスト条件が true に評価されると、必要に応じて出力されます。 *name* は、テストの対象となるコンテキスト内の属性の名前です。 コンテキストは HTTP リクエスト・パラメータ、JSP ページ・コンテキストまたは現行の HTTP セッションとして定義されます。

表 G-13 <om:notexist />

taglib での仕様	使用方法
<tag> <name>notexist</name> <tagclass>oracle.panama.studio.taglib.NotExistTag</tagclass> <bodycontent>JSP</bodycontent> <info>A logical not-exists tag.</info> <attribute> <name>name</name> <required>true</required> <rtexprvalue>true</rtexprvalue> </attribute> </tag>	<om:notexist name= *attrName* >.</om:notexist> <om:notexist> は本体タグです。つまり、本体のコンテンツが評価され、テスト条件が true に評価されると、必要に応じて出力されます。 *name* は、テストの対象となるコンテキスト内の属性の名前です。 コンテキストは HTTP リクエスト・パラメータ、JSP ページ・コンテキストまたは現行の HTTP セッションとして定義されます。

表 G-14 <om:if />

taglib での仕様	使用方法
<tag> <name>if</name> <tagclass>oracle.panama.studio.taglib.IfTag</tagclass> <bodycontent>JSP</bodycontent> <info>A logical if tag.</info> <attribute> <name>name</name> <required>true</required> <rtexprvalue>true</rtexprvalue> </attribute> <attribute> <name>value</name> <required>true</required> <rtexprvalue>true</rtexprvalue> </attribute> <attribute> <name>op</name> <required>true</required> <rtexprvalue>false</rtexprvalue> </attribute> </tag>	<om:if name= *attrName* value= *attrValue*" op= *equal* > <om:then> .</om:then> <om:elseif name=*attrName* value=*attrValue* op=*equal*> <om:then> </om:then> <om:ese> <om:then> </om:then> </om:else> </om:elseif> </om:if> <om:if> は、<om:elseif>、<om:else> および <om:then> と組み合せて使用されます。 コンテキスト内で指定された属性の値と指定された値の間に演算子を適用し、その結果が true に評価される場合は、直下の子 <om:then> タグのコンテンツが評価されて出力されます。それ以外の場合は、<om:if> の直接の子である <om:elseif> または <om:else> が評価され、そのコンテンツが必要に応じて出力されます。 name: 値をテストするコンテキスト内の属性の名前。 コンテキストは HTTP リクエスト・パラメータ、JSP ページ・コンテキストまたは現行の HTTP セッションとして定義されます。 value: テストの対象となる属性の値。

表 G-15 <om:elseif>

taglib での仕様	使用方法
<tag> <name>elseif</name> <tagclass>oracle.panama.studio.taglib.ElseIfTag</tagclass> <bodycontent>JSP</bodycontent> <info>A logical elseif tag, allowed only inside if or elseif.</info> <attribute> <name>name</name> <required>true</required> <rtexprvalue>true</rtexprvalue> </attribute> <attribute> <name>value</name> <required>true</required> <rtexprvalue>true</rtexprvalue> </attribute> <attribute> <name>op</name> <required>true</required> <rtexprvalue>false</rtexprvalue> </attribute> </tag>	<om:if name= *attrName* value= *attrValue* op= *equal* > <om:then> .</om:then> <om:elseif name=*attrName* value=*attrValue* op=*equal*> <om:then> </om:then> <om:ese> <om:then> </om:then> </om:else> </om:elseif> </om:if> <om:elseif> は、<om:if>、<om:else> および <om:then> と組み合せて使用されます。 コンテキスト内で指定された属性の値と指定された値の間に演算子を適用し、その結果が true に評価される場合は、直下の子 <om:then> タグのコンテンツが評価されて出力されます。それ以外の場合は、<om:elseif> の直接の子である <om:else> が評価され、そのコンテンツが必要に応じて出力されます。 *name* は、値をテストするコンテキスト内の属性の名前です。 コンテキストは HTTP リクエスト・パラメータ、JSP ページ・コンテキストまたは現行の HTTP セッションとして定義されます。 *value* は、テストの対象となる属性の値です。

表 G-16 <om:else />

taglib での仕様	使用方法
<tag> <name>else</name> <tagclass>oracle.panama.studio.taglib.ElseTag</tagclass> <bodycontent>JSP</bodycontent> <info>A logical else tag.</info> </tag>	<om:if name= *attrName* value= *attrValue* op= *equal* > <om:then> .</om:then> <om:elseif name=*attrName* value=*attrValue* op=*equal*> <om:then> </om:then> <om:ese> <om:then> </om:then> </om:else> </om:elseif> </om:if> <om:else> は、<om:elseif>、<om:else> および <om:then> と組み合せて使用されます。 親である <om:if> または <om:elseif> が false に評価されると、子である <om:then> タグのコンテンツが評価され、必要に応じて出力されます。それ以外の場合は、何も処理されません。

表 G-17 <om:then />

taglib での仕様	使用方法
<tag> <name>then</name> <tagclass>oracle.panama.studio.taglib.ThenTag</tagclass> <bodycontent>JSP</bodycontent> <info>A logical else tag.</info> </tag>	<om:if name= *attrName* value= *attrValue* op= *equal* > <om:then> .</om:then> <om:elseif name=*attrName* value=*attrValue* op=*equal*> <om:then> </om:then> <om:ese> <om:then> </om:then> </om:else> </om:elseif> </om:if> <om:then> は、<om:if>、<om:elseif> および <om:else> と組み合せて使用されます。 親である <om:if>、<om:elseif> および <om:else> が true に評価されると、<om:then> タグのコンテンツが評価され、必要に応じて出力されます。

表 G-18 <om:iterate />

taglib での仕様	使用方法
<tag> <name>iterate</name> <tagclass>oracle.panama.studio.taglib.IterateTag</tagclass> <!-- teiclass>oracle.panama.studio.taglib.IterateTagTEI</teiclass --> <bodycontent>JSP</bodycontent> <info>An iteration tag.</info> <attribute> <name>name</name> <required>true</required> <rtexprvalue>true</rtexprvalue> </attribute> </tag>	<om:iterate name= *collectionName* > . . . <om:iterate /> <om:iterate> は、Bean オブジェクトのコレクションを反復するために使用されます。指定した名前を持つオブジェクトがコンテキスト内で見つかり、それが Java の Collection 型の場合は、そのコレクションをループできます。また、コレクション内の各オブジェクトはループしているため使用できます。<om:iterate> タグの本体は *n* 回出力されます。*n* はコレクション内のオブジェクトの数です。

表 G-19 <om:switch />

taglib での仕様	使用方法
<tag> <name>switch</name> <tagclass>oracle.panama.studio.taglib.SwitchTag</tagclass> <bodycontent>JSP</bodycontent> <info>A switch tag.</info> <attribute> <name>name</name> <required>true</required> <rtexprvalue>true</rtexprvalue> </attribute> </tag>	<om:switch name= *attrName*> <om:case value= *value1*> </om:case> <om:case value= *value2* > </om:case> <om:default> </om:default> </om:switch> <om:switch> は、<om:case> および <om:default> と組み合せて使用されます。<om:switch> では、テストする属性の名前を指定し、<om:case> 内ではテスト対象となる属性の値を指定します。両者が一致すると、一致条件を満たしている <om:case> の本体が評価され、必要に応じて出力されます。それ以外の場合は、<om:default > が指定されていれば、その本体が評価されて出力されます。

表 G-20 <om:case />

taglib での仕様	使用方法
<tag> <name>case</name> <tagclass>oracle.panama.studio.taglib.CaseTag</tagclass> <bodycontent>JSP</bodycontent> <info>A case tag.</info> <attribute> <name>value</name> <required>true</required> <rtexprvalue>true</rtexprvalue> </attribute> </tag>	<om:switch name= *attrName*> <om:case value= *value1*> </om:case> <om:case value= *value2* > </om:case> <om:default> </om:default> </om:switch> <om:switch> は、<om:case> および <om:default> と組み合せて使用されます。<om:switch> では、テストする属性の名前を指定し、<om:case> 内ではテスト対象となる属性の値を指定します。両者が一致すると、一致条件を満たしている <om:case> の本体が評価され、必要に応じて出力されます。それ以外の場合は、<om:default > が指定されていれば、その本体が評価されて出力されます。

表 G-21 <om:default />

taglib での仕様	使用方法
<tag> <name>default</name> <tagclass>oracle.panama.studio.taglib.DefaultTag</tagclass> <bodycontent>JSP</bodycontent> <info>A default tag. </info> </tag>	<om:switch name= *attrName*> <om:case value= *value1*> </om:case> <om:case value= *value2* > </om:case> <om:default> </om:default> </om:switch> <om:switch> は、<om:case> および <om:default> と組み合せて使用されます。<om:switch> では、テストする属性の名前を指定し、<om:case> 内ではテスト対象となる属性の値を指定します。両者が一致すると、一致条件を満たしている <om:case> の本体が評価され、必要に応じて出力されます。それ以外の場合は、<om:default > が指定されていれば、その本体が評価されて出力されます。

索引

L

M

O

P

Q

R

S

T

V

W

X

Y

あ

す

せ

た

ち

つ

て

と

は

ひ

ふ

ほ

ま

め

も

ゆ

り

る

れ

ろ