Oracle® Application Server Syndication Services

開発者および管理者ガイド

10*g*（9.0.4）

部品番号 : B12348-01

2004 年 6 月

ORACLE®

Oracle Application Server Syndication Services 開発者および管理者ガイド , 10*g*（9.0.4）

部品番号 : B12348-01

原本名 : Oracle Application Server Syndication Services Developer's and Administrator's Guide, 10*g* (9.0.4)

原本部品番号 : B10667-01

原本著者 : Rod Ward and Marco Carrer

原本協力者 : Cheng Han, Xiaohua Lu, Giuseppe Panciera, Alok Srivastava, Timothy Chien, Susan Shepard, Deborah Owens

目次

3 クライアント・アプリケーションの開発

4 コンテンツ・コネクタの開発

A エラー・メッセージ

B Syndication Services のセキュリティ

C シンジケーション・クライアントのサンプル

D RSS コンテンツ・コネクタ（CPAdaptor）

索引

例

図

表

はじめに

Oracle Application Server Syndication Services（Syndication Services）は、Oracle Application Server の機能の 1 つであり、シンジケーション関係（オファーとサブスクリプション）、配信ルール（契約）に基づくコンテンツ送信、結果分析（アクセス・ログ）を管理および自動化できます。

対象読者

このマニュアルは、シンジケーション関係の確立と配信ルールに基づくコンテンツ送信を自動化する開発者を対象としています。

また、このマニュアルは、Syndication Services を管理し、結果の分析を監視する管理者も対象としています。

このマニュアルの構成

このマニュアルの構成は、次のとおりです。

第 1 章「Syndication Services の概要」
Oracle Application Server Syndication Services を紹介し、用語と概念の説明および管理タスクとユーザー・タスクの概要を示します。

第 2 章「Syndication Services の管理」
Oracle Application Server Syndication Services の管理タスクの詳細を説明します。

第 3 章「クライアント・アプリケーションの開発」
クライアント・アプリケーションの開発およびコンテンツ・プロバイダの開発と登録について説明します。

第 4 章「コンテンツ・コネクタの開発」
コンテンツ・コネクタの開発とインストールについて説明します。

付録 A「エラー・メッセージ」
Oracle Application Server Syndication Services のエラー・メッセージについて説明します。

付録 B「Syndication Services のセキュリティ」
Oracle Application Server Syndication Services のセキュリティ情報について説明します。

付録 C「シンジケーション・クライアントのサンプル」
`SampleSyndicationClient.java` プログラムのリストを示します。

付録 D「RSS コンテンツ・コネクタ（CPAdaptor）」
`RSSCPAdaptor.java` プログラムのリストを示します。

関連ドキュメント

Syndication Services の使用および管理方法の詳細は、Oracle ドキュメント・セットの次のマニュアルを参照してください。

- 『Oracle9*i* Java Developer's Guide』
- 『Oracle Application Server Web Services 開発者ガイド』

Javadoc 形式の参照情報は、Oracle API ドキュメント（Javadoc とも呼ぶ）を参照してください。API ドキュメントは、『Syndication Services Client API Reference』（Javadoc）として用意されています。

リリース・ノート、インストール関連ドキュメント、ホワイト・ペーパーまたはその他の関連ドキュメントは、OTN-J（Oracle Technology Network Japan）から、無償でダウンロードできます。OTN-J を使用するには、オンラインでの登録が必要です。登録は、次の Web サイトから無償で行えます。

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

すでに OTN-J のユーザー名およびパスワードを取得している場合は、次の URL で OTN-J Web サイトのドキュメントのセクションに直接接続できます。

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

表記規則

このマニュアルでは、Oracle Application Server Syndication Services を Syndication Services とも呼びます。

このマニュアルでは、次の表記規則を使用しています。

規則	意味
. . .	例に含まれる垂直の省略記号は、その例に直接関係のない情報が省略されていることを示します。
. . .	文またはコマンド中の水平の省略記号は、その文またはコマンドのうち、例に直接関係のない部分が省略されていることを示します。
太字	太字は、本文中で定義されている用語を示します。また、GUI 要素を示すためにも使用されています。
< >	山カッコ内は、ユーザー指定の名前を示します。
[ ]	大カッコで囲まれた句は、オプション句を示します。オプションのうち 1 つを選択するか、または選択しなくてもかまいません。

1

Syndication Services の概要

Oracle Application Server Syndication Services（Syndication Services）では、シンジケーション関係（オファーとサブスクリプション）の確立、配信ルール（契約）に基づくコンテンツ送信、結果分析（アクセス・ログ）を管理および自動化できます。管理者は Syndication Services のプロパティも管理できます。

「用語と概念」では Syndication Services の用語や概念の一部を説明し、「使用フロー・チャート」では簡単な使用フロー・チャートをいくつか使用して、オファーの作成からサブスクリプションの管理までの Syndication Services の管理プロセスを説明します。

用語と概念

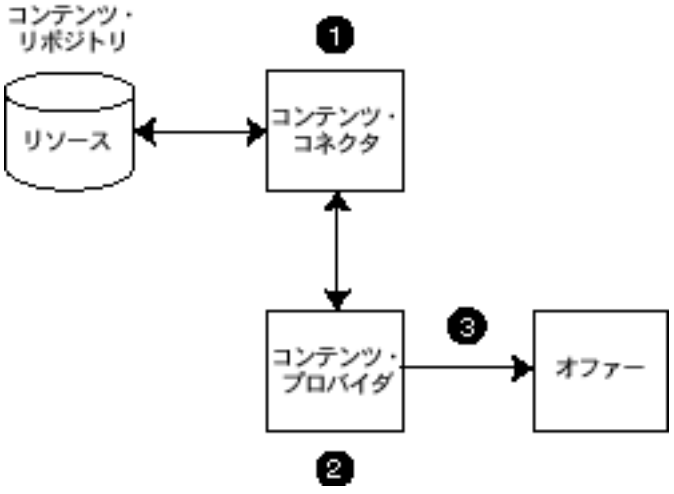
Syndication Services のコンテンツ・プロバイダに関する用語には、次のものが含まれます。

- コンテンツ -- パッケージとして配信される、ユーザーに関心のある情報。
- シンジケーション -- 外部コンテンツ・リポジトリからサブスクライバへのコンテンツの配信。
- シンジケータ -- コンテンツ配布者。この場合は Syndication Services。
- 管理者 -- タスクを実行することによって Syndication Services を管理する担当者。タスクには、コンテンツ・プロバイダ、オファーおよび契約の作成と管理や、ユーザーまたはグループに対するオファーへのアクセスの付与、サブスクリプションの管理などが含まれます。
- ユーザー -- オファーへのサブスクライブなどのタスクの実行を許可されている人。サブスクライバとも呼ばれます。
- コンテンツ・プロバイダ -- シンジケーション用にコンテンツ・コネクタを使用してシンジケータ（Syndication Services）にコンテンツを提供するエンティティ。コンテンツ・プロバイダは外部コンテンツ・リポジトリを表します。
- コンテンツ・コネクタ -- 外部コンテンツ・リポジトリと対話するために Syndication Services が使用するソフトウェア・コンポーネント。コンテンツ・プロバイダ・アダプタとも呼ばれます。Syndication Services は、ファイル・システムや Web Distributed Authoring and Versioning（WebDAV）フォルダなどのコネクタを提供します。コンテンツ・コネクタの開発の詳細は、第 4 章を参照してください。管理者は、コンテンツ・コネクタを使用して、コンテンツ・リポジトリへの接続に必要なパラメータを指定できます。
- リソース -- シンジケートされる外部コンテンツ・リポジトリの具体的な場所。ファイル・システム・コンテンツ・コネクタの場合、リソースはルート・ディレクトリの下にあるディレクトリです。WebDAV コンテンツ・コネクタの場合、リソースはベース・フォルダの下にあるフォルダです。各コンテンツ・プロバイダは、1 つのリソースとそのリソースへのアクセスに必要なパラメータ・セットを特定します。
- オファー -- ユーザーがサブスクライブ可能な最小単位のコンテンツ。オファーは、各コンテンツ・プロバイダ・リソースごとに作成できます。リソースは、たとえば、ファイル・システムのディレクトリや、WebDAV を介してアクセス可能なフォルダなどです。各オファーは、取引条件、有効期限ポリシーおよび配信ルールを含む契約に関連付けられます。

図 1-1 は、コンテンツ・プロバイダがコンテンツ・コネクタを使用して管理者にコンテンツ・リポジトリからのリソースを公開し、管理者は各リソースに対するオファーを作成するということを示しています。このオファーへのアクセスはユーザーに付与され、サブスクライブ可能になります。管理者は次のタスクを実行します。

1. コンテンツ・プロバイダを作成するときにコンテンツ・コネクタを選択します。
2. コンテンツ・プロバイダを作成および管理します。
3. 各コンテンツ・プロバイダ・リソースごとにオファーを作成します。

図 1-1 コンテンツ・プロバイダによるコンテンツの提供と管理者による各コンテンツ・プロバイダ・リソースのオファーの作成



Syndication Services のシンジケーション関係に関する用語には、次のものが含まれます。

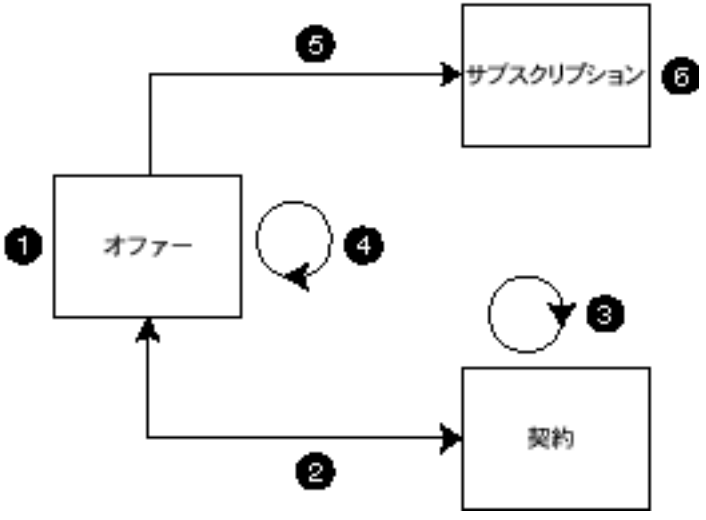
- 契約 -- 各オファーに関連付けられるもので、サブスクライバへコンテンツを配信するための取引条件、有効期限ポリシーおよび配信ルールが含まれます。
- 取引条件 -- オファーに関連付けられている契約に関する合意条件の詳細説明です。
- 有効期限ポリシー -- 契約に基づくサブスクリプションの期限がいつ切れるかを示します。
- 配信ルール -- コンテンツをサブスクライバに配信する方法、配信時期および配信頻度を詳細に示します。
- サブスクリプション -- オファーに関連付けられているコンテンツの配信に関するユーザー（サブスクライバ）とシンジケータ間の契約です。

図 1-2 は、ユーザーまたはユーザー・グループが、管理者から契約に基づいたオファーへのアクセスを付与されることを示します。管理者は各オファーとその契約を管理します。ユーザーがオファーにサブスクライブすると、関係はサブスクリプションになります。コンテンツは、オファーの有効期限ポリシーに基づいて設定された期間中、事前に規定されている配信ルールに定義されている方法で、パッケージとしてユーザーに配信されます。

管理者とユーザー（サブスクライバ）は次のタスクを実行します。

1. 管理者はオファーを作成および管理します。
2. 管理者は、ユーザーまたはユーザー・グループに対して、契約に基づいてオファーへのアクセスを付与します。
3. 管理者は、契約とその契約に関連付けられたオファーを作成および管理します。
4. ユーザー（サブスクライバ）は、オファーのカタログにアクセスします。
5. ユーザーはオファーにサブスクライブします。
6. 管理者はサブスクリプションを管理します。

図 1-2 サブスクリプション確立のための契約に基づくオファーへのユーザー（サブスクライバ）のサブスクライブ



「使用フロー・チャート」では、Syndication Services の管理者がシンジケーション・システムの管理で実行する一連のタスク（オファーの作成からサブスクリプション管理まで）を図示する簡単な使用フロー・チャートについて説明します。

使用フロー・チャート

使用フロー・チャートは、図 1-1 および図 1-2 に示されているように、シンジケーション・システムの管理者が関与する Syndication Services コンポーネント間での一連のステップと情報のフローを順番に示します。この項で、詳細を説明します。このフローには、コンテンツ・プロバイダの作成と管理、ユーザーまたはユーザー・グループに対するサブスクライブ対象オファーへのアクセスの付与、契約の作成と管理、さらにサブスクライバに属するサブスクリプションの管理が含まれます。「管理者の使用フロー・チャート」および「サブスクリプションの使用フロー・チャート」では、Syndication Services 管理者の観点からこの使用フロー・チャートをいくつか説明します。

管理者の使用フロー・チャート

Syndication Services 管理者は、シンジケーション・システムを初めて設定するときに次のタスクを実行します。

1. シンジケーション対象の新しいコンテンツ・リソースを識別するコンテンツ・プロバイダをそれぞれ登録します。

   a. コンテンツ・プロバイダのプロパティを入力します。このプロパティには、コンテンツ・プロバイダの名前、コンテンツを所有する連絡担当者の電子メール・アドレス、コンテンツ・プロバイダの簡単な説明などが含まれます。

   b. 該当するコンテンツ・リポジトリに必要なコネクタを識別する手段として、コンテンツ・コネクタのリストからコネクタを選択します。たとえば、FileConnector や WebDAVConnector などです。

   c. コンテンツ・リポジトリのアクセスに使用される設定を指定します。この設定は、選択したコンテンツ・コネクタのタイプにより異なります。

      たとえば、FileConnector コンテンツ・コネクタではコンテンツ・リポジトリとしてファイル・システムを使用し、その「ファイル・ルート・パス」設定でコンテンツのベースを定義します。ルート・ディレクトリの下にあるすべてのディレクトリは、管理者がオファーを作成できるコンテンツ・プロバイダ・リソースとみなされます。WebDAV コンテンツ・コネクタの場合はコンテンツ・リポジトリとしてフォルダを使用し、その「WebDAV ルート・フォルダ」設定でコンテンツのベース・フォルダを定義します。ベース・フォルダの下にあるすべてのフォルダは、管理者がオファーを作成できるコンテンツ・プロバイダ・リソースです。

   d. コンテンツ・リソースが指定場所にあることを確認します。指定場所とは、ファイル・システムの指定ルート・ディレクトリの下にある指定ディレクトリ内、または指定された WebDAV ベース・フォルダの下にある指定フォルダ内などです。

2. シンジケーション対象として識別された各コンテンツ・リソースにオファーを作成します。

   a. 管理者がオファーを作成するコンテンツ・プロバイダ・リソースを選択します。

   b. オファーの簡単な説明を入力し、コンテンツ使用方法プロパティ（知的所有権およびリクエストされるコンテンツ使用パターンなど）を指定します。

   c. 管理者がこのオファーへのサブスクライブ権限を付与するユーザーとグループ、およびこれらのユーザーに関連付ける契約を選択します。管理者は、特定のユーザーのオファーに対して特別に契約を作成することもできます。特別な契約をこのオファーに関連付けるには、管理者がこのステップで契約を選択できるように、契約と契約取引条件は作成済である必要があります。

3. ユーザー（サブスクライバ）とシンジケータ間のサブスクリプションを管理します。サブスクリプションの管理の詳細は、「サブスクリプションの使用フロー・チャート」を参照してください。

コンテンツ・プロバイダのファイル・システム・ルート・ディレクトリの下の追加ディレクトリ、またはコンテンツ・プロバイダの WebDAV ベース・フォルダの下の追加フォルダのように、追加のコンテンツ・リソースが使用可能になると、管理者は新規の各リソースに対してオファーを作成する必要があります。

ファイル・システムや WebDAV フォルダ以外に他のコンテンツ・リソースが特定されたため、これらの新規リソースをそれぞれ識別するために特殊なコンテンツ・コネクタを作成する必要があります。コンテンツ・コネクタの作成とインストールの詳細は、第 4 章を参照してください。特殊なコンテンツ・コネクタが作成されてインストールされた後、管理者はステップ 1 から 3 までを再び実行します。

管理タスクのフローを示す概要説明により、管理者はシンジケーション・システムの設定を開始できます。実行する最初のタスクは、2-2 ページの「Syndication Services のプロパティの設定」で説明しています。これらのタスクには、プロパティ・グループ （一般プロパティ、スケジューラ構成プロパティ、HTTP/S および SMTP トランスポート・プロパティ）に対する一連の初期パラメータの指定が含まれます。さらに、管理者はアクセス・ログを調べて、アクセス・ログ・エントリをパージする必要があります。2-20 ページの「アクセス・ログの確認およびパージ」に説明しているように、アクセス・ログは Syndication Services へのユーザー・アクセスを記録します。

サブスクリプションの使用フロー・チャート

ユーザー（サブスクライバ）とシンジケータとの間に特定のオファーに関するサブスクリプションが作成されると、Syndication Services 管理者はこのサブスクリプションを管理する必要があります。これには次のタスクが含まれます。

1. サブスクリプションの検討。サブスクリプションの状態が「アクティブ」のときに、そのサブスクリプションを保留または終了する必要がある場合、管理者は状態を「保留」または「終了」に変更できます。保留サブスクリプションは、再び「アクティブ」状態にすることができます。ただし、終了サブスクリプションはパージのみが可能です。
2. 期限切れサブスクリプションと終了サブスクリプションのパージ。
3. 確認通知、有効期限ポリシーおよび配信ルールなど、その他のサブスクリプション情報の表示。管理者は、サブスクリプションに関連付けられている承認済オファー情報を確認できます。この情報には、コンテンツ使用方法プロパティ（知的所有権、リクエストされるコンテンツ使用パターン、コンテンツ・プロバイダおよびリソースの情報）が含まれます。

2

Syndication Services の管理

Syndication Services の管理には、次のタスクが含まれます。

- インストール後の構成タスクの完了
- Syndication Services のプロパティの設定
- ユーザー管理の理解
- コンテンツ・プロバイダの登録および管理
- オファーの作成および管理
- サブスクリプションの管理
- アクセス・ログの確認およびパージ
- パフォーマンスの監視およびチューニング
- 高度な管理タスクの実行

次の各項では、このような管理タスクをそれぞれ説明します。

インストール後の構成タスクの完了

Syndication Services の使用を開始する前に、ドメイン・プロパティを設定する必要があります。これは、Oracle Application Server Infrastructure データベースがインストールされているシステムのドメイン名です。このプロパティの設定の詳細は、「Syndication Services のプロパティの設定」を参照してください。

Syndication Services のプロパティの設定

コンテンツ・プロバイダおよびオファーの登録、サブスクリプションの管理、コンテンツのシンジケーションを開始する前に Syndication Services のプロパティを設定する必要があります。**Oracle Enterprise Manager Syndication Services の管理**ページで「**システム・プロパティ**」をクリックします。「**システム・プロパティ**」ページが表示されます。

このページで設定する必要のあるプロパティは、アクセス・ロギング、ドメイン、スケジューラの状態、HTTP/S トランスポートおよび SMTP トランスポートです。表 2-1 では、Syndication Services のすべてのプロパティを記載し、設定する必要のあるプロパティに関するガイドラインを示します。

注意： Syndication Services のプロパティを設定した場合は、設定を有効にするために、Syndication Services アプリケーションがデプロイされている OC4J_Portal インスタンスを再起動する必要があります。

表 2-1 Syndication Services のプロパティ

プロパティ	説明
一般プロパティ	
インスタンス ID	Syndication Services インスタンス用に生成された ID を指定します。
インスタンス名	Syndication Services インスタンスの名前を指定します。
ユーザー・エージェント名	Syndication Services インスタンスに関連付けられており、HTTP ユーザー・エージェント名に類似したユーザー・エージェントの名前を指定します。
アクセス・ロギング（必須）	ユーザー・アクセス情報の収集を有効または無効にします。ユーザー・アクセス情報の収集を有効にする場合は、このプロパティを「**オン**」に設定します。アクセス・ロギングは、Syndication Services への各ユーザー・アクセスをアクセス・ログのエントリとして記録します。ログ・エントリは必要に応じて確認およびパージできます。

表 2-1 Syndication Services のプロパティ（続き）

プロパティ	説明
ドメイン（必須）	Oracle Application Server Infrastructure データベースがインストールされているシステムのドメイン名を、たとえば、`us.oracle.com` のような形式で指定します。ドメイン名は、Syndication Services の管理ページが Oracle Delegated Administration Services ページと対話して、ユーザーおよびユーザー・グループを選択したり、オファーに対するサブスクライブ権限を付与する際に必要になります。
スケジューラ構成プロパティ	
スケジューラの状態（必須）	自動プッシュ配信ができるようにスケジューラが有効になっているかどうかを示します。自動プッシュ配信を可能にするには、スケジューラの状態を「**オン**」に設定します。詳細は、2-26 ページの「スケジューラの構成」を参照してください。
タイマ・プール・サイズ	スケジューラの使用と同時にアクティブになるタイマーの数を指定します。デフォルト値は 16 です。詳細は、2-26 ページの「スケジューラの構成」を参照してください。
監視頻度（ミリ秒）	コンテンツのプッシュを必要とする新規のサブスクリプションをスケジューラがチェックする頻度を指定します。デフォルト値は 60000 ミリ秒（1 分）です。詳細は、2-26 ページの「スケジューラの構成」を参照してください。
HTTP/S トランスポート・プロパティ（必須）	
プロキシ・ホスト	ファイアウォールの外側にあるサーバーに、コンテンツのプッシュを実行するプロキシ・サーバーの名前を指定します。
プロキシ・ポート	HTTP プロキシ・ホストのポート番号を指定します。
接続のタイムアウト	コンテンツのソースまたはリポジトリへの接続を確立するために Syndication Services が待機する時間（秒数）を指定します。この時間が経過するとタイムアウト・メッセージが返されます。
Wallet の場所	コンテンツのプッシュ用の HTTPS 接続の確立に必要な Oracle Wallet Manager の場所を指定します。詳細は、『Oracle Advanced Security 管理者ガイド』の「Oracle Wallet Manager の使用方法」参照してください。
SMTP トランスポート・プロパティ（必須）	
SMTP サーバー	SMTP サーバーの名前を指定します。
差出人	電子メールを介してコンテンツのプッシュを実行するときに使用する送信者の電子メール・アドレスを指定します。
件名	電子メールを介してコンテンツのプッシュ配信を実行するときに使用する電子メール・ヘッダーの件名部分を指定します。

ユーザー管理の理解

Oracle Application Server Syndication Services 10*g*（9.0.4）では、デフォルト・ユーザー・リポジトリとして Oracle Application Server Infrastructure の Oracle Internet Directory（OID）を使用します。これは、Oracle Application Server Containers for J2EE（OC4J）の Oracle Application Server Java Authentication and Authorization Service（JAAS）の Lightweight Directory Access Protocol（LDAP）ベース・プロバイダを使用して行われます。

Syndication Services 固有の OID グループは、OID デフォルト・サブスクライバのグループ・サブツリーの `cn=syndication_groups` サブツリーの下にあります。Syndication Services 固有のユーザーは、OID デフォルト・サブスクライバのユーザー・サブツリーの下にあります。

表 2-2 に Syndication Services グループのタイプを要約します。

表 2-2 デフォルトの Syndication Services グループ

グループ名	説明
syndsubscribers	このグループ内のユーザーは、サーバーに対する ping の実行、カタログの取出し、サブスクリプションの作成または変更、イベント（特定のサブスクリプションに関連付けられているアクティビティ）の取得、サブスクリプション・ステータスの取出し、サブスクリプションに関係するコンテンツのプル、およびサブスクリプションの取消しができます。
syndcps	このグループ内のユーザーは、コンテンツ・プロバイダ側でのコンテンツの更新をサーバーに通知できます。イベントのプッシュ・サポートの詳細は、4-12 ページの「コンテンツ・プロバイダのイベント・プッシュ・サポート」を参照してください。
syndschedulers	Syndication Services の実装により定義されている内部グループです。管理者は、このグループに対する編集、ユーザーの追加または削除を行わないでください。このグループは、サブスクリプション配信ルールに基づくプッシュ操作を実行します。

注意： これらのデフォルトの Syndication Services グループは、どれも削除しないでください。

デフォルトでは、次に示すユーザーがインストール時に作成されます。管理者は、表 2-3 に示すように、対応するグループに対してユーザーを追加および削除できます。

表 2-3 デフォルトの Syndication Services ユーザー

グループ名	ユーザー名	コメント
syndsubscribers	syndication	サンプルのサブスクライバ。管理者は、このグループに新規ユーザーを追加できます。
	portal_syndication	portal_syndication ユーザーは、主にポータル統合のために使用されます。このユーザーは削除しないでください。
	uddi_syndication	uddi_syndication ユーザーは、主に UDDI 統合のために使用されます。このユーザーは削除しないでください。
syndcps	sample_cps	サンプルのコンテンツ・プロバイダ・ユーザー。このユーザーは削除しないでください。
syndschedulers	syndscheduler	デフォルトのスケジューラ・ユーザーはプッシュ操作を実行します。このユーザーは削除しないでください。

作成、削除、保留などの一般的なユーザー管理は、OID とその Delegated Administration Service（DAS）により処理されます。詳細は、『Oracle Internet Directory 管理者ガイド』を参照してください。

注意： DAS では、表示名が実際の名前と異なることがあります。

Syndication Services へのアクセスを既存の OID ユーザーに付与するには、DAS を使用して各ユーザーを syndsubscribers グループに割り当てます。

作成、削除、保留、ロール管理などの操作を含むユーザー管理は、OC4J Java Authentication and Authorization（JAAS）サービスにより処理されます。詳細は、『Oracle Application Server Containers for J2EE サービス・ガイド』を参照してください。

コンテンツ・プロバイダの登録および管理

ユーザーがサブスクライブ可能なオファーを作成する前に、コンテンツがどこにあるかを識別するコンテンツ・プロバイダを登録する必要があります。コンテンツ・プロバイダはコンテンツ・コネクタに依存します。コンテンツ・コネクタは、外部コンテンツ・リポジトリおよびそのリソースへの接続に使用されるソフトウェア・インタフェースです。コンテンツ・プロバイダ設定は、リポジトリの具体的な場所とリポジトリのその他の基本的な特性を識別します。外部コンテンツ・リポジトリ内で識別されているリソースごとにオファーを作成できます。使用可能と示されているコンテンツ・プロバイダは、そのコンテンツ・プロバイダのクラスがクラス・パス内に含まれているということを意味します。

たとえば、ファイル・コネクタではコンテンツ・リポジトリとしてファイル・システムを使用します。「**ファイル・ルート・パス**」設定では、コンテンツのルートの場所を定義します。ルート・ディレクトリの下にあるすべてのディレクトリは、オファーを作成できるコンテンツ・プロバイダ・リソースです。

別の例として、WebDAV コネクタではコンテンツ・リポジトリとしてフォルダを使用します。「**WebDAV ルート・フォルダ**」設定では、コンテンツのベース・フォルダを定義します。ベース・フォルダの下にあるすべてのフォルダは、オファーを作成できるコンテンツ・プロバイダ・リソースです。

Oracle Enterprise Manager Syndication Services の管理ページで「**コンテンツ・プロバイダ**」をクリックします。「**コンテンツ・プロバイダ管理**」ページが表示されます。

「**コンテンツ・プロバイダ管理**」ページでは、次のことができます。

- コンテンツ・コネクタ情報の確認
- コンテンツ・プロバイダの登録
- コンテンツ・プロバイダ情報の編集
- コンテンツ・プロバイダの削除

コンテンツ・コネクタ情報の確認

使用可能なコンテンツ・コネクタのリストを確認するには、「**コンテンツ・コネクタ**」タブをクリックします。「**コンテンツ・コネクタ**」ページが表示され、使用可能なコンテンツ・コネクタのリストを確認できます。使用可能と示されているコンテンツ・コネクタは、そのコンテンツ・コネクタ用のクラスがクラス・パス内に含まれているということを意味します。

コンテンツ・コネクタのプロパティを表示するには、「**名前**」列にあるコンテンツ・コネクタの名前をクリックするか「**選択**」列でそのコネクタのラジオ・ボタンをクリックしてから「**表示**」をクリックします。「**コンテンツ・コネクタ**」ページが表示され、コンテンツ・コネクタの名前、コンテンツ・コネクタの説明およびコネクタの Java クラスの名前などを確認できます。この情報の確認が終了したら、「**OK**」をクリックして「**コンテンツ・コネクタ**」ページに戻ります。

組込みコネクタ

Syndication Services には、次のコンテンツ・コネクタがあらかじめ構成されています。

- FileConnector -- コンテンツ・リポジトリとしてファイル・システムを使用するコネクタで、コンテンツのルートの場所としてルート・ディレクトリを定義します。ルート・ディレクトリの下にあるすべてのディレクトリは、管理者がオファーを作成できるコンテンツ・プロバイダ・リソースとみなされます。
- WebDAVConnector -- Distributed Authoring and Versioning（DAV）プロトコルを使用してコンテンツ・リソースへのバインディングを作成するコネクタです。ベース・フォルダの下にあるすべてのフォルダは、管理者がオファーを作成できるコンテンツ・プロバイダ・リソースとみなされます。
- UDDI コネクタ -- 9 つの事前構成コネクタで、管理者はこの中からコネクタを選択してコンテンツ・プロバイダ・リソースを作成し、その後一連のオファーを作成します。Oracle Application Server の UDDI（Universal Description, Discovery and Integration）レジストリ内の UDDI パブリッシャは、これらのオファーにサブスクライブし、レジストリ内の変更を監視または取得できます。詳細は、『Oracle Application Server Web Services 開発者ガイド』の「UDDI サブスクリプション・サービス」を参照してください。

Really Simple Syndication（RSS）のサンプル・コンテンツ・コネクタ（このサンプルの完全なソース・ファイルを含む）は、Oracle Technology Network Japan（OTN-J）の Web サイト（`http://otn.oracle.co.jp/sample_code/index.html`）からダウンロードできます。このコネクタについては第 4 章で説明します。このコネクタは、管理者がコンテンツ・プロバイダ・リソースに対して作成できるシンジケーション・フィード（コンテンツ更新）を取得します。ここからオファーが作成され、これに対してユーザーがサブスクライブできます。付録 D では、このコネクタを実装した `RSSCPAdaptor.java` プログラムのリストを示します。

4-2 ページの「コンテンツ・コネクタ（CPAdaptor）の開発」では新規コンテンツ・コネクタの開発方法を、4-14 ページの「新規コンテンツ・コネクタのインストール」では新規コンテンツ・コネクタのインストール方法を説明します。

コンテンツ・プロバイダの登録

コンテンツ・プロバイダを登録するには、「**コンテンツ・プロバイダの登録**」をクリックします。「コンテンツ・プロバイダ登録」ウィザードに、最初の「**コンテンツ・プロバイダの登録 : プロパティ**」ページが表示されます。

1. 「**コンテンツ・プロバイダの登録 : プロパティ**」ページに、このコンテンツ・プロバイダの必須プロパティを入力します。このプロパティには、コンテンツ・プロバイダの名前、コンテンツ・プロバイダの所有者の電子メール・アドレス、コンテンツ・プロバイダの簡単な説明などが含まれます。次に、「**次へ**」をクリックして「**コンテンツ・プロバイダの登録 : コネクタ**」ページに進みます。
2. 「**コンテンツ・プロバイダの登録 : コネクタ**」ページでコンテンツ・コネクタのラジオ・ボタンを選択し、このリポジトリへのアクセスに使用するコンテンツ・コネクタを選択します。次に、「**次へ**」をクリックして「**コンテンツ・プロバイダの登録 : 設定**」ページに進みます。
3. 「**コンテンツ・プロバイダの登録 : 設定**」ページで、登録に必要な値を入力してコンテンツ・コネクタ・プロパティを構成した後、「**完了**」をクリックしてコンテンツ・プロバイダを作成し、「**コンテンツ・プロバイダ管理**」ページに戻ります。

コンテンツ・プロバイダ情報の編集

コンテンツ・プロバイダ情報を編集するには、編集するコンテンツ・プロバイダ名を「**名前**」列でクリックするか、そのコンテンツ・プロバイダのラジオ・ボタンを「**選択**」列で選択し、「**編集**」をクリックします。「**コンテンツ・プロバイダの編集**」ページが表示されます。このページでは、次のことができます。

- コンテンツ・プロバイダ名の表示
- コンテンツ・コネクタのタイプの表示
- コンテンツ・プロバイダの所有者の電子メール・アドレスの編集
- コンテンツ・プロバイダの説明の編集
- コンテンツ・コネクタの構成設定の編集

コンテンツ・プロバイダの削除

コンテンツ・プロバイダを削除するには、そのプロバイダのラジオ・ボタンを「**選択**」列でクリックしてコンテンツ・プロバイダを選択し、次に「**削除**」をクリックします。コンテンツ・プロバイダを登録解除すると関連するオファーがすべて削除されることを示す警告メッセージが表示されます。「**削除**」をクリックして確認します。

注意： コンテンツ・プロバイダのリソースに基づくオファーに依存するサブスクリプションがアクティブである（期限切れまたは終了ではない）場合は、そのコンテンツ・プロバイダを削除できません。コンテンツ・プロバイダを削除するには、これらのサブスクリプションをすべて終了する必要があります。

オファーの作成および管理

コンテンツ・プロバイダをいくつか登録すると、ユーザーがサブスクライブ可能なオファー作成プロセスを開始できます。

Oracle Enterprise Manager Syndication Services の管理ページで「**オファー**」をクリックします。「**オファー管理**」ページが表示されます。

「**オファー管理**」ページでは、次のことができます。

- コンテンツ・プロバイダのリストの表示
- コンテンツ・プロバイダのオファーの表示
- コンテンツ・プロバイダのオファーの作成
- 契約および取引条件の管理

コンテンツ・プロバイダのリストの表示

コンテンツ・プロバイダのリストがプロバイダの説明とともに表示されます。

コンテンツ・プロバイダのオファーの表示

コンテンツ・プロバイダのオファーを表示するには、そのプロバイダのラジオ・ボタンを「**選択**」列でクリックしてリストからプロバイダを選択し、次に「**オファーの表示**」をクリックします。**コンテンツ・プロバイダのオファー**・リスト・ページが表示され、このコンテンツ・プロバイダのオファーがリストされます。

コンテンツ・プロバイダのオファー・リスト・ページでは、次のことができます。

- オファーのプロパティの編集
- オファーにアクセス可能なユーザーおよびグループのアクセス・リストの編集
- オファーの削除
- コンテンツ・プロバイダのオファーの作成

オファーのプロパティの編集

オファーのプロパティを編集するには、そのオファーのラジオ・ボタンを「**選択**」列でクリックしてオファーを選択し、次に「**編集**」をクリックします。「**オファーの編集**」ページが表示され、次のプロパティを編集できます。

- **一般プロパティ** -- オファー名、オファーの説明およびオファーの状態 （アクティブまたは保留）。

  注意： 保留状態のオファーは、サブスクリプションには使用できません。

- **コンテンツ使用方法プロパティ** -- Information Content Exchange（ICE）1.1 標準プロパティ : オファーが属する製品名、知的所有権の所有者名、コンテンツの知的所有権のステータス（表 2-4 を参照）およびコンテンツ使用パターン（表 2-5 を参照）。これらのオファー・プロパティの詳細は、表 2-11 を参照してください。Syndication Services ではこれらのプロパティを規定しません。サブスクライバのクライアント・アプリケーションがこれらを処理する必要があります。
- **コンテンツ・プロバイダ** -- 名前およびリソース名。

表 2-4 知的所有権のカテゴリ

カテゴリ	説明
パブリック・ドメイン	コンテンツにライセンス上の制限がないことを示します。
ソース明示のみ必要	コンテンツには、コンテンツ・ソースを明示するという要件以外にライセンス上の制限がないことを示します。
ライセンス参照	コンテンツには、既存のライセンス契約で合意されているライセンス上の制限があることを示します。
厳しい制限	コンテンツには、特に注意の必要なライセンス上の制限があることを示します。
機密	コンテンツは機密であり、特別に保護する必要があることを示します。
その他	コンテンツには、上記のリストにない知的所有権のステータスがあることを示します。

表 2-5 コンテンツ使用パターンのカテゴリ

カテゴリ	説明
基本使用	配信されるコンテンツをすべて使用するか、まったく使用しないかのいずれかであることを示します。デフォルトは false（チェックなし）で、コンテンツの基本使用が必要でないことを示します。
編集可能	ユーザー（サブスクライバ）が情報を変更できるか、または配信されたままの情報を使用する必要があるかを示します。デフォルト値は false（チェックなし）で、コンテンツが編集できないため、情報を配信されたまま使用する必要があることを示します。
クレジットの表示	ユーザー（サブスクライバ）がコンテンツ・ソースの属性を明示する必要があるかどうかを示します。デフォルト値は false（チェックなし）で、ユーザーはデータのソースを明示する必要がないことを示します。
使用方法必須	シンジケートされたコンテンツの意図するビューアまたは最終ビューアに関する使用方法のデータがユーザーから返されることを、シンジケータが予想するかどうかを示します。

オファーにアクセス可能なユーザーおよびグループのアクセス・リストの編集

特定のオファーへのサブスクライブ権限を付与するユーザーおよびグループを選択するには、ユーザー・アイコンをクリックしてユーザーを選択し、該当するユーザー名をクリックします。グループを選択するには、グループ・アイコンをクリックしてから該当するグループ名をクリックします。

作成、削除、保留などの一般的なユーザー管理は、OID と DAS により処理されます。インストール時に作成されるデフォルトのグループおよびユーザーのリストは、2-4 ページの「ユーザー管理の理解」を参照してください。詳細は、『Oracle Internet Directory 管理者ガイド』を参照してください。

注意： 選択可能なユーザーおよびグループのリストは、OID に定義されているすべてのユーザーおよびグループです。Syndication Services に接続してシンジケーション操作を実行できるのは、`syndsubscribers` グループに属するユーザーのみであることに注意してください。詳細は、2-4 ページの「ユーザー管理の理解」を参照してください。

Netscape 7.0 を使用している場合は、ユーザーを選択しても実行されないことがあります。つまりユーザーが選択されないということです。この問題を回避するには、Internet Explorer 6.0 以降を使用してください。

次に、選択したユーザーおよびグループに関連付ける契約を変更できます。これには、ドロップダウン・メニューから該当する契約を選択し、「**追加**」をクリックして契約を変更します。ユーザーまたはグループのオファーへのアクセス権は、特定のユーザーまたはグループ名の削除アイコンをクリックして取り消すことができます。ユーザーまたはグループに対して、このオファーへのアクセス権を付与または取り消した後、「**OK**」をクリックしてすべての変更を保存すると、「**オファー管理 : オファー**」ページに戻ります。

オファーの削除

オファーを削除するには、オファーのラジオ・ボタンを「**選択**」列でクリックしてから「**削除**」をクリックします。警告メッセージが表示されます。「**削除**」をクリックしてオファーを削除します。

コンテンツ・プロバイダのオファーの作成

詳細は、「コンテンツ・プロバイダのオファーの作成」を参照してください。

コンテンツ・プロバイダのオファーの作成

オファーを作成するには、「**オファーの作成**」をクリックします。「オファーの作成」ウィザードに、「**オファーの作成 : リソース**」の最初のページが表示されます。

1. 「**オファーの作成 : リソース**」ページで、懐中電灯をクリックしてリソースを選択し、オファー作成対象のコンテンツ・プロバイダ・リソースを選択します。コンテンツ・プロバイダからのリソースは、コンテンツ・プロバイダを登録した後に使用可能になります(2-8 ページの「コンテンツ・プロバイダの登録」を参照してください)。「**次へ**」をクリックし、「オファーの作成」ウィザードの次のページに進みます。
2. 「**オファーの作成 : プロパティ**」ページで、次の情報を指定します。
   - **一般プロパティ** -- オファーの名前およびオファーの説明。
   - **コンテンツ使用方法プロパティ** -- ICE 1.1 標準プロパティ : 知的所有権の所有者名。知的所有権のコンテンツ・ステータス（表 2-4 を参照）: パブリック・ドメイン、ソース明示のみ必要、ライセンス参照、厳しい制限、機密、その他。コンテンツ使用パターン（表 2-5 を参照）: 基本使用、編集可能、クレジットの表示および使用方法必須。これらのコンテンツ使用方法プロパティは、オファー・コンテンツの使用に関連付けられている可能性のある制限についてユーザーに通知するオプションのプロパティです。

   「**次へ**」をクリックし、「オファーの作成」ウィザードの次のページに進みます。
3. 「**オファーの作成 : 契約**」ページで、このオファーへのアクセス権を付与するユーザーおよびグループを選択し、選択したユーザーおよびグループに関連付ける契約を選択します。ユーザーを選択するには、ユーザー・アイコンをクリックし、該当するユーザー名をクリックします。グループを選択するには、グループ・アイコンをクリックし、該当するグループ名をクリックします。ユーザーまたはグループに対するオファーへのアクセスを変更するには、特定のユーザー名またはグループ名の削除アイコンをクリックし

ます。オファーへのアクセス権をユーザーおよびグループに付与した後、「**完了**」をクリックしてオファーを作成します。

注意： 選択可能なユーザーおよびグループのリストは、OID に定義されているすべてのユーザーおよびグループです。Syndication Services に接続してシンジケーション操作を実行できるのは、syndsubscribers グループに属するユーザーのみです。詳細は、2-4 ページの「ユーザー管理の理解」を参照してください。

契約および取引条件の管理

「**オファー管理 : 契約**」ページでは、次のことができます。

- 契約の編集
- 契約の取引条件の編集
- 契約の削除
- 契約の作成
- 契約の取引条件の作成

契約の編集

契約を編集するには、その契約のラジオ・ボタンを「**選択**」列で選択し、次に「**編集**」をクリックします。「**契約の編集 : 一般**」ページが表示されます。このページでは、次の項目を編集できます。

- **一般プロパティ** -- 契約名、契約の説明および確認の要求。確認の要求とは、次のコンテンツ・パッケージをユーザーに配信する前にコンテンツ・パッケージの受信確認が必要かどうかを意味します。コンテンツ・パッケージは、配信されるコンテンツの 1 単位を表します。
- **有効期限ポリシー** -- この契約に基づくサブスクリプションの有効期限情報（表 2-6 を参照）。これには、タイム・ゾーンを含む開始日および停止日、サブスクリプションの最大配信数、有効期限の優先順位（表 2-7 を参照。時間ベース、数量ベース、最初または最後のいずれか）などが含まれます。

表 2-6 有効期限ポリシーのカテゴリ

カテゴリ	説明
タイム・ゾーン	同一標準時間が使用される地域を示します。通常は、3 つのアルファベット文字で示されます。たとえば、EST は東部（米国）標準時を表します。
開始日	コンテンツの配信が開始される日付を示します。これは、デフォルトでは、1 日の始まり（00:00:00 AM）を意味します。
有効期限の優先順位	有効期限の基準を指定します。使用可能な値の説明の詳細は、表 2-7 を参照してください。
停止日	コンテンツの配信が期限切れになる日付を示します。これは、デフォルトでは、1 日の終わり（23:59:59 PM）を意味します。無制限の有効期限ポリシーを指定するには、時間ベースの有効期限優先順位を指定し、停止日は指定しないでください。
最大量	許可されている配信回数の最大値を 0 以上の整数値として示します。

表 2-7 有効期限の優先順位の値

値	説明
時間ベース	停止日に達するとサブスクリプションが期限切れになることを示します。
数量ベース	許可されている配信回数に達するとサブスクリプションが期限切れになることを示します。
最初	停止日に達するか許可されている配信回数に達するとサブスクリプションが期限切れになることを示します。
最後	停止日に達し、かつ許可されている配信回数に達した場合にのみサブスクリプションが期限切れになることを示します。

- **プル配信ルール** -- 「**プル配信の有効化**」が選択されている場合、一連の制御によってコンテンツがサブスクライバにオンデマンドで配信されます。プル配信ルールでは、コンテンツ・プルが可能な日毎の期間（指定開始時間から指定された時間継続する期間）を記述します（表 2-8 を参照）。タイム・ゾーン、プル・コンテンツ配信の開始時間（時間および分）、コンテンツ配信の合計継続時間（時間および分）、およびプル配信が可能な曜日または日付を指定します。

表 2-8 プルおよびプッシュ配信のカテゴリの説明

カテゴリ	説明
タイム・ゾーン	同一標準時間を使用する地域を示します。通常は、3 つのアルファベット文字で示されます。たとえば、EST は東部（米国）標準時を表します。
開始時間	プル・コンテンツ配信が開始される時間（時間および分）および AM または PM を指定します。開始時間は、デフォルトでは 1 日の始まり（00:00 AM）です。
継続時間	配信するコンテンツに対して許可されている合計配信時間（時間および分）を指定します。この値は 24 時間を超えないように指定してください。
継続時間当たりの更新回数	指定された配信時間枠内でユーザーのコンテンツが更新される頻度を指定します。このオプションは、プッシュ配信専用です。
曜日指定	コンテンツを配信する曜日（週）を指定します。曜日を指定する場合は、「**選択**」を選択し、コンテンツを配信する曜日にチェックを付けます。「**任意**」を選択すると、配信に曜日の制限がないことを意味します。
日付指定	コンテンツが配信される日付を指定します。日付を指定する場合は、「**選択**」を選択し、コンテンツを配信する日付にチェックを付けます。「**任意**」を選択すると、配信に日付の制限がないことを意味します。「**LAST**」は、月の最終日のみを意味します。

注意： コンテンツ配信を曜日または日付で制限する場合は、適切なオプションを選択してください。両方のオプションを一緒に使用してコンテンツ配信を制限する場合は、両方の条件を満たす必要があります。たとえば、曜日として「M」または「月曜日」を選択し、日付として「3」を選択すると、コンテンツ配信は 3 日の月曜日のみに限定されます。

- **プッシュ配信ルール** -- 「**プッシュ配信の有効化**」が選択されている場合、一連の制御によってコンテンツが時間ベースのスケジュールでサブスクライバに自動的に配信されます。プッシュ配信ルールでは、コンテンツ・プッシュが可能な日毎の期間（指定開始時間から指定された時間継続する期間）を記述します（表 2-8 を参照）。タイム・ゾーン、プッシュ・コンテンツ配信の開始時間（時間および分）、コンテンツ配信のための合計継続時間（時間および分）、配信期間ごとに許可されるコンテンツ更新回数、およびプッシュ配信が可能な曜日または日付を指定します。

注意： コンテンツ配信を曜日または日付で制限する場合は、適切なオプションを選択してください。両方のオプションを一緒に使用してコンテンツ配信を制限する場合は、両方の条件を満たす必要があります。たとえば、曜日として「M」または「月曜日」を選択し、日付として「3」を選択すると、コンテンツ配信は 3 日の月曜日のみに限定されます。

契約の取引条件の編集

契約の取引条件を編集するには、「**取引条件**」タブをクリックします。「**契約の編集 : 取引条件**」ページが表示されます。このページでは、ユーザーがオファーにサブスクライブする際に同意する必要のある合意事項を定義する条件が定義されています。取引条件の名前および説明を編集し、契約に関連する合意内容を編集できます。編集が終了したら、「**OK**」をクリックしてすべての変更を保存すると、「**オファー管理 : 契約**」ページに戻ります。

契約の削除

契約を削除するには、その契約のラジオ・ボタンを「**選択**」列でクリックし、次に「**削除**」をクリックします。「契約を削除すると、関連付けられているオファーのアクセス・ポリシーも変更されます。続行してよろしいですか ?」という警告メッセージが表示されます。「**削除**」をクリックして契約の削除を確定します。

契約の作成

契約を作成するには、「**作成**」をクリックします。「**契約の作成 : 一般**」ページが表示されます。このページでは、詳細を記入して契約を作成できます。詳細には次のものが含まれます。

- **一般プロパティ** -- 契約名、契約の説明および確認の要求。確認の要求とは、次のコンテンツ・パッケージをユーザーに配信する前にコンテンツ・パッケージの受信確認が必要かどうかを意味します。詳細は、2-24 ページの「確認の説明」を参照してください。
- **有効期限ポリシー** -- この契約に基づくサブスクリプションの有効期限情報です（表 2-6 を参照）。これには、タイム・ゾーンを含む開始日および停止日、サブスクリプションの最大配信数、有効期限の優先順位（表 2-7 を参照。時間ベース、数量ベース、最初、または最後のいずれか）などが含まれます。

- **プル配信ルール** -- 「**プル配信の有効化**」が選択されている場合、一連の制御によってコンテンツがサブスクライバにオンデマンドで配信されます。プル配信ルールは、コンテンツ・プルが可能な日毎の期間（指定開始時間から指定された時間継続する期間）を記述します（表 2-8 を参照）。タイム・ゾーン、プル・コンテンツ配信の開始時間（時間および分）、コンテンツ配信に有効な期間（時間および分）、およびプル配信が可能な曜日または日付を指定します。

  注意： コンテンツ配信を曜日または日付で制限する場合は、適切なオプションを選択してください。両方のオプションを一緒に使用してコンテンツ配信を制限する場合は、両方の条件を満たす必要があります。たとえば、曜日として「M」または「月曜日」を選択し、日付として「3」を選択すると、コンテンツ配信は 3 日の月曜日のみに限定されます。

- **プッシュ配信ルール** -- 「**プッシュ配信の有効化**」が選択されている場合、一連の制御によってコンテンツが時間ベースのスケジュールでサブスクライバに自動的に配信されます。プッシュ配信ルールでは、コンテンツ・プッシュが可能な日毎の期間（指定開始時間から指定された時間継続する期間）を記述します（表 2-8 を参照）。タイム・ゾーン、プッシュ・コンテンツ配信の開始時間（時間および分）、コンテンツ配信のための合計継続時間（時間および分）、継続時間ごとのコンテンツ更新回数、およびプッシュ配信が可能な曜日または月の日付を指定します。

  注意： コンテンツ配信を曜日または日付で制限する場合は、適切なオプションを選択してください。両方のオプションを一緒に使用してコンテンツ配信を制限する場合は、両方の条件を満たす必要があります。たとえば、曜日として「M」または「月曜日」を選択し、日付として「3」を選択すると、コンテンツ配信は 3 日の月曜日のみに限定されます。

契約の取引条件の作成

契約の取引条件を作成するには、「**取引条件**」タブをクリックします。「**契約の作成 : 取引条件**」ページが表示されます。このページでは、ユーザーがオファーにサブスクライブする際に同意する必要のある合意事項を定義する取引条件が定義されています。取引条件の名前および説明を入力し、契約に関連する合意内容を入力できます。取引条件の詳細の入力が終了したら、「**OK**」をクリックして変更を保存すると、「**オファー管理 : 契約**」ページに戻ります。

サブスクリプションの管理

ユーザーがオファーにサブスクライブすると、サブスクリプションが作成されます。サブスクリプションのリストは「**サブスクリプション管理**」ページで確認できます。

Oracle Enterprise Manager Syndication Services の管理ページで「**サブスクリプション**」をクリックします。「**サブスクリプション管理**」ページが表示されます。このページでは、次のことができます。

- サブスクリプション（一般情報）の表示または編集
- サブスクリプションのオファー情報の表示または編集
- サブスクリプションの終了
- 期限切れおよび終了したサブスクリプションのパージ

サブスクリプション（一般情報）の表示または編集

サブスクリプションを表示または編集するには、「**サブスクリプション管理**」ページで、編集するサブスクリプションの名前を「**サブスクリプション**」列でクリックするか、サブスクリプションのラジオ・ボタンを「**選択**」列でクリックして選択してから、「**編集**」をクリックします。「**サブスクリプションの編集 : 一般**」ページが表示されます。このページでは、サブスクリプションの一般情報を次のグループに分類しています。

- **一般情報** -- サブスクライバの名前、サブスクリプションの開始日、パッケージ順序の状態、およびサブスクリプションの状態 （アクティブ、保留、または停止）。変更できるサブスクリプション情報は、サブスクリプションの状態のみです。サブスクリプションを一時的に終了するには、状態を「保留」に指定してください。サブスクリプションが保留になると、再びアクティブにするまで、ユーザーはプルまたはプッシュのコンテンツ配信ができません。状態が「保留」のサブスクリプションは、「アクティブ」の状態に戻すことができます。ただし、状態を「終了」に変更した場合は、サブスクリプションの状態を「アクティブ」に戻すことはできません。サブスクリプションの状態が「終了」に設定されている場合、サブスクリプションは永久に停止し、「**サブスクリプション管理**」ページの「**期限切れおよび終了のパージ**」をクリックすることでのみパージできます。
- **確認** -- 次のコンテンツ・パッケージをユーザーに配信する前に、このサブスクリプションに対するコンテンツ・パッケージの受信確認が必要かどうかを示します。
- **有効期限ポリシー** -- サブスクリプションの有効期限が時間ベースか数量ベースかあるいはその両方に基づくか（有効期限の優先順位のカテゴリの説明は、表 2-7 を参照）、サブスクリプションが期限切れとなる日付と時刻、およびサブスクリプションの残りの配信数。有効期限ポリシーのカテゴリの説明は、表 2-6 を参照してください。
- **配信ルール** -- コンテンツの配信方法（プル配信、プッシュ配信またはその両方）。プル配信は、ユーザーがコンテンツをオンデマンドで受信することを示し、プッシュ配信は、時間ベースのスケジュールに基づいてコンテンツがサブスクライバに自動的に配信されることを示します。プル配信およびプッシュ配信の両方の配信ルールついて、タイ

ム・ゾーンが開始時間（時間および分）とともに表示されています。コンテンツを配信可能な合計時間（時間および分）が指定されています。配信する曜日または日付が指定されています。最後の配信が完了した日付と時間も指定されています。プッシュ配信の場合のみ、1 日に許可される更新回数も表示されています。配信ルールのカテゴリの説明は、表 2-8 を参照してください。

サブスクリプションのオファー情報の表示または編集

このサブスクリプションのオファー情報を表示または編集するには、「**サブスクリプション管理**」ページで「**オファー**」タブをクリックします。「**サブスクリプションの編集 : オファー**」ページが表示されます。このページでは、このサブスクリプションのオファー情報を次のグループに分類しています。いずれのフィールドも編集できないので注意してください。

- **一般情報** -- オファーの説明。
- **コンテンツ使用方法プロパティ** -- 製品名、知的所有権の所有者名、コンテンツの知的所有権のステータス（表 2-4 を参照）およびコンテンツ使用パターン（表 2-5 を参照）。
- **コンテンツ・プロバイダ情報** -- 名前およびリソース名。

サブスクリプションの終了

サブスクリプションを終了するには、「**サブスクリプション管理**」ページで、終了するサブスクリプションの名前を「**サブスクリプション**」列でクリックするか、サブスクリプションのラジオ・ボタンを「**選択**」列でクリックして選択してから、「**終了**」をクリックします。サブスクリプションの状態が「終了」に変更されます。サブスクリプションの状態は、いったん「終了」に設定すると再び「アクティブ」に変更できないため注意してください。サブスクリプションの状態が「終了」の場合のみ、「**サブスクリプション管理**」ページの「**期限切れおよび終了のページ**」をクリックしてパージできます。

期限切れおよび終了したサブスクリプションのパージ

期限切れまたは終了したサブスクリプションをパージするには、「**サブスクリプション管理**」ページで「**期限切れおよび終了のページ**」をクリックします。期限切れまたは終了したサブスクリプションが、すべてサブスクリプション・リストから削除されます。

アクセス・ログの確認およびパージ

アクセス・ロギングが有効の場合は、Syndication Services を使用するユーザーが実行した各アクセス操作のエントリを含むアクセス・ログを確認できます。

Oracle Enterprise Manager Syndication Services の管理ページで「**アクセス・ログ**」をクリックします。「**アクセス・ログ**」ページが表示されます。このページでは、管理者が検索フォームを使用して、様々な基準でフィルタ処理されたエントリ・レコードのサブセットを選択できます。管理者は基本または拡張検索フォーム（拡張リンクに注意）を使用して、ユーザー、イベント・タイプ（access または admin）、サブスクリプション ID、および日付時間範囲ごとにログ内のアクセス・エントリを検索できます。検索結果は、常に最大 200 エントリに限定されます。検索フォームに値を指定せずに「**実行**」をクリックすると、システム・イベントをすべて含むリストが出力されます。「**選択**」列でエントリのラジオ・ボタンをクリックしてエントリを選択し、次に「**表示**」をクリックすると、「**ログの表示**」ページが表示されます。このページには、特定ユーザーによる Syndication Services へのアクセスに関する詳細なエントリ・レコードが含まれています。コンテンツ配信の問題のトラブルシューティングには、エントリ・レコードの詳細を確認します。

パフォーマンスの監視およびチューニング

`syndserver.ear` 内の Syndication Services サーブレット・エンジンおよび Oracle Application Server Metadata Repository に対する関連 JDBC 接続プールは、Oracle Enterprise Manager やその他の標準データベースの監視およびチューニング用ユーティリティを使用してすべて監視できます。

OC4J スタンドアロン環境では、パフォーマンス情報は次の場所から使用できます。

`http://<oc4j-host-name>:<port-number>/dmsoc4j/Spy`

高度な管理タスクの実行

この項では、高度な管理タスクの一部を説明します。「モデリングの観点からの Syndication Services 管理機能の ICE へのマッピング」では、契約や取引条件、オファー、および Syndication Services のプロパティについて、Syndication Services の管理機能が ICE の要素と属性にどのようにマップされるかを説明します。「確認の説明」では確認の動作方法について説明し、「ネゴシエーション」ではネゴシエーション・プロセスについて説明します。

モデリングの観点からの Syndication Services 管理機能の ICE へのマッピング

「契約および取引条件」、「オファーのプロパティ」および「Syndication Services のプロパティ」では、契約や取引条件、オファー、および Syndication Services のプロパティに関する Syndication Services の管理機能が ICE 1.1 の要素および属性とどのように対応するかを説明します。

契約および取引条件

表 2-9 は、Syndication Services クライアント API の管理フィールドによる契約プロパティと ICE の要素および属性とのマッピングを示します。

表 2-9 契約プロパティのマッピング

管理フィールド	Syndication Services クライアント API	ICE の要素および属性	備考
名前	N/A	N/A	管理者が契約の識別用にのみ使用する。
説明	N/A	N/A	管理者が契約の記述用にのみ使用する。
確認の要求	N/A	N/A	選択すると、この契約に基づくすべてのサブスクリプションでパッケージ配信の確認が必要になる。詳細は、2-24 ページの「確認の説明」を参照してください。
有効期限の優先順位	Offer.getExpirationPriority()	expiration-priority@ice-offer	
開始日	DeliveryPolicy.getStartDate()	start-date@ice-delivery-policy	
停止日	DeliveryPolicy.getStopDate()	stop-date@ice-delivery-policy	
数量	Offer.getQuantity()	quantity@ice-offer	
プル配信の有効化	N/A	N/A	選択すると、対応する ICE-offer にプル配信ルールが追加される。
プル開始時間	DeliveryRule.getStartTime()	start-time@ice-delivery-rule	PUSH DeliveryRule と区別するため、DeliveryRule モードが PULL に設定される。
プル継続時間	DeliveryRule.getDuration()	duration@ice-delivery-rule	PUSH DeliveryRule と区別するため、DeliveryRule モードが PULL に設定される。

表 2-9 契約プロパティのマッピング（続き）

管理フィールド	Syndication Services クライアント API	ICE の要素および属性	備考
プル曜日指定（週）	DeliveryRule.getWeekDay ()	weekday@ice-delivery-rule	PUSH DeliveryRule と区別するため、DeliveryRule モードが PULL に設定される。
プル日付指定（月）	DeliveryRule.getMonthDay()	monthday@ice-delivery-rule	PUSH DeliveryRule と区別するため、DeliveryRule モードが PULL に設定される。
プッシュ配信の有効化	N/A	N/A	選択すると、対応する ICE-offer にプッシュ配信ルールが追加される。
プッシュ開始時間	DeliveryRule.getStartTime()	start-time@ice-delivery-rule	PULL DeliveryRule と区別するため、DeliveryRule モードが PUSH に設定される。
プッシュ継続時間	DeliveryRule.getDuration()	duration@ice-delivery-rule	PULL DeliveryRule と区別するため、DeliveryRule モードが PUSH に設定される。
プッシュ継続時間当たりの更新回数	DeliveryRule.getMinNumUpdates()	min-num-updates@ice-delivery-rule	PULL DeliveryRule と区別するため、DeliveryRule モードが PUSH に設定される。
プッシュ曜日指定（週）	DeliveryRule.getWeekDay()	weekday@ice-delivery-rule	PULL DeliveryRule と区別するため、DeliveryRule モードが PUSH に設定される。
プッシュ日付指定（月）	DeliveryRule.getMonthDay()	monthday@ice-delivery-rule	PULL DeliveryRule と区別するため、DeliveryRule モードが PUSH に設定される。

表 2-10 は、Syndication Services クライアント API の管理フィールドによる取引条件プロパティと ICE の要素および属性とのマッピングを示します。

表 2-10 取引条件プロパティのマッピング

管理フィールド	Syndication Services クライアント API	ICE の要素および属性	備考
名前	BusinessTerm.getName()	name@ice-business-term	
取引条件	BusinessTerm.getText()	text@ice-business-term	
N/A	N/A	type@ice-business-term	常に「licensing」に設定される。

オファーのプロパティ

表 2-11 は、ICE の要素および属性に対する、Syndication Services クライアント API の管理フィールドによるオファー・プロパティのマッピングを示します。

表 2-11 オファー・プロパティのマッピング

管理フィールド	Syndication Services クライアント API	ICE の要素および属性	備考
名前	N/A	N/A	オファーの識別に使用される。
説明	Offer.getDescription()	description@ice-offer	
オファーの状態	N/A	N/A	オファーの状態を管理するために管理者が使用する。ユーザーがカタログを要求したときに、状態がアクティブなオファーのみが表示される。
製品名	Offer.getProductName()	product-name@ice-offer	
権限所有者	Offer.getRightsHolder()	rights-holder@ice-offer	
知的所有権のステータス	Offer.getIPStatus()	ip-status@ice-offer	
基本使用	Offer.isAtomic ()	atomic-use@ice-offer	
編集可能	Offer.isEditable()	editable@ice-offer	
クレジットの表示	Offer.hasShowCredit()	show-credit@ice-offer	
使用方法必須	Offer.hasUsageRequired()	usage-required@ice-offer	
コンテンツ・プロバイダ名	N/A	N/A	このオファーの提供元の識別に使用される。

表 2-11 オファー・プロパティのマッピング（続き）

管理フィールド	Syndication Services クライアント API	ICE の要素および属性	備考
リソース名	N/A	N/A	コンテンツ・プロバイダ側からこのオファーに関連付けられているリソースの識別に使用される。

Syndication Services のプロパティ

表 2-12 は、ICE の要素および属性に対する、Syndication Services クライアント API の管理フィールドによる Syndication Services プロパティのマッピングを示します。

表 2-12 Syndication Services プロパティのマッピング

管理フィールド	Syndication Services クライアント API	ICE の要素および属性
インスタンス ID	Sender.getSenderID()	sender-id@ice-sender
インスタンス名	Sender.getName()	name@ice-sender
ユーザー・エージェント名	Header.getUserAgent()	ice-user-agent/ice-header

確認の説明

Syndication Services では、コンテンツをパッケージの形式で配信し、次のコンテンツ・パッケージを送信する前に、パッケージが正しく配信されたかを確認するようにユーザーに対して要求できます。サーバーでは最後の要求以降のコンテンツの更新のみを送信するため、送信された更新がすべて正しく受信および処理されているという確認をサーバーが受信するまで、次の更新は送信されません。

これは、契約に対して確認フラグが設定されている場合、その契約に依存するすべてのサブスクリプションでは、ユーザーが新規コンテンツを受信する前に明示的に確認を送信する必要がある、ということを意味します。この動作は、プルおよびプッシュの配信ルール両方に適用されます。たとえば、ユーザーが確認を送信する前にコンテンツのプル更新を行うと、Syndication Services では、「ICE-602: 未処理の確認が多すぎます」という例外を使用して応答します。これはデフォルトの動作です。

ユーザーからのコンテンツ配信の確認を受信しないように、またはコンテンツ配信の確認でコンテンツ配信が多すぎる場合に、ユーザーが確認や拒否をしなくて済むように、次の 2 つのプロパティを使用して境界を設定できます。

- `oracle.syndicate.server.delivery.max_outstanding_confirms`

  未処理の確認の最大数 : このプロパティでは、コンテンツ・パッケージの配信に対してユーザー（サブスクライバ）からの確認を必要とするあるサブスクリプションについて、確認せずにユーザーに配信されるコンテンツ・パッケージの最大数を定義します。

たとえば、5 という値を設定すると、このサブスクリプションに対して、未確認の配信に続いてコンテンツ・パッケージが 5 つ配信されるということを意味します。6 つ目以降の配信は確認が受信されるまで保留になります。このプロパティの詳細は、2-28 ページの「Syndication Services の拡張プロパティ」を参照してください。

- `oracle.syndicate.server.delivery.max_unconfirmed_packages`

  未確認パッケージの最大数 : このプロパティでは、コンテンツ・パッケージの配信に対してユーザー（サブスクライバ）からの確認を必要とするサブスクリプションが指定されている場合に、2 つの配信確認間で許可される未確認のコンテンツ・パッケージまたは拒否したコンテンツ・パッケージの最大数を定義します。たとえば、2 という値を設定すると、ユーザーは、任意の 2 つのコンテンツ配信確認間で 2 つ以上のコンテンツ配信のパッケージ受信を未確認または拒否したままにできないということを意味します。このパラメータを使用して、ユーザーが契約に定められている数以上のコンテンツ・パッケージ配信を受信しないようにできます。このプロパティの詳細は、2-28 ページの「Syndication Services の拡張プロパティ」を参照してください。

ネゴシエーション

Syndication Services では、サブスクリプション時にユーザーはオファーをカスタマイズできます。サポートされているカスタマイズは、次の変更に限定されています。

- ユーザーは、オファーで提供されている配信ルールの中から、サブスクリプションに使用する一連の配信ルールを選択できます。たとえば、ユーザーにプルおよびプッシュの配信ルールを両方とも含むオファーが提示された場合、ユーザーはプル配信ルールにのみサブスクライブすることをサブスクリプション時に決定できます。したがって、プッシュ配信ルールはサブスクリプション・リクエストから除外されます。
- ユーザーは、プッシュ URL（`http`、`https`、`ftp` または `mailto` に基づく）とコンテンツ配信オプション（ICE 配信または非 ICE 配信）を指定して、プッシュ配信ルールをカスタマイズできます。指定した URL に認証が必要な場合、ユーザーは必要な資格証明を指定できます。詳細は、3-13 ページの「配信ポリシー」を参照してください。

サブスクリプション要求に指定されたオファーを他の形式に変更する場合、Syndication Services では要求を拒否し、例外「`ICE-441: 代替案`」を使用して応答します。

スケジューラの構成

Syndication Services には、時間ベースのコンテンツ・プッシュを管理するスケジューラ・コンポーネントが備わっています。スケジューラ・コンポーネントは Syndication Services J2EE アプリケーションの一部で、Syndication Services のインストール時にデプロイされます。この項では、スケジューラの操作と、スケジューラのチューニングに使用できるプロパティについて説明します。

新規スケジューラ・ルールの割当て

プッシュ配信ルールを含む新規サブスクリプションが作成されると、Syndication Services は新規スケジューラ・ルールを Oracle Application Server Metadata Repository に格納します。スケジューラ・ルールは、更新の頻度と具体的なプッシュ日時の計算に使用されます。スケジューラ・コンポーネントでは、メタデータ・リポジトリ内に新規スケジューラ・ルールがあるかどうかを定期的にチェックします。新規スケジューラ・ルールが見つかると、スケジューラがそのルールの所有者になり、その時点からルールのタスク処理を開始します。

Syndication Services 管理の「システム・プロパティ」ページにある `monitor frequency` システム・パラメータは、スケジューラ・コンポーネントが新規ルールのチェックを行う頻度を管理します。監視頻度のデフォルト値は 1 分です。これは、新規サブスクリプションに関連付けられているプッシュ処理は、最大でもそのサブスクリプションが作成された 1 分後に実行されることを意味します。新規作成サブスクリプションをより速やかに処理する場合は、監視頻度を上げることもできますが、結果としてメタデータ・リポジトリへのラウンドトリップが頻発することになり、システム全体のパフォーマンスに影響を及ぼします。

スケジューラ・ルールの処理

スケジューラ・インスタンスがルールを所有すると、スケジューラ・コンポーネントは次のプッシュ・タスクの日時を計算し、タイマーをスケジュールします。Syndication Services 管理の「**システム・プロパティ**」ページにある `timer pool size` システム・パラメータは、各スケジューラ・インスタンスで同時にアクティブになるタイマーの数を管理します。デフォルト値は 16 です。これは、どの時点でも、スケジューラにアクティブなタイマーが 16 個あることを意味します。つまり、そのインスタンスに割り当てられているすべてのスケジューラ・ルール間にある、16 個の各プッシュ・タスクにつき、タイマーが 1 つあることになります。これより多い数のタスクを短時間に処理する計画がある場合は、タイマ・プール・サイズを増やすことができます。

システムの停止または再起動後のスケジューラのリカバリ

Syndication Services が停止された場合（デフォルト・インストレーションでは、OC4J_Portal インスタンスの停止を意味します）は、スケジューラ・コンポーネントも停止します。Syndication Services の停止時には、スケジュールされたプッシュ配信は実行されません。

再起動時に、Syndication Services の各スケジューラ・インスタンスは、割り当てられているスケジューラ・ルールをリカバリし、コンテンツのプッシュ配信を再開します。プッシュ

処理が停止時間中にスケジュールされていたために実行されなかった場合は、再起動するとプッシュ処理が先に実行され、その後、通常の操作が再開します。

ソフトウェアの障害により OC4J インスタンスが 1 つ停止した場合は、Oracle Process Manager and Notification（OPMN）Server により自動的にリカバリされます。リカバリ時に、そのインスタンスに関連付けられているスケジューラ・ルールは前述した動作に従ってリカバリされます。

Oracle Application Server クラスタでの Syndication Services スケジューラの使用

Syndication Services が複数の OC4J インスタンスにデプロイされているクラスタ環境では、Syndication Services のすべてのスケジューラ・コンポーネントにより、新規スケジューラ・ルールは定期的にチェックされます。ただし、割り当てられた後は、各スケジューラ・ルールは 1 つの OC4J インスタンスのみが所有します。割当てプロセスにより、スケジューラ・ルールは使用可能なすべての OC4J インスタンスに均等に配分され、ロード・バランシングが達成されます。したがって、各 OC4J インスタンスは一連のスケジューラ・ルールを独自に所有します。これは、すべてのスケジューラ・ルールを処理するには、すべての OC4J インスタンスが実行可能である必要があることを意味します。

Syndication Services のスケジューラ・インスタンスでは、OC4J_ID を使用してスケジュールされるルールの所有者を識別します。Syndication Services がデプロイされているアイランドや OC4J インスタンスが削除されるときは常に、これらのアイランドや OC4J インスタンス用としてマークされているルールを解放し、Syndication Services の他のスケジューラ・インスタンスで引き継ぐ必要があります。

Syndication Services では、分散構成管理（DCM）のビジネス・ルール・フレームワークを使用してこの場合の処理を行います。DCM のビジネス・ルール・フレームワークでは、Syndication Services などのコンポーネントがプラグインを登録し、関連イベントの発生時に対応するメソッドを起動できます。プラグインを登録する構文は、次のとおりです。

```
$<MIDTIER_ORACLE_HOME>/dcm/bin/dcmctl registerplugin -f $<MIDTIER_ORACLE_
HOME>/syndication/config/bizruleReg.xml
```

プラグインが登録されると、Syndication Services は OC4J インスタンスの削除やアイランドの削除イベントをリスニングします。このようなイベントが発生するときは常に、プラグインで対応する所有者フラグをクリーン・アップします。続いて、別の Syndication Services のスケジューラ・インスタンスがこれらのルールを取得します。

Syndication Services の拡張プロパティ

この項では、Syndication Services の拡張プロパティについて説明します。これらの拡張プロパティは、2-28 ページの「Syndication Services の拡張プロパティの設定方法」に説明されている手順でのみ設定できます。拡張プロパティには、次のものが含まれます。

- `oracle.syndicate.transport.ftp.act_connect_mode`：アクティブまたはパッシブ FTP モードを選択する FTP トランスポート・プロパティ。使用可能な値は、`true` または `false` です。デフォルト値は `false` で、パッシブ FTP モードです。
- `oracle.syndicate.server.delivery.max_outstanding_confirms`：未確認配信に続いてユーザー（サブスクライバ）に配信されるコンテンツ・パッケージの最大数を定義するプロパティ。配信が確認されるまで、以降の配信は保留されます。使用可能な値は、任意の正の整数です。デフォルト値は 1 です。
- `oracle.syndicate.server.delivery.max_unconfirmed_packages`：配信が保留される前の 2 つの配信確認間で許可される、未確認または拒否されたコンテンツ・パッケージの最大数を定義するプロパティ。使用可能な値は、任意の正の整数です。デフォルト値は 1 です。

Syndication Services の拡張プロパティの設定方法

拡張プロパティは、Syndication Services の機能を細かくチューニングするために使用します。このようなパラメータのカスタマイズは一般的とはみなされないため、Oracle Application Server Syndication Services の管理ページを使用してプロパティを設定することはできません。拡張プロパティは、Oracle Application Server Metadata Repository で直接設定する必要があります。

拡張プロパティの設定の際は、Oracle Application Server Metadata Repository で SQL*Plus セッションを開き、SQL 文を実行して適切なプロパティ値を設定するという手順に従います。

UNIX 環境で SQL*Plus セッションを開くには、次の一連のコマンドを使用します。

```
cd <ORACLE_HOME>
setenv ORACLE_HOME <ORACLE_HOME>
setenv ORACLE_SID  <ORACLE_SID>
bin/sqlplus "sys/<sys_password> as sysdba"
```

SQL*Plus セッションで次のコマンドを使用し、セッション・スキーマを Syndication Services スキーマに設定します。

```
alter session set current_schema=dsgateway;
```

PROPERTIES 表には、Syndication Services で使用されるプロパティの名前、値および説明が含まれます。プロパティの現在の値は、表の SELECT 文を実行すると検証できます。

```
SELECT propname, propvalue
FROM PROPERTIES;
```

プロパティ値は、次の構文を使用して更新できます。

```
update PROPERTIES set PROPVALUE = 'pvalue' where PROPNAME = 'pname';
```

新規プロパティは、次の構文を使用して挿入できます。

```
insert into PROPERTIES (PROPNAME, PROPVALUE, DESCRIPTION) VALUES ('pname', 'pvalue',
'pdesc');
```

3

クライアント・アプリケーションの開発

この章では、クライアント・アプリケーションの開発について説明します。

Syndication Services では、Java ベースの Java 2 Platform Standard Edition（J2SE）または Java 2 Platform Enterprise Edition（J2EE）アプリケーションに Syndication Services を統合するための Java クライアント・ライブラリを提供しています。Syndication Services のクライアント・ライブラリを使用して、ユーザーは、付与されているオファーのカタログのリクエスト、1 つ以上のオファーへのサブスクライブ、および関連コンテンツの受信を行うことができます。次の各項では、実行する際の Syndication Services クライアント・アプリケーションのコード・シーケンスのサンプルについて説明します。

- Syndication Services の接続のオープン
- オファーのカタログの取得
- オファーへのサブスクライブ
- コンテンツの配信
- サブスクリプションの有効期限ステータスの検証
- サブスクリプションの取消し
- サブスクリプションに関連付けられているアクティビティの検証

サンプルの完全なソース・ファイルを含む一連のデモ・ファイルは、Oracle Technology Network Japan（OTN-J）の Web サイト（`http://otn.oracle.co.jp/sample_code/index.html`）からダウンロードできます。

`SampleSyndicationClient.java` プログラムの一部はこの章で説明していますが、完全なリストは C-2 ページの「SampleSyndicationClient.java プログラムのリスト」を参照してください。Syndication Services クライアント API の使用方法を説明した API ドキュメントは、『Oracle Application Server Syndication Services API Reference』（Javadoc）として用意されています。クライアント・ライブラリ（`syndclient.jar`）は、UNIX の場合は *`$ORACLE_HOME`*`/syndication/lib`、Windows の場合は *`<ORACLE_HOME>`*`¥syndication¥lib` にあります。

Syndication Services のクライアント・ライブラリは、Information Content Exchange（ICE）1.1 標準をモデルとして使用しています。このライブラリは、ICE 1.1 準拠のサーバーである Syndication Services との情報交換では、実際に ICE クライアントとして機能します。ICE 標準は、シンジケーション関係を確立するためのデータ構造と情報交換を定義するコンテンツ・シンジケーション標準です。Syndication Services クライアント・ライブラリは汎用的で相互運用可能な ICE クライアントを実装していますが、クライアント・ライブラリ上のドキュメントは、Syndication Services サーバーへのアクセス時の使用方法に関連する操作のサブセット、データ構造および属性の説明が中心です。Syndication Services と ICE とのマッピングの詳細は、2-21 ページの「契約および取引条件」、2-23 ページの「オファーのプロパティ」および 2-24 ページの「Syndication Services のプロパティ」を参照してください。ICE 1.1 仕様は、`http://www.icestandard.org/` を参照してください。

Syndication Services の接続のオープン

Syndication Services と通信するアプリケーションを構築する場合、クライアント・コードで最初に SyndicateConnection インスタンスを取得する必要があります。例 3-1 は、SyndicateConnection インスタンスの取得方法を示します。

例 3-1 SyndicateConnection インスタンスの取得

```
// Acquire a SyndicateConnectionFactory,
// optionally set the connection parameters,
// such as proxy info, timeout, and credentials.
SyndicateConnectionFactory scf = SyndicateConnectionFactory.getInstance();
// scf.setTimeout(1000);
// scf.setProxy("myproxyhost", iMyProxtPort);
// Create a default XML state handler storing subscription state in a file.
SyndicateClientStateHandler scsh =
 scf.getDefaultSyndicateClientStateHandler("synd-client.xml");
SyndicateConnection sc =
 scf.createSyndicateConnection("http://myias/syndserver/server",  // server url
                                "myusername",  // username
                                "mypassword",  // password
                                scsh);
```

SyndicateConnection インスタンスは、SyndicateConnectionFactory オブジェクトを使用して取得します。後者には、作成前に接続プロパティを設定するための API が含まれています。接続プロパティには、タイムアウト、HTTPS 接続に使用する資格証明および接続に使用する HTTP プロキシなどが含まれます。SyndicateConnection インスタンスをオープンすると、Syndication Services との間で HTTP 接続が確立され、有効なユーザーとして認証されます。Oracle Application Server のデフォルト・インストールでは、Syndication Services サーバーは OC4J_Portal インスタンス内にデプロイされ、URL の相対パスは /syndserver/server（たとえば、http://myias:7777/syndserver/server）です。Syndication Services への接続を正常に確立するには、指定するユーザー名とパスワードが、有効な Syndication Services ユーザーの資格証明と一致する必要があります。Syndication Services ユーザーの管理の詳細は、2-4 ページの「ユーザー管理の理解」を参照してください。

Syndication Services クライアントは、通常はステートフルです。実際、クライアントは作成した一連のサブスクリプションを追跡する必要があり、コンテンツの増分更新を受信すると、各サブスクリプションの状態を追跡する必要があります。SyndicateConnection インスタンスをネストする場合は、SyndicateClientStateHandler のインスタンスをオプションで指定できます。SyndicateClientStateHandler は、サブスクリプションの作成時や更新時など、クライアントの状態の変更が必要になるたびに SyndicateConnection インスタンスにより呼び出される一連のインタフェースを公開します。デフォルトの SyndicateClientStateHandler インスタンスは、サブスクリプション状態情報を XML ファイルに格納します。サブスクリプション状態のデータベースへ

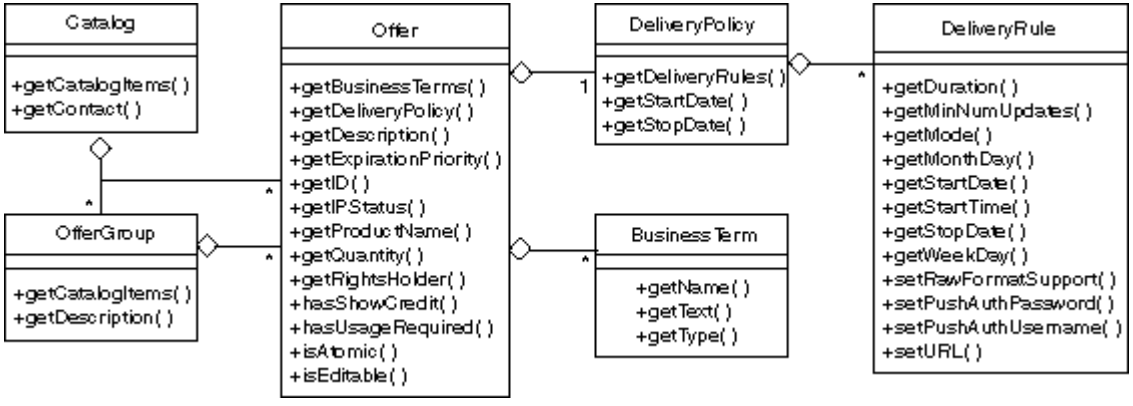
の格納など、SyndicateClientStateHandler インタフェースのカスタム実装を提供することもできます。

SyndicateConnection オブジェクトは同期化されません。呼出しを作成するプログラマは、マルチスレッド・アプリケーションを作成する場合に、オブジェクトのアクセスを同期化するか、接続のプールを作成する必要があります。

オファーのカタログの取得

カタログとは、サブスクリプションに使用可能な一連のオファーです。カタログ構造を Unified Modeling Language（UML）図として図 3-1 に示します。

図 3-1 カタログの構造と主要プロパティ



カタログとは一連のオファーであり、オプションでオファー・グループにグループ化されています。オファーはコンテンツ単位（たとえば、リポジトリ内の 1 つのディレクトリ）を表し、コンテンツの更新を受信するためにユーザーがサブスクライブできます。オファーは、コンテンツ使用方法プロパティ用のアクセッサ、使用ライセンス（ある場合）用のアクセッサ、およびサブスクリプションが確立されている場合にオファー・コンテンツが配信されるモードと時間用のアクセッサを提供します。例 3-2 は、カタログの取得方法とそのオファーを介した参照方法を示します。この例では、カタログはコンテンツ・プロバイダごとにグループ化されています。

例 3-2 カタログの取得とオファーを介した参照

```
Catalog cat = sc.getCatalog();
Offer   off = getFirstOffer(cat);
}

  private Offer getFirstOffer(Catalog cat)
    throws SyndicateException
  {
    // Iterates through the catalog and returns the
    // first offer encountered.
    Offer   off = null;
    Iterator it = cat.getCatalogItems();
    while (it.hasNext() && (off == null)) {

      CatalogItem item = (CatalogItem)it.next();
      if (item.getCatalogItemType() == CatalogItem.OFFER) {
        off = (Offer) item;
      }
      else {
        // You got an offer group.
        off = getFirstOffer((OfferGroup) item);
      }
    }

    // Print a few offer details.
    System.out.println("offer: "+off.getID()+" - "+off.getDescription());
    DeliveryPolicy dp = off.getDeliveryPolicy();
    if (dp.getStartDate() != null) {
      System.out.println("¥t start date: "+dp.getStartDate());
    }
    if (dp.getStopDate() != null) {
      System.out.println("¥t  stop date: "+dp.getStopDate());
    }
    Iterator dlrs = dp.getDeliveryRules();
    while (dlrs.hasNext()) {
      DeliveryRule dlr = (DeliveryRule) dlrs.next();
      System.out.println("¥t  dlr mode: "+dlr.getMode());
    }
    return off;
  }
private Offer getFirstOffer(OfferGroup offgrp)
    throws SyndicateException
  {
    Offer   off = null;
    Iterator it = offgrp.getCatalogItems();
    while (it.hasNext() && (off == null)) {
```

```
    CatalogItem item = (CatalogItem)it.next();
    if (item.getCatalogItemType() == CatalogItem.OFFER) {

      off = (Offer) item;
    }
    else {
      // You got an offer group.
      off = getFirstOffer((OfferGroup) item);
    }
  }
  return off;
}
```

オファーへのサブスクライブ

例 3-3 は、ユーザーがオファーにサブスクライブする方法を示します。オファーへのサブスクライブは、提供されているオファーをユーザーのニーズに応じてカスタマイズし、SyndicateConnection.subscribe メソッドを呼び出して行います。典型的なカスタマイズには、宛先アドレスの設定とプッシュされるコンテンツ・パッケージのコンテンツ形式の設定が含まれます。ユーザーがサブスクライブ時に実行できるオファーのカスタマイズの詳細は、2-25 ページの「ネゴシエーション」を参照してください。

例 3-3 オファーへのサブスクライブ

```
// If the offer contains a push delivery rule,
// you can use the offer APIs to set the destination URL.
DeliveryPolicy dp = off.getDeliveryPolicy();
Iterator   itdlrs = dp.getDeliveryRules();
while (itdlrs.hasNext()) {

  DeliveryRule dlr = (DeliveryRule) itdlrs.next();
  if (DeliveryRule.DELIVERY_RULE_MODE_PUSH.equals(dlr.getMode())) {

    dlr.setURL("http://mysyndicationclient.com/syndclient/listener");
    // Optionally set user name/password.
    // dlr.setPushAuthUsername("me");
    // dlr.setPushAuthUsername("pwd");
    // If raw content is requested (for example, a mailto or
    // ftp url has been supplied), set the raw content flag.
    // dlr.setRawFormatSupport(true);
  }
}

Subscription sbt = sc.subscribe(off);
System.out.println("subscription: "+sbt.getSubscriptionID());
```

コンテンツの配信

オファーへのサブスクリプションが確立された後、オファーの配信ルールに定義されている期間に従ってコンテンツを配信できます。サブスクリプションにプッシュ配信ルールが含まれている場合、Syndication Services は配信ルールの頻度に従って自動プッシュ更新をスケジュールします。提供されている `SyndicateClientServlet` サーブレットを構成して HTTP ベースのリスナーを作成する方法は、『Oracle Application Server Syndication Services API Reference』（Javadoc）という API ドキュメントを参照してください。リスナーは、プッシュ・コンテンツ・パッケージを受信すると、受信されたコンテンツを抽出し、ローカル・ファイル・システムに格納します。提供されている Java API を使用すると、`SyndicateClientServlet` サーブレットを使用して、受信されたコンテンツのカスタム処理を実行できます。

サブスクリプションにプル配信ルールが含まれている場合、例 3-4 で示すようにコンテンツの更新をリクエストできます。

例 3-4 コンテンツの更新のリクエスト

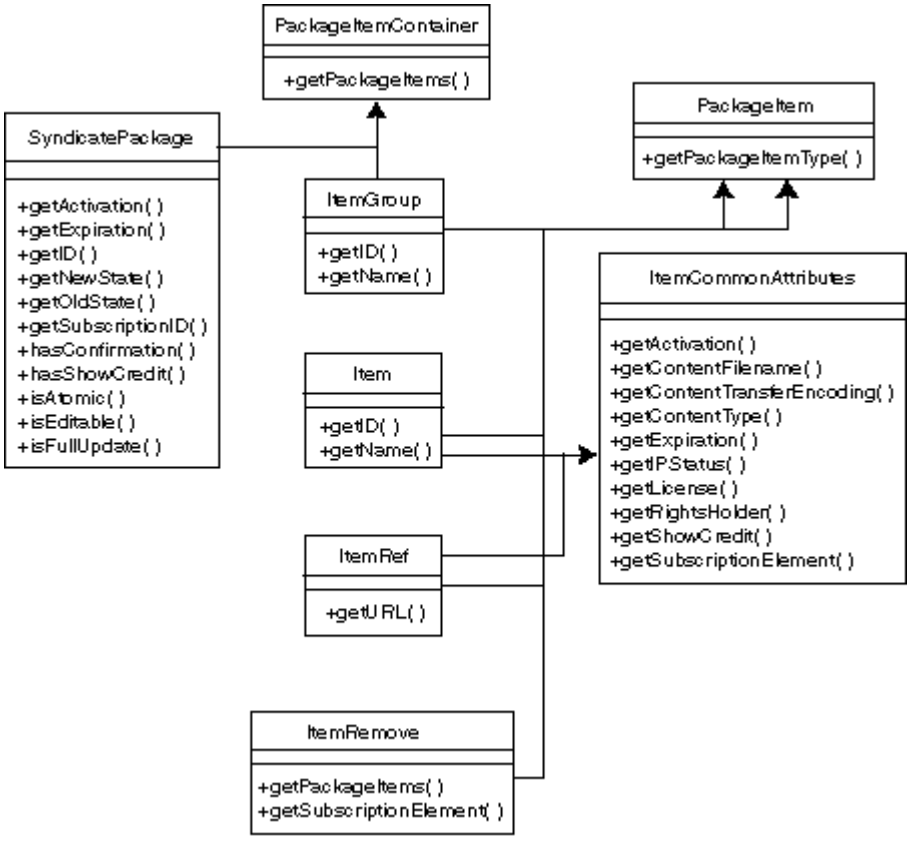
```
// Use a FileSAXPackageHandler instance so that
// syndicate content will be stored in
// the local file system /tmp directory.
FileSAXPackageHandler fsph = FileSAXPackageHandler.getInstance("/tmp/");

// This will get content incremented since the last update.
// The current subscriber state will be fetched from
// the SyndicateClientStateHandler object used by this connection.
// To request a full update of the content versus an incremental
// one, use the following syntax:
// sc.getPackage(sbt.getSubscriptionID(),
//                SyndicateSubscription.STATE_ICE_INITIAL,
//                null,
//                fsph);
SyndicatePackage pkg = sc.getPackage(sbt.getSubscriptionID(), null, fsph);
```

前述のコード例では、提供されている `FileSAXPackageHandler` インスタンスを使用して、受信されたコンテンツをファイル・システムのディレクトリに格納しています。`SAXPackageHandler` インタフェースを実装することにより、Syndication Services クライアント・ライブラリを使用してカスタムのパッケージ・ハンドラを開発できます。詳細は、『Oracle Application Server Syndication Services API Reference』（Javadoc）という API ドキュメントを参照してください。

返される `SyndicatePackage` オブジェクトを使用すると、コンテンツ更新で受信された一連のアイテム全体を反復できます。図 3-2 は、コンテンツ・パッケージの構造と主要プロパティの UML 図を示します。

図 3-2 コンテンツ・パッケージの構造と主要プロパティ



パッケージ確認要求

Syndication Services では、次のコンテンツ・パッケージの送信が可能になる前に、コンテンツ・パッケージの配信が正しいかどうかの確認をユーザーに対して要求できます。サーバーでは最後の要求以降はコンテンツの更新のみを送信するため、送信された更新がすべて正しく受信および処理されたという確認をサーバーが受信するまで、新規の更新は送信されません。

提供されている `SyndicateClientServlet` サーブレットに基づいてリスナーが実装されているプッシュ配信ルールでは、指定された `SAXPackageHandler` インタフェースによりコンテンツ・パッケージが正しく処理された場合、サーブレット・コードはプッシュされたコンテンツ・パッケージを自動的に確認します。また、プッシュ URL が SMTP ベースまたは FTP ベースで、コンテンツの転送中にトランスポート・レベルのエラーが検出されない場合は、ユーザーの状態が自動的に更新され、暗黙的に確認されると考えられます。

プル配信ルールでは、受信されたコンテンツ・パッケージに確認が必要な場合、Syndication Services では追加の配信（プルまたはプッシュ）を実行する前に確認を受信する必要があります。例 3-5 は、確認が必要かどうかを検証する方法、および必要な場合に確認を送信する方法を示します。

例 3-5 確認が必要かどうかの検証と確認の送信

```
    // Check if the content package requires confirmation.
    // If so, confirm it.
    if (pkg.hasConfirmation()) {

     Response resp = _sc.confirm(pkg.getID());
      System.out.println("confirmed "+pkg.getID()+":
"+resp.getCode().getPhrase());
    }
```

サブスクリプションの有効期限ステータスの検証

例 3-6 は、サブスクリプション・ステータスの検証方法を示します。返された情報を使用して、サブスクリプションの設定と有効期限の基準を決定できます。

例 3-6 サブスクリプションのステータスの検証

```
    Status sts = sc.getStatus(sbt.getSubscriptionID());

    Subscription sbtNew = (Subscription) sts.getSubscriptions().next();
    System.out.println(" sbt "+sbtNew.getSubscriptionID()+":");

    int pri = sbtNew.getExpirationPriority();
System.out.println("     expiration priority: "+
 Subscription.EXPIRATION_PRIORITIES[pri]);
    System.out.println("     expiration date: "+sbtNew.getExpirationDate());
    System.out.println("     quantity
 remaining:"+sbtNew.getQuantityRemaining());
```

サブスクリプションの取消し

ユーザーはサブスクリプションが失効する前にサブスクリプションの終了を決定できます。例 3-7 は、サブスクリプションの終了方法を示します。

例 3-7 サブスクリプションの終了

```
    Cancellation canc = sc.cancelSubscription(sbt.getSubscriptionID(),
                                                        "boring", "en_us");
    System.out.println("cancelled: "+canc.getCancellationID());
```

サブスクリプションに関連付けられているアクティビティの検証

ユーザーは、特定のサブスクリプションに関連付けられているアクティビティのログを Syndication Services に対して要求できます。オプションで、時間枠を指定して返されるイベントのセットを制限できます。例 3-8 は、サブスクリプションのアクティビティの取得方法を示します。

例 3-8 サブスクリプションのアクティビティの取得

```
    Events evt = sc.getEvents(null, null, sbt.getSubscriptionID());
    EventLog evtlog = evt.getEventLog();
    Iterator itEvts = evtlog.getEventItems();
    while (itEvts.hasNext()) {

      EventItem ei = (EventItem) itEvts.next();
      if (ei.getType() ==  EventItem.EVENT_TYPE_MSG) {

        EventMsg evtmsg = (EventMsg) ei;
        System.out.println(evtmsg.getRequestStart()+" "
                           +evtmsg.getRequest()+" "+
                           evtmsg.getResponseType());
      }
    }
  }
}
```

クライアント・ライブラリの参照情報

3-12 ページの「オファーのプロパティ」から 3-17 ページの「Item および ItemRef インタフェースの属性」では、Syndication Services サーバーに関連するインタフェース属性について説明します。これらのインタフェースの詳細は、Syndication Services クライアント API の使用方法を説明している『Oracle Application Server Syndication Services API Reference』（Javadoc）を参照してください。

オファーのプロパティ

ここでは、Syndication Services サーバーに関連する次のオファー属性について説明します。

- **ID** : オファーの ID。オファーを一意に識別します。
- **説明** : オファーと関連するコンテンツのテキストによる説明。この説明により、ユーザーはオファーへのサブスクライブに関心があるかどうかを判断できます。
- **製品名** : サブスクリプションまたはオファーを他と区別するために使用される名前。本来は、オファーにわかりやすく短い説明（「Julia Child の新しいフランス料理コラム」など）を提供するために使用されます。
- **基本使用** : true の場合、サブスクリプションの配信済コンテンツをすべて同時に使用する必要があるか、使用しないでください。false の場合、ユーザーはデータのサブセットを必要に応じて使用できます（ライセンス条件で許可されている範囲内に限ります）。
- **編集可能** : true の場合、ユーザーはコンテンツを使用する前に編集または変更できます。false の場合、ユーザーは変更せずにコンテンツを使用すると予想されます。
- **知的所有権のステータス（IPStatus）** : コンテンツの知的所有権のステータスを指定する文字列。ICE 1.1 では、次の固有の文字列値が定義されています。
  - PUBLIC-DOMAIN: コンテンツにはライセンス上の制限はありません。
  - FREE-WITH-ACK: コンテンツ・ソースを明示するという要件以外にライセンス上の制限はありません。
  - SEE-LICENSE: コンテンツには既存のライセンス契約ですでに合意されている制限があります。これは、デフォルト動作を表します。
  - SEVERE-RESTRICTIONS: コンテンツには、特に注意が必要なライセンス上の制限があります。
  - CONFIDENTIAL: コンテンツは機密であり、特に保護する必要があります。
- **権限所有者** : シンジケーション権限の元のソースを指定する文字列。
- **クレジットの表示** : true の場合、サブスクライバはデータのソースを認識していることが明らかです。

- **使用方法必須** : シンジケートされたコンテンツの最終ビューアに関する使用方法データをサブスクライバから返されることをシンジケータが想定するかどうか。Syndication Services では、この使用方法データの形式（またはトランスポート）を定義しませんが、この属性により、シンジケータではサブスクライバに対するデータの必要性をプログラム上で示します。

取引条件

取引条件により、追加コンテンツとパラメータがパーティ間で通信およびネゴシエーションするための手段を指定します。通常、取引条件には、サブスクリプションを受け入れて関連コンテンツを配信する前に、シンジケータによってユーザーの同意を確認するライセンス契約が含まれます。

配信ポリシー

オファー配信ポリシーは、サブスクリプションの継続期間を示します。**配信ポリシー開始日**により、コンテンツの配信が開始する日付が決定されます。サブスクリプションの有効期限は、属性の組合せにより決定されます。オファーの**有効期限の優先度**属性は、サブスクリプションの有効期限を決定する基準を定義します。サブスクリプションは、特定の回数のコンテンツ配信が実行された後、指定された日付または前述の基準の組合せで期限切れにできます。有効期限の優先度の値が `first` の場合は、**第 1 の数量**または**配信ポリシー停止日**に達したときにサブスクリプションが終了します。有効期限の優先度の値が `last` の場合は、**数量**および**配信ポリシー停止日**の両方が達したときにサブスクリプションは終了します。有効期限の優先度の値が `time` の場合は、**配信ポリシー停止日**に達したときにサブスクリプションが終了します。有効期限の優先度の値が `quantity` の場合は、数量に達したときにサブスクリプションが終了します。

配信ルールは、配信が実行される期間を定義します。各配信ルールは、**プッシュ**（コンテンツはシンジケータによりユーザーに自動的に配信される）か**プル**（コンテンツ配信がユーザーにより要求される）のいずれかで、配信が実行可能な年、月、日付、曜日、更新期間の開始日時と終了日時、および実行可能な更新回数を定義できます。

これは、配信が行われる日次的な期間（開始時刻と継続時間により定義）を定義するためのものです。開始日、曜日、日付および停止日は、日次的な期間が適用される日を定義します。最も関連性のある配信ルール属性を次に要約します。

- **mode** : `push` または `pull` いずれかの値。プッシュ配信は、更新がシンジケータにより開始されることを意味します。プル配信は、更新がユーザーにより開始されることを意味します。
- **Monthday** : 配信は指定された月の日付に限定されます。この属性の範囲は1から31まで（1 と 31 を含む）です。または制限なしを意味する特別な値 `any` にするか、月の最終日を意味する特別な値 `last` に指定することも可能です。複数の値を指定する場合は、値を ''（スペース）で区切ります。

- **Weekday** : 配信は指定された曜日に限定されます。この属性の範囲は 1 から 7 まで（1 と 7 を含む）、または制限なしを意味する特別な値 any のいずれかである必要があります。複数の値を指定する場合は、値を ' '（スペース）で区切ります。[ISO860] の 5.2.3 項に従い、曜日は次のように割り当てます。
  - 月曜日は日番号 1
  - 火曜日は日番号 2
  - 水曜日は日番号 3
  - 木曜日は日番号 4
  - 金曜日は日番号 5
  - 土曜日は日番号 6
  - 日曜日は日番号 7
- **Startdate** : 配信ルール期間の開始。この日付が配信ポリシー内の開始日よりも前である場合、開始日と同じ日付と認識されるように実装で処理する必要があります。開始日が指定されていない場合、期間は配信ポリシーで指定されている開始日に始まります。配信ポリシーに開始日が含まれていない場合、期間はただちに開始されます。
- **Stopdate** : 配信ルール期間の終了。この日付が配信ポリシー内の停止日よりも後である場合、停止日と同じ日付と認識されるように実装で処理する必要があります。停止日が指定されていない場合、配信は配信ポリシーの停止日に基づいて終了します。配信ポリシーに停止日が含まれていない場合、配信は終了しません。
- **Starttime** : 期間の開始時刻。サブスクリプションの継続時間に定義されていない場合、協定世界時（UTC）の 00:00:00 が開始時刻として適用されます。対象の日付が 1 日目の場合、期間は開始時刻または開始日の時刻部分のいずれか遅い方で開始します。
- **Duration** : 日次的な期間の長さ。継続時間は 24 時間を超えることはありません。開始時刻および継続時間が指定されていない場合、丸 1 日が範囲となるように期間が指定されます。開始時刻が指定されているが、継続時間が指定されていない場合、開始時刻から 24:00:00 UTC まで期間が継続するように継続時間が指定されているものとして解釈されます。
- **min-num-updates** : 開始時刻および継続時間により特定された期間中にユーザーが受信する更新の回数。Syndication Services では、このパラメータはプッシュ配信ルールにのみ適用されます。
- **url、pushAuthUsername、pushAuthPassword** : 更新が通常の ICE 通信エンド・ポイントに送信されない場合の送信先アドレス（表 3-1 を参照）。Syndication Services では、指定された URL は、プッシュ・パッケージをリスニングするクライアントが実行されている HTTP、または HTTPS プロトコルのいずれかである可能性があります。他にサポートされているプロトコルには、SMTP（例 : `mailto:myemail.com`）または FTP（例 : `ftp://myserver/mydir`）があります。認証が必要な場合は、使用するユーザーのユーザー名とパスワードをオプションで指定できます。

表 3-1 プッシュ配信オプション

プッシュ URL プロトコル	ICE プッシュ (packageFormat=ICE_PACKAGE_FORMAT)	非 ICE プッシュ (packageFormat=NON_ICE_PACKAGE_FORMAT または NON_ICE_NO_LOG_PACKAGE_FORMAT)
HTTP （例：`http://user.com/syndclient/client`）	Syndication Services は、ICE 1.1 仕様に従って、ICE XML パッケージを含むユーザー HTTP エンド・ポイントに対して、HTTP POST を実行する。	Syndication Services は、MIME マルチパート・メッセージを含むユーザー HTTP エンド・ポイントに対して、HTTP POST を実行する。このメッセージの各部はコンテント・パッケージ項目である。NON_ICE_PACKAGE_FORMAT オプションが選択されている場合は、コンテンツ・パッケージ・サマリーとともに追加部分（ログ）も添付される。
FTP （例：`ftp://user.com/usr/homedir`）	Syndication Services は、指定された FTP サイトにログインし、カレント・ディレクトリを `usr/homedir` に変更する。次に、ICE XML パッケージを、`<package-id>.xml` という名前の単一 XML ファイルとしてアップロードする。	Syndication Services は、指定された FTP サイトにログインし、カレント・ディレクトリを `usr/homedir` に変更する。次に、コンテンツ・パッケージの各アイテムを別々のファイルとしてアップロードし、アイテムの content-filename 属性を保持して必要なサブディレクトリを作成する。NON_ICE_PACKAGE_FORMAT オプションが選択されている場合は、コンテンツ・パッケージ・サマリーとともに `<package-id>.log` という名前の追加ファイルもアップロードされる。
SMTP （例：`mailto:user@user.com`）	Syndication Services は、ICE XML パッケージを含む単一部分とともに電子メール・メッセージを送信する。	Syndication Services は、コンテンツ・パッケージの各アイテムにメール添付した、電子メール・メッセージを送信する。NON_ICE_PACKAGE_FORMAT オプションが選択されている場合は、コンテンツ・パッケージ・サマリーとともに追加テキスト添付（ログ）も添付される。

注意： FTP プロパティ `oracle.syndicate.transport.ftp.act_connect_mode` は、アクティブまたはパッシブ FTP 接続モードを使用して実行される、FTP コンテンツ・プッシュを制御します。Syndication Services のこの拡張プロパティの詳細は、2-28 ページの「Syndication Services の拡張プロパティ」を参照してください。

- **Raw 形式のサポート**: コンテンツ更新の形式が、XML ペイロード（ICE XML パッケージ）、メール添付（SMTP）、または FTP URL が提供されている場合の指定された FTP サーバーへの単なるアップロードのうち、いずれかであるか。

  Syndication Services は、XML ペイロード（ICE XML パッケージ）の形式でコンテンツ更新を指定されたアドレスに送信します。パッケージ構造を理解し、その中からコンテンツを抽出する作業は、受信クライアントに任されます。提供されている `SyndicateClientServlet` サーブレットを構成して HTTP ベースのリスナーを作成する方法は、『Oracle Application Server Syndication Services API Reference』（Javadoc）という API ドキュメントを参照してください。HTTP ベースのリスナーは、プッシュ・コンテンツ・パッケージを受信すると、受信されたコンテンツを抽出し、ローカル・ファイル・システムにそれを格納します。提供されている Java API を使用すると、ユーザーは `SyndicateClientServlet` サーブレットを使用して受信されたコンテンツのカスタム処理を実行できます。

  Syndication Services には、更新済みのコンテンツを ICE XML パッケージに入れずに受信するオプションもあります。このオプションを設定すると、SMTP を使用している場合はユーザーが更新ファイルをメール添付の形式で受信し、FTP URL が提供されている場合は指定された FTP サーバーに更新済みのコンテンツがアップロードされます。

パッケージのインタフェースと属性

`SyndicatePackage` インタフェースは、一連のコンテンツ操作（削除および追加）を記述します。削除操作は、`ItemRemove` インタフェースを使用して指定されます。コンテンツの追加内容には、追加または更新する必要があり、`Item` インタフェースおよび `ItemRef` インタフェースを使用して指定されるコンテンツが含まれます。`ItemGroup` インタフェースを使用すると、シンジケータは `Item` インタフェースおよび `ItemGroup` インタフェースを同時に使用して指定したコンテンツを関連付けることができます。`Item` インタフェースおよび `ItemRef` インタフェースは、コンテンツを含める方法によって区別されます。`Item` インタフェースは、配信コンテンツにコンテンツを直接含める場合に使用します。`ItemRef` インタフェースは、実際のコンテンツに対する間接参照を分散する場合に使用します。

`SyndicatePackage` インタフェースは、古い状態および新規の状態を指定します。新規の状態に到達する前に、コンテンツ・パッケージ内に含まれるすべての操作を処理する必要があります。操作を正常に終了できない場合は、ユーザーを一貫性のない状態にしないため、コンテンツ・パッケージで指定した実行済の操作をすべて元に戻す必要があります。

`SyndicatePackage` インタフェースでは、属性を管理するために一連のアクセッサ・メソッドを公開します。`SyndicatePackage` インタフェースに最も関連性のある属性を、次に要約します。

- **activation**: ユーザーとシンジケータ間の特定のビジネス関係により定義されているとおりに、コンテンツ・パッケージに含まれているコンテンツが使用される可能性のある日付と時刻。この属性がコンテンツ・パッケージ内の特定のアイテムまたはグループで使用される場合、コンテンツ・パッケージ階層の該当部分では、そこに指定された値が一般的な仕様を上書きします。この属性が省略された場合、パッケージ内に含まれているコンテンツは、ユーザーが任意で配置できます。

- **atomic-use** : 配信されるコンテンツをすべて使用する必要がある、または配信されるコンテンツは使用できないという表示制約。
- **confirmation** : 配信されたコンテンツ・パッケージが実際に配信され、正常に処理されたという確認をシンジケータが必要とするかどうか。
- **editable** : ユーザーが情報を変更できるか、または配信されたとおりに使用する必要があるかどうか。
- **expiration** : パッケージ内に含まれているコンテンツをそれ以降は使用できなくなる日付と時刻。この属性がコンテンツ・パッケージ内の特定のアイテムまたはグループに対して使用された場合、コンテンツ・パッケージ階層の該当部分では、そこに指定された値が一般的な仕様を上書きします。
- **fullupdate** : コンテンツ・パッケージを処理する際に、サブスクリプションで以前に配信されたすべてのコンテンツを完全に更新する必要があるかどうか。
- **new-state** : コンテンツ・パッケージを正常に処理した後のサブスクリプションの状態。これは、ユーザーには不透明な値であるか、ユーザーが直接アクセスできない値です。以降の増分更新を受信するために、ユーザーはこの状態を次の `getPackage` 要求に指定できます。
- **old-state** : コンテンツ・パッケージを処理する前のサブスクリプションの状態を指定する文字列。
- **package-id** : サブスクリプションの範囲内のコンテンツ・パッケージ。これは、必要な場合にコンテンツ・パッケージを確認するときに使用される値です。
- **show-credit** : サブスクライバがコンテンツ・ソースの属性を明示する必要があるかどうか。
- **subscription-id** : コンテンツ・パッケージが属するサブスクリプション。

Item および ItemRef インタフェースの属性

`Item` および `ItemRef` インタフェースの属性には、配布されるコンテンツが明示的に含まれています。`Item` インタフェースの属性のデフォルト・コンテンツ・モデルは、文字データです。`base64` 値を使用すると、`Item` インタフェースの属性内でバイナリ・データを送信できます。`ItemRef` インタフェースの属性は、コンテンツへのポインタを管理します。コンテンツを参照すると、コンテンツ・パッケージ配信に含まれます。アイテム・コンテンツ・データのアクセス方法と処理方法は、『Oracle Application Server Syndication Services API Reference』（Javadoc）という API ドキュメントで、`SAXPackageHandler` インタフェースの詳細を参照してください。`Item` および `ItemRef` インタフェースに最も関連のある属性を、次に要約します。

- **activation** : パッケージに含まれるコンテンツを使用できる日付と時刻。この属性がコンテンツ・パッケージ内の特定のアイテムまたはグループに対して使用された場合、コンテンツ・パッケージ階層の該当部分では、そこに指定された値が一般的な仕様を上書きします。

- **expiration** : コンテンツの使用期限が失効する日付と時刻。この属性がコンテンツ・パッケージ内の特定のアイテムまたはグループに対して使用された場合、コンテンツ・パッケージ階層の該当部分では、そこに指定された値が一般的な仕様を上書きします。
- **content-filename** : シンジケータにより示されるアイテムの相対宛先。`content-filename` 属性に記述されているパスは、パス要素のデリミタとして前方スラッシュを使用して表示されます。サブスクリプション・ルートは、そのサブスクリプションに関してシンジケータから受信する、すべてのファイル・ベースのコンテンツをユーザーが配置する論理ディレクトリとして定義されます。`content-filename` 属性がスラッシュで始まる場合も、ユーザーはパスをサブスクリプション・ルート・ディレクトリの相対パスとして処理する必要があります。
- **content-transfer-encoding** : `PackageItem` 属性に含まれているか、この属性により参照されるデータのエンコード。有効な値は、`x-native-xml` および `base64` のみです。これらは、RFC-2045 の 6.2 および 6.3 項から導出された値です。`x-native-xml` 値は、XML 1.0 W3C 勧告の 2.4 項で特定された特殊文字を保護するため、文字エンティティが使用されている可能性があることを示します。`base64` 値は、RFC-2045 の 6.8 項に定義されているのと同じプロパティを示します。`base64` 値は、グラフィックス形式などのバイナリ・データや人間が判別できないその他のデータ形式を配布する手段を提供します。ICE の今後のバージョンでは、この一連の値が拡張される可能性があります。
- **content-type** : コンテンツのメディア・タイプ。RFC-2045 および RFC-2046（`charset` パラメータの設定を含む）を遵守する必要があります。デフォルト値の `application/octet-stream` は、配布されるコンテンツが不透明な特性を持つことを示します。
- **ip-status** : コンテンツの知的所有権のステータスを指定する文字列。
- **item-id** : コンテンツ・パッケージが含まれる範囲内で一意に識別されるアイテム。
- **rights-holder** : この要素内のコンテンツに対して元のシンジケーション権限を保持するエンティティ。
- **show-credit** : 要素のコンテンツにソースのクレジットを明記する必要があるかどうか。この属性の値は、`true`、`yes`、`false` または `no` に指定する必要があります。値 `yes` または `true` は、この要素のコンテンツを使用する際にソースのクレジット表示が必要であることを示します。値 `no` または `false` は、ソースのクレジット表示がオプションであることを意味します。デフォルト値は `no` です。
- **subscription-element** : サブスクリプションの継続期間を超えて、永続的に識別されるコンテンツ・アイテム。個々のコンテンツ・アイテムを明示的に更新または削除する唯一の方法は、後続のコンテンツ・パッケージ内で操作する際に、subscription-element 識別子を使用することです。それ以外の場合は、サブスクリプションに関連付けられているすべてのコンテンツを削除する、完全更新をする必要があります。
- **name** : コンテンツの論理識別子。

4

コンテンツ・コネクタの開発

コンテンツ・コネクタ（コンテンツ・プロバイダ・アダプタまたは CPAdaptor とも呼ばれる）は、Syndication Services が外部コンテンツ・リポジトリと通信するために使用するソフトウェア・コンポーネントです。開発者はコンテンツ・コネクタを作成して、リポジトリのリソースをサブスクリプション可能なオファーとして公開できます。

コンテンツ・コネクタは、Java プログラミング言語を使用して開発します。コンテンツ・コネクタを開発するには、`oracle.syndicate.server.cp` パッケージに `CPAdaptor` インタフェースを実装する JavaBean を作成する必要があります。次の各項では、サンプル・コンテンツ・コネクタのコード・シーケンスを説明します。このサンプルでは、入力として RSS（Really Simple Syndication）フィード（コンテンツ・ソース）を使用し、これをユーザーがサブスクライブできるリソースにします。

この例の完全なソース・ファイルを含むデモ・ファイルのセットは、Oracle Technology Network Japan（OTN-J）の Web サイト（`http://otn.oracle.co.jp/`）からダウンロードできます。

`RSSCPAdaptor.java` プログラムの一部はこの章で説明していますが、完全なリストは D-2 ページの「RSSCPAdaptor.java プログラムのリスト」を参照してください。コンテンツ・コネクタ開発用の Syndication Services API の使用方法を説明した API ドキュメントは、『Oracle Application Server Syndication Services API Reference』（Javadoc）として用意されています。

この章では、次の内容について説明します。

コンテンツ・コネクタ（CPAdaptor）の開発

Java プログラミングの観点からは、コンテンツ・コネクタは `oracle.syndicate.server.cp` パッケージに CPAdaptor インタフェースを実装する JavaBean クラスです。コンテンツ・コネクタの機能には次のものが含まれます。

- JavaBeans の *getter* および *setter* 規則を使用した、一連のプロパティの公開。通常、コネクタはその機能に必要なパラメータをプロパティとして公開します。たとえば、RDBMS システムへのインタフェースであるコネクタでは、コネクタ・プロパティとして URL、ユーザー名、パスワードなどのデータベース接続パラメータを公開する場合があります。コンテンツ・プロバイダの登録処理中に Syndication Services の管理ページで編集できるのは、Java プリミティブ型のプロパティのみです。
- Syndication Services 管理者がオファーの作成に使用できるリソース・リストの提供。
- コンテンツ・コネクタのリソースの 1 つに関連付けられているコンテンツ・パッケージの作成。コンテンツ・コネクタでは、コンテンツ・パッケージを増分として作成する（最終更新以降に変更されたアイテムのみを返す）か、または完全更新する（リソースの 1 つに関連付けられているすべてのアイテムを返す）か、あるいはその両方にするかを決定できます。

`oracle.syndicate.server.cp.CPAdaptor` インタフェースは、コンテンツ・コネクタにより実装される一連の API を定義します。コンテンツ・コネクタは、ステートレス Java クラスとして設計する必要があります。このクラスの唯一の状態情報はプロパティの値により決定され、その値はコンテンツ・プロバイダの登録中に管理者により決定されます。プロパティの値は管理者インタフェースを使用して設定または変更でき、操作中はコネクタによる変更が禁止されているため、状態はコネクタにより不変になります（変更できません）。つまり、`CPAdaptor` クラスは、マルチスレッド環境の場合は注意して設計する必要があります。たとえば、複数のコンテンツ・パッケージ作成要求がコンテンツ・プロバイダに同時に送信される可能性があります。

API はサンプルのコンテンツ・コネクタを使用して説明されています。そのサンプルでは、RSS フィードを読み取り、リソースを作成します。RSS とその仕様の詳細は、`http://backend.userland.com/rss` を参照してください。

サンプルのコネクタでは、RSS フィードに定義されている各アイテムに対してコンテンツ・パッケージ・アイテムを作成します。

次の各項では、コンテンツ・コネクタ開発プロセスの各段階を説明します。

- コンテンツ・プロバイダのクローズ
- コンテンツ・プロバイダのイベント・プッシュ・サポート

コンテンツ・コネクタのプロパティの公開

コンテンツ・コネクタは、その設定の一部をカスタマイズ可能プロパティとして公開できます。外部コンテンツ・リポジトリ・プロパティのクラス全般に対して汎用的なコンテンツ・コネクタを作成する場合は、その設定の一部を公開して、コンテンツ・プロバイダ・インスタンスをカスタマイズし、特定のリポジトリに接続できます。この点で、コンテンツ・コネクタは一種のコンテンツ管理システムのドライバであり、そのプロパティは特定のリポジトリ・インスタンスに対する接続パラメータであるとみなすことができます。

例 4-1 にあるサンプルのコンテンツ・コネクタでは、プロパティを 1 つのみ公開しています。このプロパティは、リソースとして公開される RSS フィードの URL を識別します。公開される一連のプロパティをフィルタ処理し、プロパティの表示名と説明を追加するために、JavaBeans の `BeanInfo` クラスを指定することもできます。

例 4-1 コンテンツ・コネクタのプロパティの公開

```
public class RSSCPAdaptor
  implements CPAdaptor, OSSExceptionConstants
{
  private static final String LAST_BUILD_DATE_FORMAT = "EEE, dd MMM yyyy H:mm:ssz";

  private String     _rssurl;
  private CPContext _cpctx;

  public RSSCPAdaptor()
  {}

  /**
   * Returns the URL of the RSS feed.
   */
  public String getRSSURL()
  {
    return _rssurl;
  }

  /**
   * Sets the URL of the RSS feed.
   */
  public void setRSSURL(String rssurl)
  {
    _rssurl = rssurl;
  }
```

コンテンツ・コネクタの初期化

コンテンツ・コネクタの初期化中に、Syndication Services は CPAdaptor インタフェースを実装するクラス（この例では、RSSCPAdaptor クラス）のインスタンスを作成し、コンテンツ・プロバイダの登録中に管理者により指定される値に従ってそのプロパティを設定します。

Bean 状態のインスタンス化および復元後、CPAdaptor.init メソッドが呼び出され、CPContext オブジェクトのインスタンスがコンテンツ・コネクタに提供されます。CPContext オブジェクトは、コンテンツ・コネクタの接続中に有用な一連の情報をコンテンツ・コネクタ・インスタンスに提供します（例 4-2 を参照）。これには、CPMessageFactory オブジェクトや、コンテンツ・コネクタが処理して返す必要のあるデータ構造を作成する場合に使用する CPAdaptor に関する情報などが含まれます。コンテンツ・コネクタは、提供された CPContext オブジェクトへの参照をフィールドの 1 つに格納すると想定されます。

ping 方式は、Syndication Services の管理コードで使用し、コンテンツ・コネクタが正しく構成されているかどうか、コネクタの操作の準備ができているかどうかを検証できます。たとえば、この RSS コンテンツ・コネクタでは、RSS URL がアクティブであり、RSS フィードを正しく返すことを検証できます。

例 4-2 コンテンツ・コネクタの初期化

```
/**
   * Receives and stores the CPContext object from which
   * you can access the CPMessageFactory object.
   */
  public void init(CPContext cpctx)
    throws CPException
  {
    _cpctx = cpctx;
  }

  public boolean ping()
    throws CPException
  {
    return true;
  }
```

リソースのリストの公開

コンテンツ・コネクタがシステムにインストールされ、このコンテンツ・コネクタを使用して 1 つ以上のコンテンツ・プロバイダが登録されると、Syndication Services 管理者はコンテンツ・プロバイダの 1 つ以上のリソースをオファーとして公開することを決定できます。これを実行するには、管理者は「オファーの作成」ウィザードの手順に従い、オファーとして公開するコンテンツ・プロバイダ・リソースを選択します。

コンテンツ・コネクタは、オファーの作成に使用できるリソースのリストを `CPAdaptor.buildCatalog` API を使用して公開します。コンテンツ・コネクタは、`CPMessageFactory` オブジェクトを使用して `CPOffer` オブジェクトのリストを作成し、オファー・メタデータを設定して、使用可能なリソースのリストに対するイテレータを返します。例 4-3 は、RSS フィードから `CPOffer` を作成する方法を示します。フィード・メタデータのいくつかは、リソースのプロパティに対するデフォルトを指定するために使用されます。これらのプロパティは、管理者がオファーの作成時に確認および編集できます。

例 4-3 RSS フィードからの CPOffer オブジェクト・リストの作成

```
public Iterator buildCatalog()
  throws CPException
{
  Document docrss = parseRSS();
  Element   elrss = docrss.getDocumentElement();
  NodeList  nlchs = elrss.getElementsByTagName("channel");

 // Loop over the channels defined in this RSS
 // and for each one create an offer.
  CPMessageFactory cpmf = _cpctx.getCPMessageFactory();
  ArrayList      alOffs = new ArrayList();
  for (int i=0; i<nlchs.getLength(); i++) {

    // For each channel, you will create
    // a CPOffer object. CPOffer objects are shown to
    // syndication administrators as content
    // provider resources from which you can build offers.
    Element  elCh = (Element) nlchs.item(i);
    CPOffer cpoff = cpmf.createCPOffer();

    // Each offer must have a unique ID for its content provider.
    // Set the mandatory ID attribute and any other
    // optional attributes.
    String title = getChildElementValue(elCh, "title");
    if ((title == null) || (title.trim().length() == 0)) {
      title = "unknown";
    }

    // Set the offer properties.
```

```
      cpoff.setCPOfferID(title);
      cpoff.setName(title);
      cpoff.setDescription(getChildElementValue(elCh,  "description"));
      cpoff.setRightsHolder(getChildElementValue(elCh, "copyright"));
      cpoff.setProductName("OracleAS Syndication Services RSS Feed Import");
      cpoff.setAtomicUse(false);
      cpoff.setEditable(true);
      cpoff.setShowCredit(false);
      cpoff.setUsageRequired(false);

      alOffs.add(cpoff);
    }

    return alOffs.iterator();
  }
```

返された各オファーには、setCPOfferID メソッドにより設定されたオファー ID が含まれている必要があります。

コンテンツ・パッケージの作成

ユーザー（サブスクライバ）がオファーにサブスクライブすると、コンテンツの更新が配信されます。コンテンツ更新は、コンテンツをリクエストするオファーに関連付けられているコネクタの buildPackage メソッドを呼び出すことで作成されます。buildPackage メソッドでは、次の 2 つのパラメータを使用します。

- CPPackageRequest: このメソッドは、このコンテンツ・パッケージ・リクエストに関連するパラメータのアクセッサを提供します。これらのパラメータ・アクセッサには、コンテンツ・パッケージのリクエスト先となるコンテンツ・プロバイダ・リソースの ID（CPOfferID。カタログの作成時にコンテンツ・コネクタが CPOffer.setCPOfferID メソッドを使用して設定した ID と同じ ID）、コンテンツ・パッケージの更新をリクエストするサブスクライバの ID、およびこのコンテンツ・パッケージ・リクエストに関連付けられているサブスクリプションの ID が含まれます。このメソッドには、このコンテンツ・パッケージ・リクエストに対してユーザーが指定したパラメータも含まれます（ある場合）。コンテンツ・コネクタは、この一連のパラメータを使用してリソースの更新を判断します。
- CPRequestContext: このメソッドは、コンテンツ・コネクタが、返された CPPackage オブジェクトを作成する目的で確保または割当て（あるいはその両方）る必要があるリソースを格納する場合に使用できます。このようなリソースは、コンテンツ・パッケージの送信後に、同じ CPRequestContext インスタンスがコンテンツ・コネクタに再指定される場合に、closePackage パッケージで解放可能です。

例 4-4 は、サンプルの RSSCPAdaptor クラスでコンテンツ・パッケージを作成する方法を示します。コンテンツ・コネクタは、構成に使用した URL に続く RSS フィードを読み取ります。次に、コンテンツ・コネクタは、フィード内で検出された各 RSS ニュース・アイテム

のそれぞれに CPItem オブジェクトを作成します。CPItem コンテンツは、ContentAccessors メソッドを介してアクセスできます。

CPPackage インスタンスと CPItem インスタンスは、CPContext オブジェクトを介してコンテンツ・コネクタに提供された CPMessageFactory オブジェクトを使用して作成する必要があります。ContentAccessors メソッドは、Java InputStream 上の単純なインタフェース・ラッパーで、アイテムのコンテンツを返します。CPMessageFactory オブジェクトには、最も一般的な ContentAccessors インタフェース用ファクトリ・メソッドがいくつか含まれています（この例で使用する StringContentAccessor メソッドも含まれています）。開発者は、必要に応じて独自の ContentAccessors インタフェースを実装するように作成できます。

例 4-4 コンテンツ・パッケージの作成

```
public CPPackage buildPackage(CPPackageRequest req,
                              CPRequestContext ctx)
  throws CPException
{
  // Parse the RSS document.
  Document docrss = parseRSS();
  Element   elrss = docrss.getDocumentElement();
  Element    elCh = getChannelElement(elrss, req.getOfferID());

  // Create a new content provider package and initialize
  // its properties by using RSS feed values.
  CPMessageFactory cpmf = _cpctx.getCPMessageFactory();
  CPPackage  pkg = cpmf.createCPPackage();

  // Check if the RSS feed has been modified since the
  // the last content package request.
  String oldState = req.getState();
  String newState = getChildElementValue(elCh, "lastBuildDate");

  Date dNewState = getDate(newState);
  Date dOldState = null;
  if (!oldState.equals(INITIAL_STATE)) {
    dOldState = getDate(oldState);
    if ((dOldState != null) &&
        (dNewState != null) &&
        dOldState.equals(dNewState)) {

      // The RSS feed has not been updated
      // since the last visit. Throw an exception
      // notifying that the content package sequence is
      // up-to-date and no update needs to be reported.
      throw new CPException(CP_PACKAGE_UP_TO_DATE);
    }
```

```
  // This is an incremental update over
  // the previous update that was sent.
  // Set the full update flag accordingly.
  pkg.setFullUpdate(false);
}
else {

  // This is a request for a full new update.
  // Set the full update flag accordingly.
  pkg.setFullUpdate(true);
}
pkg.setOldState(oldState);
pkg.setNewState(newState);

// Scan through the RSS news items building
// corresponding CPItem objects.
NodeList nlItems = elCh.getElementsByTagName("item");
for (int i=0; i<nlItems.getLength(); i++) {

  Element elItem = (Element) nlItems.item(i);
  CPItem cpi = cpmf.createCPItem();

  // Sets item file name and content type.
  // Each RSS news item will have a dummy file name
  // and an XML code excerpt for the associated news.
  String itemFilename = "item"+i+".xml";
  cpi.setContentFilename(itemFilename);
  cpi.setContentType("text/xml");

  // Sets the Item ID.
  String guid = getChildElementValue(elItem, "guid");
  if (guid != null) {
    cpi.setID(guid);
  }

  // Sets the Item name.
  String title = getChildElementValue(elItem, "title");
  if (title != null) {
    cpi.setName(title);
  }
  else {
    // The Item name is mandatory
    // so you need to set it
    // to a value.
    cpi.setName(itemFilename);
  }
```

```
    // Sets the Item Activation Date, if any.
    String pubDate = getChildElementValue(elItem, "pubDate");
    if (pubDate != null) {
      Date dPubDate = getDate(pubDate);
      cpi.setActivation( new ISODateTime(dPubDate));
    }

    // Sets the Item content to the RSS Item XML element.
    try {
      StringWriter sw = new StringWriter();
      ((XMLElement) elItem).print( new PrintWriter(sw));
      ContentAccessor ca = cpmf.createContentAccessor(sw.toString());
      cpi.setContent(ca);
    }
    catch(Exception e) {
      throw new CPException(CP_GENERIC_ERROR, e, e);
    }

    // Add the Item to the content package.
    pkg.addPackageItem(cpi);
  }

  return pkg;
}

public void closePackage(CPRequestContext ctx)
  throws CPException
{
}
```

次に示す CPPackage および CPPackageItem オブジェクトのプロパティは必須であり、返されたコンテンツ・パッケージを配信するには、開発者が設定する必要があります。

- CPPackage.oldstate および CPPackage.newstate: これらのプロパティは、コンテンツ・パッケージの配信前および配信後の、このリソースに関連付けられているサブスクリプションの状態を指定します。
- CPItem.ID および CPItem.name: ID プロパティは、コンテンツ・パッケージの有効範囲にあるアイテムを一意に識別します。名前付きプロパティは論理名です。開発者は、CPItem.subscriptionElement プロパティの設定についても考慮する必要があります。このプロパティは、コンテンツ・プロバイダ・リソースおよび関連するサブスクリプションの期間中にアイテムを一意に識別します。
- CPItem.contentFilename: コンテンツ・ファイル名のプロパティは、サブスクリプション・ルートを基準とするアイテムの相対的な宛先を指定します。つまり、サブスクリプション・ルートを基準とし、その中に論理ディレクトリとしてサブスクリプション・ルートが定義されています。論理ディレクトリ内には、サブスクリプションに対してシンジケータから受信したすべてのファイル・ベースのコンテンツがサブスクライバ

により配置されます。コンテンツ・ファイル名に記述されるパスは、パス要素のデリミタとして前方スラッシュを使用して表す必要があります。

- `CPItemRef.url`: URL プロパティは、エンド・ユーザーがコンテンツを取り出すために使用します。
- `CPItemGroup.ID`: ID プロパティは、コンテンツ・パッケージの有効範囲内にあるグループを一意に識別します。
- `CPItemRemove.subscriptionElement`: このプロパティは、コンテンツ・プロバイダ・リソースおよびその関連サブスクリプションの期間中にアイテムを一意に識別します。

注意： 削除されたアイテムの伝播 -- Syndication Services では、使用するコンテンツ・コネクタがアイテム削除通知をサポートする場合に、削除されたアイテムの伝播をサポートします。これは、コンテンツ・パッケージの作成操作中に `CPItemRemove` オブジェクトを作成して行います（4-6 ページの「コンテンツ・パッケージの作成」および API ドキュメントを参照）。

削除されたアイテムを検出できないコンテンツ・コネクタを使用してコンテンツ・プロバイダが登録された場合、コンテンツ・ソースでのアイテムの削除はサブスクライバには通知されることがなく、その結果、サブスクライバ・サイトにはアイテムが残ることになります。

たとえば、コンテンツ・プロバイダのコンテンツ・ソースとして Oracle Application Server Portal を使用する（WebDAV コンテンツ・コネクタを使用してポータル・ページに構成する）と、イメージ・アイテムの更新は旧アイテムの削除および新アイテムの作成とみなされます。削除されたアイテムがサブスクライバに伝播されないため、旧イメージと新イメージが両方ともサブスクリプション内に残ります。

回避策として、パッケージ作成時にアイテム削除通知をサポートするように（`CPItemRemove` クラスを使用したカスタム開発を介して）コンテンツ・コネクタを改善できます。

増分更新

コンテンツ・パッケージの新規更新のリクエストを受信したとき、コンテンツ・プロバイダには最後の配信以降に追加または更新された一連のアイテムのみを返すオプションがあります。増分更新を実行するために、コンテンツ・コネクタ API ではリソースの現在の状態を識別する状態トークンの設定ができます。ユーザーがコンテンツ更新をリクエストしたとき、リソースの現在の状態がコンテンツ・プロバイダに提供されます。コンテンツ・プロバイダはその状態を使用して、更新の配信処理をします。次に、コンテンツ・プロバイダは返されたコンテンツ・パッケージ内にある新しい状態識別子を配信できます。

たとえば、簡単な実装の場合、コンテンツ・プロバイダは状態識別子として更新の日付を格納するか決定できます。コンテンツ・パッケージ・リクエストを受信すると、コンテンツ・プロバイダは `CPPackageRequest.getState()` メソッドを使用してユーザーのサブスクリプションの現在の状態を取り出し、日付にキャストしてから、リソースをスキャンしてその日付以降に変更されているすべてのアイテムを検出します。次に、スキャン完了後にコンテンツ・パッケージを作成し、パッケージを返した後、コンテンツ・プロバイダは状態（`CPPackage.setNewState`）を現在の日付と時刻に設定します。

サンプルの `RSSCPAdaptor` クラスでは同様の方法を使用していますが、RSS フィードの最終変更日を状態識別子として参照しています。例 4-5 は、`buildPackage` コードでの状態管理ロジックを示します。また、特殊な `INITIAL_STATE` 状態識別子の処理方法と、コンテンツ・パッケージの `fullUpdate` フラグのプロパティ設定も示します。

例 4-5 増分更新の実行

```
// Check if the RSS feed has been modified since the
// the last content package request.
String oldState = req.getState();
String newState = getChildElementValue(elCh, "lastBuildDate");

Date dNewState = getDate(newState);
Date dOldState = null;
if (!oldState.equals(INITIAL_STATE)) {

  dOldState = getDate(oldState);
  if ((dOldState != null) &&
      (dNewState != null) &&
      dOldState.equals(dNewState)) {

    // The RSS feed has not been updated
    // since the last visit. Throw an exception
    // notifying that the package sequence is
    // up-to-date and no update needs to be reported.
    throw new CPException(CP_PACKAGE_UP_TO_DATE);
  }
  // This is an incremental update over
  // the previous update that was sent.
```

```
  // Set the full update flag accordingly.
  pkg.setFullUpdate(false);
}
else {

  // This is a request for a full new update.
  // Set the full update flag accordingly.
  pkg.setFullUpdate(true);
}
pkg.setOldState(oldState);
pkg.setNewState(newState);
```

コンテンツ・プロバイダのクローズ

コンテンツ・コネクタのインスタンスは Syndication Services によりキャッシュされます。コンテンツ・プロバイダのプロパティ値が管理者により変更された場合、コンテンツ・プロバイダのキャッシュ済インスタンスのコピーは Infrastructure データベースにより解放され、新しいコンテンツ・プロバイダ・インスタンスが作成されて、新しいプロパティ値を使用して構成されます。コンテンツ・プロバイダのインスタンスを解放するときは、`CPAdaptor.close` メソッドを呼び出します。この方法で開発者は、`init` メソッドを使用して取得したすべてのリソースを解放できます。

コンテンツ・プロバイダのイベント・プッシュ・サポート

コンテンツ・プロバイダから新規コンテンツが使用可能になると、イベントを起動して、コンテンツ・プロバイダのリソースに関連付けられているすべてのサブスクリプションに新規コンテンツを即座にプッシュできます。「概要」では、コンテンツ・プロバイダのイベント・プッシュ・サポートの概要を説明し、「イベントの構文」では、イベント構文のサポートを説明します。

概要

Syndication Services では、コンテンツ・プロバイダはコンテンツの新規バージョンが使用可能になるとイベントを起動できます。この機能は、2-13 ページの「契約および取引条件の管理」に説明されている時間ベースのコンテンツ更新を補完するものです。このようなイベントの起動条件は各コンテンツ・プロバイダに固有ですが、通常は、コンテンツ・プロバイダのリソースに関連付けられているコンテンツの新規バージョンが配信可能になったことを暗黙的に示します。このようなイベントが起動されると、Syndication Services では適格なサブスクリプションのすべてにコンテンツ更新をプッシュします。

次に、コンテンツ・プロバイダのイベント・ベースのプッシュの使用例を示します。

1. Syndication Services 管理者が、コンテンツ・プロバイダ cpA のリソース rsrcB に対してオファー offA を作成します。
2. 管理者は、プッシュ配信ルールを有効にする契約 conA を作成します。契約 conA は、プッシュ配信ルールに 24 時間の期間を定義し、更新回数として 0（ゼロ）を指定しま

す。この方法では、コンテンツ・プロバイダによりプッシュ・イベントが起動されるときは常に期間の違反は発生せず、したがって、コンテンツ・プロバイダからの任意のイベントによってサブスクライバにコンテンツがプッシュされる結果となります。

3. 管理者は契約 conA を使用してサブスクライバ sbbU にオファー offA を付与します。
4. サブスクライバ sbbU はオファー offA にサブスクライブし、新規サブスクリプション sbtE を取得します。
5. コンテンツ・プロバイダ cpA がリソース rsrcB に関してコンテンツ・プロバイダ・イベントを起動します。
6. Syndication Services は、コンテンツ・プロバイダ cpA のリソース rsrcB に関連付けられているすべてのサブスクリプション（新規サブスクリプション sbtE を含む）にコンテンツ更新をプッシュします。

イベントの構文

Oracle Application Server リリース 10*g*（9.0.4）以降の Portal and Wireless 中間層インストール・タイプでは、Syndication Services コンテンツ・プロバイダ・イベント・サーブレットがデフォルトでインストールされます。URL エンド・ポイントは次のとおりです。

```
http://<host>:<port>/syndserver/cpevent
```

Syndication Services が提供するインタフェースは、HTTP GET リクエストに基づきます。構文は次のとおりです。

```
http:// <host>:<port>/syndserver/cpevent?cp-id=XXX&cp-offer-id=YYY
```

XXX は cp-id 値で、YYY は cp-offer-id 値です。

コンテンツ・プロバイダは、コンテンツ・プロバイダ ID（cp-id）およびコンテンツ・プロバイダ・オファー ID（cp-offer-id）を提供することにより、すべてのサブスクリプションに対してプッシュ配信を起動できます。サブスクリプションは、指定のコンテンツ・プロバイダ ID cp-id を持つコンテンツ・プロバイダが指定する、コンテンツ・プロバイダ・オファー ID cp-offer-id を持つコンテンツ・プロバイダのオファーに基づいています。

たとえば、cp-id が 801 で cp-offer-id が fcpatest の場合、URL は次のようになります。

```
http://<host>:<port>/syndserver/cpevent?cp-id=801&cp-offer-id=fcpatest
```

例として、「概要」で説明している使用例では、次の HTTP GET リクエストにより、コンテンツ・プロバイダ cpA の rsrcB に関連付けられているすべてのサブスクリプション（サブスクリプション sbtE を含む）に対して、コンテンツ・プロバイダのイベント・プッシュを起動します。

```
http://syndication_services_host: syndication_services_
port/syndserver/cpevent?cp-id=cpA&cp-offer-id=rsrcB
```

あるいは、イベントを Syndication Services に送信して、サブスクリプション ID のリストに対してコンテンツ更新をプッシュすることもできます。構文は次のとおりです。

```
http:// <host>:<port>/syndserver/cpevent?sbt-ids=XX1,XX2,...
```

XX1 は最初のサブスクリプション ID、XX2 は 2 番目のサブスクリプション ID、などのように続きます。次に例を示します。

```
http://<host>:<port>/syndserver/cpevent?sbt-ids=C5E5702058205CC0E0340003BA1906A5,C56
09B440D656B40E0340003BA1906A5
```

コンテンツ・コネクタの管理

「新規コンテンツ・コネクタのインストール」では新規コンテンツ・コネクタをパッケージ化しインストールする方法を説明します。「インストール済のコンテンツ・コネクタのリスト表示」では現在インストールされているコンテンツ・コネクタをリストする方法を説明します。「コンテンツ・コネクタのアンインストール」ではコンテンツ・コネクタをアンインストールする方法を説明します。

新規コンテンツ・コネクタのインストール

コンテンツ・コネクタは、Java プログラミング言語を使用して開発します。コンテンツ・コネクタを開発するには、oracle.syndicate.server.cp パッケージに CPAdaptor インタフェースを実装する JavaBean を作成する必要があります。コーディングの完了後、コンテンツ・コネクタ・コードをコンパイルし、インストールを準備できます。コンテンツ・コネクタ・クラスをコンパイルするには、クラスパスに syndserver.jar ライブラリを必ず含めてください。syndserver.jar ライブラリには、コンテンツ・コネクタの開発に関連するすべてのインタフェースの定義が含まれています。たとえば、RSSCPAdaptor.java プログラムをコンパイルするには、例 4-6 に示されているコマンドを発行します。

例 4-6 RSSCPAdaptor.java ファイルのコンパイル

```
On UNIX
javac -classpath ${ORACLE_HOME}/lib/xmlparserv2.jar:${ORACLE_
HOME}/syndication/lib/syndserver.jar RSSCPAdaptor.java

On Windows
javac -classpath <ORACLE_HOME>¥lib¥xmlparserv2.jar:<ORACLE_
HOME>¥syndication¥lib¥syndserver.jar RSSCPAdaptor.java
```

コンテンツ・コネクタの開発者は、Syndication Services システムにコンテンツ・コネクタをインストールする前に、ユニット・テストをいくつか実行することをお薦めします。コンテンツ・コネクタのユニット・テストを実行する 1 つの方法に、buildCatalog および buildPackage API を呼び出す main メソッドをコンテンツ・コネクタ・クラスに追加するという方法があります。サンプルの RSSCPAdaptor.java プログラムは、その実行方法を示しています。

コンテンツ・コネクタ・コードをコンパイルした後は、コンパイル済クラスを含む JAR ファイルを作成します。例 4-7 は、例 4-6 でコンパイルした RSS コンテンツ・コネクタの JAR ファイルを作成するコマンドを示します。

例 4-7 rsscp.jar ファイルの作成

```
jar cvf rsscp.jar *.class
```

作成した rsscp.jar ファイルは、Syndication Services のランタイムおよび管理コンポーネントが必要時にライブラリを検出してロードすることが可能なディレクトリにコピーする必要があります。コンテンツ・コネクタのライブラリのデフォルトの位置は、UNIX システムの場合は *${ORACLE_HOME}/syndication/lib/cp*、Windows システムの場合は *<ORACLE_HOME>¥syndication¥lib¥cp* です。コネクタが依存する可能性のあるその他のライブラリが Syndication Services クラスパスにまだ含まれていない場合は、このディレクトリにコピーする必要があります。次の手順は、JAR ファイルのコピーです。UNIX または Windows 環境におけるコマンドは例 4-8 に示すとおりです。

例 4-8 Syndication Services ロード・ディレクトリへの JAR ファイルのコピー

```
On UNIX
cp rsscp.jar ${ORACLE_HOME}/syndication/lib/cp

On Windows
cp rsscp.jar <ORACLE_HOME>¥syndication¥lib¥cp
```

これで、新規コンテンツ・コネクタを Syndication Services にインストールできます。この手順を完了するには、例 4-9 に示すコマンドを実行します。

例 4-9 Syndication Services への新規コンテンツ・コネクタのインストール

```
On UNIX
java -DORACLE_HOME=${ORACLE_HOME} -jar ${ORACLE_HOME}/syndication/lib/syndserver.jar
-installConnector -name "RSSConnector" -description "Connector exposing RSS feeds as
syndication offers" -className RSSCPAdaptor

On Windows
java -DORACLE_HOME=<ORACLE_HOME> -jar <ORACLE_HOME>¥syndication¥lib¥syndserver.jar
-installConnector -name "RSSConnector" -description "Connector exposing RSS feeds as
syndication offers" -className RSSCPAdaptor
```

注意： Solaris オペレーティング・システムでは、例 4-9 に示す Java コマンドを実行する前に、環境変数 LD_LIBRARY_PATH が *<ORACLE_HOME>*/lib に設定されていることを確認してください。

Syndication Services のランタイム・コンポーネントの再起動（Syndication Services のアプリケーションがデプロイされている OC4J_Portal インスタンスの再起動）および管理コンポーネントの再起動（Oracle Enterprise Manager の再起動）は、これらのコンポーネントで新しいライブラリを検出するために必要です。コンテンツ・コネクタが正しくインストールされ、Syndication Services のランタイム・コンポーネントと管理コンポーネントの OC4J コンテナが再起動されると、新規コンテンツ・コネクタベースのコンテンツ・プロバイダを登録できます。コンテンツ・プロバイダの登録プロセスの詳細は、2-6 ページの「コンテンツ・プロバイダの登録および管理」を参照してください。

インストール済のコンテンツ・コネクタのリスト表示

syndserver.jar ライブラリには、システムに現在インストールされていおり、アンインストールの対象ともなるコンテンツ・コネクタのリストを表示するコマンドライン・オプションがいくつか提供されています。例 4-10 は、Syndication Services に現在インストールされているコンテンツ・コネクタのリストを返すコマンドを示します。

例 4-10 インストール済のコンテンツ・コネクタのリスト表示

```
On UNIX
java -DORACLE_HOME=${ORACLE_HOME} -jar ${ORACLE_HOME}/syndication/lib/syndserver.jar
-listConnectors

On Windows
java -DORACLE_HOME=<ORACLE_HOME> -jar <ORACLE_HOME>¥syndication¥lib¥syndserver.jar
-listConnectors

 Installed CPAs
----------------
ID   : 1
Name : FileConnector
Class: oracle.syndicate.server.cp.impl.file.FileCPAdaptor
Description: Content Provider Adaptor fetching content from a file system.

ID   : 2
Name : WebDAVConnector
Class: oracle.syndicate.server.cp.impl.dav.DAVCPAdaptor
Description: Content Provider Adaptor fetching content from repositories
 accessible through the WebDAV protocol.

ID   : 3
Name : RSS Connector
Class: RSSCPAdaptor
Description: Content Connector for RSS feeds
```

コンテンツ・コネクタのアンインストール

コンテンツ・コネクタをアンインストールするには、まず、コンテンツ・コネクタを使用するコンテンツ・プロバイダが現在登録されていないことを確認します。コンテンツ・コネクタを使用するコンテンツ・プロバイダがいくつか登録されている場合は、それぞれ削除する必要があります。この条件を満たした後に、例 4-11 に示すコマンドをコネクタ ID を指定して使用すると、コンテンツ・コネクタをアンインストールできます。

例 4-11 コネクタ ID が 3 のコンテンツ・コネクタのアンインストール

```
On UNIX
java -DORACLE_HOME=${ORACLE_HOME} -jar ${ORACLE_HOME}/syndication/lib/syndserver.jar
-uninstallConnector 3

On Windows
java -DORACLE_HOME=<ORACLE_HOME> -jar <ORACLE_HOME>¥syndication¥lib¥syndserver.jar
-uninstallConnector 3
```

A

エラー・メッセージ

この付録では、管理者や開発者用に、Syndication Services で使用されるエラー・メッセージのリストについて説明します。この付録は、サーバーのエラー・メッセージ、クライアントのエラー・メッセージ、プロバイダのエラー・メッセージおよび ICE のエラー・メッセージという 4 つの項で構成されています。

ランタイム・エラーは、`application.log` ファイルに記録されます。

UNIX では、このログ・ファイルは *`$ORACLE_HOME/j2ee/OC4J_Portal/application-deployments/syndserver/$ISLAND_NUMBER`* ディレクトリにあります。*`$ORACLE_HOME`* は、Oracle Application Server Portal および Oracle Application Server Wireless がインストールされている場所です。

Windows では、このログ・ファイルは *`<ORACLE_HOME>¥j2ee¥OC4J_Portal¥application-deployments¥syndserver¥<ISLAND_NUMBER>`* ディレクトリにあります。*`<ORACLE_HOME>`* は、Windows 上で Oracle Application Server Portal および Oracle Application Server Wireless がインストールされている場所です。

`$ISLAND_NUMBER` または *`<ISLAND_NUMBER>`* の値は、たとえば `OC4J_Portal_default_island_1` のようになります。

Oracle Enterprise Manager Oracle Application Server Syndication Services Administrator を使用中に検出されたエラーは、GUI 内のメッセージとして管理者に返されます。

サーバーのエラー・メッセージおよび説明

次の各項では、サーバーのエラー・メッセージについて説明します。

一般的なエラー・メッセージ

エラー・コード : OSS-00120

JAXP API と XML パーサーが正しく構成されていません

原因 : XML パーサーの実行中に XML パーサーにより構成エラーが報告されました。

処置 : `xmlparserv2.jar` ライブラリが Oracle Application Server のクラスパスにあり、クラスが OC4J で使用できることを確認してください。

無効なデータ・ソース init モードのエラー・メッセージ

エラー・コード : OSS-01002

プロトコル *{0}* はサポートされていません

原因 : サポートされていないプロトコルが着信リクエストに含まれています。

処置 : クライアント側のメッセージ・ライブラリをチェックし、Syndication Services がサポートするものに準拠していることを確認してください。

データベースのエラー・メッセージ

エラー・コード : OSS-01100

内部例外が発生しました *{0}*

原因 : 例外テキストに表示されている内部例外が発生しました。

処置 : `application.log` ファイルをチェックしてください。

UNIX では、このログ・ファイルは *$ORACLE_HOME/j2ee/OC4J_Portal/application-deployments/syndserver/$ISLAND_NUMBER* ディレクトリにあります。*$ORACLE_HOME* は、Oracle Application Server Portal および Oracle Application Server Wireless がインストールされている場所です。

Windows では、このログ・ファイルは *<ORACLE_HOME>¥j2ee¥OC4J_Portal¥application-deployments¥syndserver¥<ISLAND_NUMBER>* ディレクトリにあります。*<ORACLE_HOME>* は、Windows 上で Oracle Application Server Portal および Oracle Application Server Wireless がインストールされている場所です。

$ISLAND_NUMBER または *<ISLAND_NUMBER>* の値は、たとえば `OC4J_Portal_default_island_1` のようになります。

問題が解決されない場合は、オラクル社カスタマ・サポート・センターにお問い合せください。

エラー・コード : OSS-01102

無効な ID フォーマット *{0}* です

原因 : 内部エラーです。指定された ID が無効です。

処置 : ID には有効な整数を指定する必要があります。

ネゴシエーションのエラー・メッセージ

エラー・コード : OSS-01203

シンジケータがカウンタ・オファーを勧告しています

原因 : サブスクリプションに対して指定されたオファーが正しくないため、シンジケータがサブスクライバに対してカウンタ・オファーを勧告しています。

処置 : サブスクリプションを続行するには、カウンタ・オファーを検討して応答してください。

データベース接続マネージャのエラー・メッセージ

エラー・コード : OSS-02100

初期化されたデータソースがありません

原因 : 初期化されたデータソースがありません。

処置 : Infrastructure データベースをチェックし、必要な場合は現在のインスタンスを再起動してください。

エラー・コード : OSS-02101

データベース接続のオープン中にエラーが発生しました

原因 : データベース接続のオープン中にエラーが発生しました。

処置 : Infrastructure データベース・ステータスをチェックし、必要な場合はリスナーまたはデータベース・インスタンス（あるいはその両方）を再起動してください。

エラー・コード : OSS-02103

JNDI ルックアップ *{0}* の実行中にエラーが発生しました

原因 : リストされているエントリの JNDI ルックアップの実行中にエラーが発生しました。

処置 : OC4J インスタンスの `config` ディレクトリにある `data-sources.xml` ファイルをチェックし、UNIX の場合は `jdbc/OracleOSS` エントリ、Windows の場合は `jdbc¥OracleOSS` エントリが存在し、その中に正しいデータベース接続情報が含まれていることを確認してください。

エラー・コード : OSS-02104

OID 接続のオープン中にエラーが発生しました

原因 : OID 接続のオープン中にエラーが発生しました。

処置 : Infrastructure インスタンスの Oracle Internet Directory をチェックし、必要な場合は再起動してください。

エラー・コード : OSS-02105

ユーザー / グループ検索ベースがありません

原因 : ユーザーまたはグループの検索ベースがありません。

処置 : Infrastructure の設定中に Oracle Internet Directory が正しくインストールおよび構成されていることを確認してください。

エラー・コード : OSS-02106

Syndication の中間層（バージョン *{0}*）は Infrastructure（バージョン *{1}*）と互換性がありません。Infrastructure を更新してください。

原因 : シンジケーションの中間層が Infrastructure と互換性がありません。

処置 : Infrastructure のインストールを最新リリースに更新してください。

コンテンツ・コネクタのエラー・メッセージ

エラー・コード : OSS-03100

CPAdaptor クラスの確認中にエラーが発生しました

原因 : CPAdaptor の実装の検査に失敗しました。

処置 : 配置されているコンテンツ・コネクタ（CPAdaptor）の実装をチェックし、実装クラスまたはライブラリがクラスパスにあることを確認してください。

エラー・コード : OSS-03101

CPAdaptor クラスが見つかりませんでした

原因 : 対応する CPAdaptor クラスが見つかりませんでした。

処置 : 配置されているコンテンツ・コネクタ（CPAdaptor）のライブラリが、UNIX システムの場合は `$ORACLE_HOME/syndication/lib/cp` ディレクトリ、Windows システムの場合は `ORACLE_HOME¥syndication¥lib¥cp` ディレクトリにあることを確認し、対応する Oracle Application Server インスタンスを再起動してください。

エラー・コード : OSS-03102

CPAdaptor クラスをインスタンス化できません

原因 : CPAdaptor クラスをインスタンス化できません。

処置 : コンテンツ・コネクタ（CPAdaptor）の実装をチェックし、必要に応じて Syndication Services のドキュメントを参照してください。

エラー・コード : OSS-03103

CPAdaptor クラスにアクセスできません

原因 : CPAdaptor クラスにアクセスできません。

処置 : コンテンツ・コネクタ（CPAdaptor）の実装をチェックし、必要に応じて Syndication Services のドキュメントを参照してください。

エラー・コード : OSS-03104

CPA はターゲットを起動できません

原因 : コンテンツ・プロバイダの現在のアダプタ・インスタンスがターゲットを起動できません。

処置 : コンテンツ・コネクタ（CPAdaptor）の実装をチェックし、必要に応じて Syndication Services のドキュメントを参照してください。

エラー・コード : OSS-03105

セキュリティ例外

原因 : セキュリティ例外です。

処置 : コンテンツ・コネクタ（CPAdaptor）の実装をチェックし、必要に応じて Syndication Services のドキュメントを参照してください。

エラー・コード : OSS-03106

無効な引数

原因 : 無効な引数です。

処置 : コンテンツ・コネクタ（CPAdaptor）の実装をチェックし、必要に応じて Syndication Services のドキュメントを参照してください。

エラー・コード : OSS-03107

不正なプロパティ書式

原因 : プロパティの書式が不正です。

処置 : コンテンツ・コネクタ（CPAdaptor）の構成をチェックし、コネクタのプロパティの値が正しいプロパティ・タイプに対応していることを確認してください。

エラー・コード : OSS-03110

プロパティ *{0}* の値がありません

原因 : メッセージにリストされているコンテンツ・コネクタ（CPAdaptor）のプロパティの値がありません。

処置 : コンテンツ・コネクタ（CPAdaptor）の構成を調べ、問題のプロパティの値を追加してください。

オファーのエラー・メッセージ

エラー・コード : OSS-04100

オファー *{0}* はアクティブではありません

原因 : 指定されたオファーがアクティブではありません。

処置 : 特定のオファーに関する詳細は、Syndication Services の管理者にお問い合せください。

エラー・コード : OSS-04101

オファー *{0}* はサブスクライバ *{1}* に権限付与されていません

原因 : 指定されたオファーの権限がユーザーに付与されていません。

処置 : ユーザーは、権限が付与されていないオファーにサブスクライブしようとしています。オファーのアクセス設定を調べ、必要な場合は変更してください。

エラー・コード : OSS-04102

{0} は正しくありません。*{3}* の受信中に、フィールド *{1}* の値は *{2}* が予想されました

原因 : ユーザーがオファーにサブスクライブしようとしていますが、オファーではユーザーに付与されているものとは異なるサブスクリプションを要求しています。

処置 : ユーザーは、オファーのサブスクリプションを再発行する前に、オファーを調べて変更する必要があります。ユーザーがサブスクリプション・リクエストでカスタマイズできるオファーのフィールドは、push_url または remove_ice_element のみです。

サブスクリプションのエラー・メッセージ

エラー・コード : OSS-05101

認可されていないサブスクライバ *{0}*

原因 : サブスクライバが認可されていません。無効な ID のユーザーがログインしようとしました。

処置 : サブスクライバの ID をチェックし、正しいサブスクライバ ID がサーバーに送信されていることを確認します。

エラー・コード : OSS-05103

サブスクリプション *{0}* を認識できません

原因 : リストされている ID を持つサブスクリプションを認識できません。存在しないサブスクリプションまたは状態がアクティブでないサブスクリプションに対してユーザーが操作を実行しようとしました。

処置 : サブスクリプション ID が正しいことを確認し、該当する場合は、サブスクリプションの状態をアクティブに変更して、ユーザーがサブスクリプション操作を実行できるようにしてください。

エラー・コード : OSS-05104

サブスクリプション *{0}* はアクティブではありません

原因 : リストされている ID のサブスクリプションがアクティブではありません。存在しないまたは状態がアクティブでないサブスクリプションに対してユーザーが操作を実行しようとしました。

処置 : サブスクリプション ID が正しいことを確認し、該当する場合は、サブスクリプションの状態をアクティブに変更して、ユーザーがサブスクリプション操作を実行できるようにしてください。

エラー・コード : OSS-05105

サブスクライバ *{1}* には自身のサブスクリプション *{0}* がありません

原因 : 表示されている ID を持つサブスクライバには、指定されたサブスクリプションがありません。

処置 : サブスクライバ ID とサブスクリプション ID の両方を確認してください。

エラー・コード : OSS-05106

サブスクリプション *{0}* には配布ルールがありません

原因 : 表示されている ID のサブスクリプションには配信ルールがありません。

処置 : サブスクリプション ID を確認してください。また、サブスクライブされているオファーも確認し、その契約に有効な配信ルールが含まれていることを確認してください。

エラー・コード : OSS-05107

{1} にサブスクリプション *{0}* へのアクセスが許可されている配布ルールはありません

原因 : コンテンツ配信がリクエストされましたが、サブスクリプションに関連付けられている配信ルールで、現時点でアクセスが許可されている配信ルールはありません。

処置 : サブスクリプションに関連付けられている配信ルールを確認してください。ユーザーは、一定時間内に配信リクエストを再試行する必要があります。

エラー・コード : OSS-05108

URL *{0}* は無効です

原因 : シンジケータは新規コンテンツ・パッケージをユーザーにプッシュしようとしていますが、ユーザーのプッシュ URL はサポートされていません。

処置 : サブスクリプションのプッシュ配信ルールに関してユーザーのプッシュ URL を調べ、必要な場合はプッシュ URL を変更するようにユーザーに通知してください。

エラー・コード : OSS-05109

サブスクリプション *{0}* には配布ルール *{0}* がありません

原因 : サブスクリプションには、ユーザー・リクエストに一致する配信ルールがありません。

処置 : サブスクリプションに関連付けられている配信ルールを確認してください。ユーザーは、付与されている配信ルールに従って配信リクエストを再試行する必要があります。

エラー・コード : OSS-05110

サブスクリプション *{0}* は期限切れです

原因 : 有効期限ポリシーの基準が満たされたため、サブスクリプションの状態が自動的に期限切れに変更されています。

処置 : サブスクリプションの状態を調べ、期限切れになっていることを確認してください。

エラー・コード : OSS-05111

サブスクリプション *{0}* には未承認の認証があります

原因 : 未確認の認証を持つサブスクリプションに対して、ユーザーがコンテンツ配信をリクエストしています。

処置 : ユーザーは、次のコンテンツ配信をリクエストする前に、それまでに受信したインバウンド・コンテンツ・パッケージを確認する必要があります。

エラー・コード : OSS-05112

サブスクライバ *{1}* に対する *{0}* を確認するためのパッケージが見つかりません

原因 : ユーザーは、以前に配信されたコンテンツ・パッケージが認識されないことを確認しようとしています。

処置 : サブスクリプション・ステータスとユーザーに配信された最後のコンテンツ・パッケージの ID を確認してください。ユーザーに ID を通知し、ユーザーが配信を確認できるようにしてください。

スケジューラのエラー・メッセージ

エラー・コード : OSS-07001

無効なスケジューラ・ルール : 開始日 *{0}* が停止日 *{1}* より後に設定されています

原因 : 内部例外です。プッシュ・スケジューラ・ルールの開始日が停止日より後になっています。

処置 : サブスクライブされたオファーに関連付けられている契約をチェックし、開始日と停止日に関するプッシュ配信ルールをチェックしてください。

エラー・コード : OSS-07002

スケジュールが初期化されていません : *NULL* データベース・プロバイダ

原因 : Infrastructure データベースに接続できないため、Oracle Application Server Syndication Services のスケジューラ・コンポーネントを初期化できません。

処置 : Infrastructure データベース・ステータスをチェックし、必要な場合はリスナーまたはデータベース・インスタンス（あるいはその両方）を再起動してください。

エラー・コード : OSS-07003

ID *{0}* のスケジューラは停止しています

原因 : Oracle Application Server Syndication Services のスケジューラ・コンポーネントに関係する操作が必要ですが、スケジューラ・ステータスが停止になっています。

処置 : Oracle Application Server Syndication Services のシステム・プロパティを調べ、必要に応じてスケジューラ・コンポーネントをオンにしてください。Syndication Services アプリケーションがデプロイされている OC4J_Portal インスタンスの再起動が必要です。

エラー・コード : OSS-07004

無効なスケジューラ・ルール : 更新の数がゼロ以下です

原因 : 期間当たりの更新回数がゼロ未満のプッシュ配信ルールは、スケジュールできません。

処置 :「契約の編集」ページで期間当たりの更新回数を調べ、正の数値に変更してください。

サーバー・メッセージ関連のエラー・メッセージ

エラー・コード : OSS-08781

OSS ペイロード検証の失敗 : メッセージ *{0}* の属性 *{1}* の値 *{2}* が無効です

原因 : ユーザーは無効なメッセージ・ペイロードでリクエストを発行しました。メッセージには無効な属性が含まれています。

処置 : ユーザーがアプリケーションの開発に Oracle Application Server Syndication Services クライアント・ライブラリを使用していることを確認してください。

エラー・コード : OSS-08782

OSS ペイロード検証の失敗 : 要素 *{0}* のかわりに予期せぬ要素 *{1}* が見つかりました

原因 : ユーザーは無効なメッセージ・ペイロードでリクエストを発行しました。メッセージには予期せぬ要素が含まれています。

処置 : ユーザーがアプリケーションの開発に Oracle Application Server Syndication Services クライアント・ライブラリを使用していることを確認してください。

エラー・コード : OSS-08783

OSS ペイロード検証の失敗 : 必要なメンバー *{0}* がありません

原因 : ユーザーは無効なメッセージ・ペイロードでリクエストを発行しました。メッセージには想定されている要素が含まれていません。

処置 : ユーザーがアプリケーションの開発に Oracle Application Server Syndication Services クライアント・ライブラリを使用していることを確認してください。

SQL 例外エラー・メッセージ

エラー・コード : OSS-20001

操作の完了を中断する従属オブジェクト *{0}* があります

原因 : Syndication Services のメタデータ・リポジトリにはメッセージ内に従属オブジェクトが含まれており、リポジトリ・エンティティの 1 つを削除できません。

処置 : 最初に従属オブジェクトを削除した後、現在のオブジェクトを削除してください。

エラー・コード : OSS-20002

{0} 従属オブジェクトがありません。操作を完了できません

原因 : 想定されている従属オブジェクトが Syndication Services のメタデータ・リポジトリにありません。現在のオブジェクトの外部参照がすでに存在しないため、バックエンドの記憶域が一貫性のない状態になっています。

処置 : Syndication Services のメタデータ・リポジトリに現在存在するエンティティを確認し、問題のエンティティを削除して、システムを一貫性のある状態にしてください。

エラー・コード : OSS-20003

{0} オブジェクトが見つかりません

原因 : オブジェクトが Syndication Services のメタデータ・リポジトリに見つかりませんでした。

処置 : サーバー内部エラーです。オブジェクト ID を確認してください。

エラー・コード : OSS-20004

重複しているため *{0}* を作成できません

原因 : リストにある指定のエンティティは重複しているため、Syndication Services のメタデータ・リポジトリに作成できません。

処置 : 新規エンティティを作成するときは、重複を避けてください。

エラー・コード : OSS-20005

{0} と *{1}* の関連付けが存在しません

原因 : サーバー内部エラーです。指定された 2 つのエンティティに関連付けは存在しません。

処置 : 2 つのエンティティを確認し、問題を解決できるかどうか検討してください。

エラー・コード : OSS-20007

{0} と *{1}* の関連付けはすでに存在します

原因 : 指定された 2 つのエンティティには関連付けがすでに存在しています。

処置 : リクエストされた関連付けはすでに実行されているため、処置は不要です。

エラー・コード : OSS-20008

オブジェクト *{0}* はアクティブではありません

原因 : 指定されたオブジェクトはアクティブではありません。

処置 : オブジェクトの状態を調べ、該当する場合はアクティブに変更してください。

エラー・コード : OSS-20009

{1} 状態のオブジェクト *{0}* は期限切れです

原因 : 指定されたオブジェクトの状態は期限切れに変更されています。

処置 : サブスクリプションの状態を調べ、期限切れになっていることを確認してください。

エラー・コード : OSS-20012

オファー *{0}* は *{1}* に権限付与されていません

原因 : 指定されたユーザーまたはグループにはオファーの権限が付与されていません。

処置 : オファーのアクセス情報を調べ、該当する場合は指定されたユーザーにアクセス権を付与してください。

クライアントのエラー・メッセージおよび説明

クライアントのエラー・メッセージは、次のとおりです。

一般的なエラー・メッセージ

エラー・コード : OSS-00120

JAXP API と XML パーサーが正しく構成されていません

原因 : XML パーサーの実行中にパーサーにより構成エラーが報告されました。

処置 : `xmlparserv2.jar` ライブラリが Oracle Application Server のクラスパスにあり、クラスが OC4J で使用できることを確認してください。

エラー・コード : OSS-00121

予期しないシンジケーション・クライアント内部例外が検出されました

原因 : シンジケーション・クライアントにより、予期しない例外が検出されました。

処置 : クライアント実装に関連付けられているログ情報を調べ、追加エラー情報を取得してください。

XML クライアント状態ハンドラのエラー・メッセージ

エラー・コード : OSS-00150

ファイル *<filename>* はディレクトリであるため、クライアント状態の保存には使用できません

原因 : 指定されたファイル名はディレクトリであるため、クライアント状態の保存には使用できません。

処置 : クライアント状態の保存用として、既存のディレクトリと競合しない有効なファイル名を指定してください。

エラー・コード : OSS-00151

ファイル *<filename>* は読み取ることができないため、クライアント状態の保存には使用できません

原因 : 指定されたファイルは読み取れないため、クライアント状態の保存には使用できません。

処置 : クライアント状態用に指定されたファイルが読取り可能かどうかを調べてください。必要な場合は、ファイルに読取り権限を付与してください。

エラー・コード : OSS-00152

クライアント状態を保存するためのファイル *<filename>* を作成できません

原因 : クライアント状態を保存するために指定されたファイルを作成できません。

処置 : 指定されたディレクトリにファイルを作成できるかどうかを調べてください。必要な場合は、ディレクトリに書込み権限を付与してください。

エラー・コード : OSS-00153

クライアント状態ファイル *<filename>* の解析中にエラーが発生しました

原因 : 指定されたファイルは有効なクライアント状態ファイルでないか、破損している可能性があります。

処置 : クライアント状態保存ファイルが存在し、破損していないか確認し、ファイルの読取りに使用する状態ハンドラに準拠していることを確認してください。

エラー・コード : OSS-00154

ファイル *<filename>* にクライアント状態を書き込むことができません

原因 : 指定されたファイルにクライアント状態情報を書き込もうとしてアプリケーションでエラーが発生しました。

処置 : クライアント状態保存ファイルに書込み権限が付与されていることを確認してください。

エラー・コード : OSS-00155

重複しているため、ID *<subscriberid>* の新規サブスクライバを追加できません

原因 : クライアント状態保存ファイルにサブスクライバ情報を追加しようとしてアプリケーションでエラーが発生しました。ファイルには、指定された ID を持つサブスクライバの情報がすでに含まれています。同一 ID を持つ新規サブスクライバを追加することはできません。

処置 : 新規サブスクライバに別のサブスクライバ ID を使用してください。

エラー・コード : OSS-00156

重複しているため、ID *{0}* の新規サブスクリプションを追加できません

原因 : クライアント状態保存ファイルにサブスクリプション情報を追加しようとしてアプリケーションでエラーが発生しました。ファイルには、指定された ID を持つサブスクリプションの情報がすでに含まれています。同一 ID を持つ新規サブスクリプションを追加することはできません。

処置 : 新規サブスクリプションに別のサブスクリプション ID を使用してください。

FileSAXPackageHandler のエラー・メッセージ

エラー・コード : OSS-00160

ディレクトリ *{0}* を初期化できません

原因 : 指定されたディレクトリを着信コンテンツ・パッケージの保存ルート宛先として使用しようとしてアプリケーションでエラーが発生しました。

処置 : 指定されたディレクトリが存在し、クライアント・アプリケーションからアクセス可能で、書込み権限があることを確認してください。

エラー・コード : OSS-00161

アイテム *{0}* のファイルを作成できません

原因 : アプリケーションは、コンテンツ・パッケージを使用して配信された指定のアイテムを保存するファイルを作成できませんでした。

処置 : 宛先ディレクトリに適切な書込み権限があり、作成するファイルの名前が既存のディレクトリと競合していないことを確認してください。

エラー・コード : OSS-00162

アイテム *{0}* のフラッシュ・ファイルを作成できません

原因 : アプリケーションは、指定されたアイテムと一緒に配信されたコンテンツを宛先ファイルにフラッシュできませんでした。

処置 : 指定されたアイテムのコンテンツを保持するために十分な領域が宛先ファイル・システムにあり、宛先ディレクトリに書込み権限があることを確認してください。

エラー・コード : OSS-00163

アイテム *{0}* のファイルを閉じることができません

原因 : 指定されたアイテムのファイル出力ストリームを閉じようとしてエラーが発生しました。

処置 : 指定されたアイテムの宛先ファイルが他のアプリケーションで使用されていないことを確認してください。

SyndicateClientServlet のエラー・メッセージ

エラー・コード : OSS-00170

プロパティ *{0}* の値がありません

原因 : クライアント・サーブレットは、初期化するときに、`web.xml` 構成ファイルの指定されたプロパティを検出できませんでした。

処置 : 指定されたプロパティがアプリケーションの `web.xml` ファイルに定義されていることを確認してください。

エラー・コード : OSS-00171

クラス *{0}* をロードできません

原因 : SyndDelegtor 実装用の `web.xml` ファイルに指定されているクラスを実行時にロードできません。

処置 : 指定されたクラスが OC4J クラス・ローダーを使用して使用可能であることを確認してください。

DAVSAXPackageHandler のエラー・メッセージ

エラー・コード : OSS-00180

無効な DAV URL *{0}* です

原因 : 指定された URL に接続し、URL を使用して WebDAV 接続を確立しようとしてクライアント・アプリケーションでエラーが発生しました。

処置 : 指定された URL が有効な WebDAV エンド・ポイントであり、DAV サーバーが稼働していることを確認してください。

エラー・コード : OSS-00181

コンテンツ・パッケージの配信にリソース *{0}* を使用できません

原因 : 指定された DAV リソースを着信コンテンツ・パッケージの保存ルート宛先として使用しようとして、アプリケーションでエラーが発生しました。

処置 : 指定された DAV コレクションが存在し、クライアント・アプリケーションからアクセス可能で、接続ユーザーに書込み権限があることを確認してください。

エラー・コード : OSS-00182

アイテム *{0}* のリソースを作成できません

原因 : アプリケーションは、コンテンツ・パッケージを使用して配信された指定アイテムを格納する DAV リソースを作成できませんでした。

処置 : 宛先コレクションに対する書込み権限が接続ユーザーに付与されており、作成するリソースの名前が既存の DAV コレクションと競合していないことを確認してください。

エラー・コード : OSS-00183

アイテム *{0}* のリソースをフラッシュできません

原因 : アプリケーションは、指定されたアイテムと一緒に配信されたコンテンツを宛先 DAV リソースにフラッシュできませんでした。記憶領域が不十分か、宛先コレクションに対する権限に問題がある可能性があります。

処置 : 指定されたアイテムのコンテンツを保持するために十分な領域が宛先 DAV システムにあることを確認してください。

エラー・コード : OSS-00184

アイテム *{0}* のリソースを閉じることができません

原因 : リソース出力ストリームを閉じて、指定されたアイテムと一緒に配信された新規コンテンツを DAV リポジトリにコミットしようとしてエラーが発生しました。

処置 : 指定されたアイテムの宛先リソースが他のアプリケーションで使用されていないことを確認してください。また、DAV リポジトリに接続されているユーザーに、リソースの書込みおよび更新のための適切な権限があることも確認してください。

エラー・コード : OSS-00185

アイテム *{0}* のリソースを削除できません

原因 : DAV リポジトリのリソースを削除しようとしてエラーが発生しました。

処置 : 指定されたアイテムの宛先リソースが他のアプリケーションで使用されていないことを確認してください。また、DAV リポジトリに接続されているユーザーに、リソースを削除する適切な権限があることも確認してください。

エラー・コード : OSS-00186

DAV エンドポイントに接続できません

原因 : DAV リポジトリに HTTP 接続を確立しようとしてエラーが発生しました。

処置 : DAV エンド・ポイントが作動しており、接続に使用したユーザー名とパスワードが有効で、認証システム（SSO/OID）が稼働していることを確認してください。

シンジケーション接続の実装のエラー・メッセージ

エラー・コード : OSS-00196

状態ハンドラが使用できません

原因 : クライアント・アプリケーションが、構成済の状態ハンドラを使用できません。

処置 : 状態ハンドラの設定をチェックしてください。

エラー・コード : OSS-00197

サブスクライバを削除できません

原因 : 削除する必要のある従属オブジェクトがサブスクライバにあるか、エラーが発生したため、サブスクライバを削除できませんでした。

処置 : 状態ハンドラ・システムが正常に作動しており、従属オブジェクトがすべて削除されていることを確認してから、再試行してください。

プロバイダのエラー・メッセージおよび説明

プロバイダのエラー・メッセージは次のとおりです。

一般的なエラー・メッセージ

エラー・コード : OSS-00120

JAXP API と XML パーサーが正しく構成されていません

原因 : XML パーサーの実行中に XML パーサーにより構成エラーがレポートされました。

処置 : `xmlparserv2.jar` ライブラリが Oracle Application Server のクラスパスにあることと、クラスが OC4J で使用できることを確認してください。

SyndicateClientServlet のエラー・メッセージ

エラー・コード : OSS-00170

プロパティ *{0}* の値がありません

原因 : クライアント・サーブレットは、初期化するときに、`web.xml` 構成ファイルの指定されたプロパティを検出できませんでした。

処置 : 指定されたプロパティがアプリケーションの `web.xml` ファイルに定義されていることを確認してください。

エラー・コード : OSS-00171

クラス *{0}* をロードできません

原因 : SyndDelegator 実装用の `web.xml` ファイルに指定されているクラスを実行時にロードできません。

処置 : 指定されたクラスが OC4J クラス・ローダーを使用して使用可能であることを確認してください。

DAVSAXPackageHandler のエラー・メッセージ

エラー・コード : OSS-00180

無効な DAV URL *{0}* です

原因 : 指定された URL に接続し、URL を使用して WebDAV 接続を確立しようとしてクライアント・アプリケーションでエラーが発生しました。

処置 : 指定された URL が有効な WebDAV エンド・ポイントであり、DAV サーバーが稼働していることを確認してください。

エラー・コード : OSS-00181

パッケージの配信にリソース *{0}* を使用できません

原因 : 指定された DAV リソースを着信コンテンツ・パッケージの保存ルート宛先として使用しようとして、アプリケーションでエラーが発生しました。

処置 : 指定された DAV コレクションが存在し、クライアント・アプリケーションからアクセス可能で、接続ユーザーに書込み権限があることを確認してください。

エラー・コード : OSS-00182

アイテム *{0}* のリソースを作成できません

原因 : アプリケーションは、コンテンツ・パッケージを使用して配信された指定アイテムを格納する DAV リソースを作成できませんでした。

処置 : 宛先コレクションに対する書込み権限が接続ユーザーに付与されており、作成するリソースの名前が既存の DAV コレクションと競合していないことを確認してください。

エラー・コード : OSS-00183

アイテム *{0}* のリソースをフラッシュできません

原因 : アプリケーションは、指定されたアイテムと一緒に配信されたコンテンツを宛先 DAV リソースにフラッシュできませんでした。記憶領域が不十分か、宛先コレクションに対する権限に問題がある可能性があります。

処置 : 指定されたアイテムのコンテンツを保持するために十分大きな領域が宛先 DAV システムにあることを確認してください。

エラー・コード : OSS-00184

アイテム *{0}* のリソースを閉じることができません

原因 : リソース出力ストリームを閉じて、指定されたアイテムと一緒に配信された新規コンテンツを DAV リポジトリにコミットしようとしてエラーが発生しました。または指定されたタイムアウト値が小さすぎます。

処置 : 指定されたアイテムの宛先リソースが他のアプリケーションで使用されていないことを確認してください。また、DAV リポジトリに接続されているユーザーに、リソースの書込みおよび更新のための適切な権限があり、`oracle.syndicate.ui.provider.ProviderSyndDelegator.TimeoutSyndConn` 初期化パラメータに十分

なタイムアウト値が指定されていることも確認してください。詳細は、『Oracle Application Server Portal 構成ガイド』を参照してください。

エラー・コード : OSS-00185

アイテム *{0}* のリソースを削除できません

原因 : DAV リポジトリのリソースを削除しようとしてエラーが発生しました。

処置 : 指定されたアイテムの宛先リソースが他のアプリケーションで使用されていないことを確認してください。また、DAV リポジトリに接続されているユーザーに、リソースを削除する適切な権限があることも確認してください。

エラー・コード : OSS-00186

DAV エンドポイントに接続できません

原因 : DAV リポジトリに HTTP 接続を確立しようとしてエラーが発生しました。

処置 : DAV エンド・ポイントが作動しており、接続に使用したユーザー名とパスワードが有効で、認証システム（SSO/OID）が稼働していることを確認してください。

ProviderSyndRequestHandler のエラー・メッセージ

エラー・コード : OSS-00206

プッシュ・リクエストのチャネル ID が見つかりません

原因 : リクエスト URL にチャネル ID が正しく指定されていなかったため、プッシュ・リクエストを完了できません。

処置 : サーバーがサブスクライブ時に作成された URL を使用してコンテンツをプッシュしていることを確認してください。

エラー・コード : OSS-00207

リクエストされたチャネル ID *{0}* の WebDAV URL が見つかりません

原因 : プロバイダ・アプリケーションは、コンテンツ・リポジトリに接続するための DAV URL 設定を取得できません。

処置 : プロバイダ設定が正しく入力されていることをチェックしてください。これらの設定をチェックまたは更新するには、「デフォルトの編集」ページを使用します。

エラー・コード : OSS-00208

現行のトランザクションに対する接続が見つかりません

原因 : アプリケーションは、プッシュ・プロセスに関連付けられている現行トランザクションをコミットするための接続の取得に失敗しました。

処置 : プロバイダの Web アプリケーションを再起動してください。問題が解決されない場合は、管理者にお問い合せください。

ICE のエラー・メッセージおよび説明

ICE のエラー・メッセージは次のとおりです。

3*nn*: ペイロード・レベルのステータス・メッセージ

エラー・コード : ICE-300

一般致命的ペイロード・エラーです

原因 : 受信したペイロードを包含できないことを示す一般ステータス・コードです。

処置 : リクエスト・ペイロードをチェックして再送してください。問題が解決しない場合は、Syndication Services 管理者にリクエスタのアカウントのステータスをチェックするように依頼してください。

エラー・コード : ICE-301

ペイロードが不完全です / 解析できません

原因 : 送信されたペイロードに重大な誤りがあり、XML ペイロードではないため解析できません。XML ペイロードが期待されましたが、XML ではないペイロードが受信されました。

処置 : ペイロードをリセンドし、XML ペイロードであることを確認してください。

エラー・コード : ICE-302

ペイロードで XML が正常に構成されませんでした

原因 : 送信されたペイロードは XML として認識できますが、XML 定義による適切な書式ではありません。

処置 : Syndication Services 管理者に、対応するシンジケーション操作モジュールを修正するように依頼してください。

エラー・コード : ICE-303

ペイロードの検証に失敗しました

原因 : Document Type Definition（DTD）に従ったペイロードの検証に失敗しました。

処置 : リクエスト・ペイロードをチェックしてください。リクエスト・ペイロードは、Syndication Services により提供されている指定のリクエスト DTD に準拠する必要があります。

エラー・コード : ICE-304

一時的な応答問題です

原因 : 応答者がビジー状態であるか、更新が進行中などです。実際には、同じ再試行リクエストが成功する場合があります。

処置 : リクエスタのローカル・メッセージ・システムで、システム・ダウンタイムなどの通知の有無をチェックするか、後で再試行してください。

エラー・コード : ICE-320

互換性のないバージョンです

原因 : リクエストに使用された通信プロトコルはサポートされていません。

処置 : Syndication Services クライアント・ライブラリを使用している場合は、サポートされるプロトコル・バージョンのリストをチェックしてから、オラクル社カスタマ・サポート・センターにお問い合せください。Syndication Services クライアント・ライブラリを使用していない場合は、ローカル・システム管理者に連絡してください。

エラー・コード : ICE-331

外部データのフェッチに失敗しました

原因 : 受信者が、送信者から外部エンティティ参照として与えられた外部参照（URL）に従っていません。Syndication Services ではこのコードを使用して応答しないでください。

処置 : コンテンツ・オファーの参照リンクとその関連アクセス・ポリシー（HTTP または HTTPS に使用する証明書など）をすべて再確認してください。

エラー・コード : ICE-390

ペイロードの一時リダイレクトです

原因 : ペイロードが、提供された新規トランスポート通信エンド・ポイントを使用して一時的にリダイレクトされています。ICE または HTTP では、これは新規 URL が指定されたことを意味します。

処置 : オラクル社カスタマ・サポート・センターにお問い合せください。

エラー・コード : ICE-391

ペイロードの永続リダイレクトです

原因 : ペイロードが、提供された新規トランスポート通信エンド・ポイントを使用して永続的にリダイレクトされています。ICE または HTTP では、これは新規 URL が指定されたことを意味します。

処置 : オラクル社カスタマ・サポート・センターにお問い合せください。

4*nn*: リクエスト・レベルのステータス・メッセージ

エラー・コード : ICE-400

一般リクエスト・エラー

原因 : リクエストを包含できないことを示す汎用ステータス・コードです。

処置 : リクエスト・ペイロードのエラーの有無をチェックしてリセンドしてください。

エラー・コード : ICE-401

不完全です / 解析できません

原因 : 送信されたリクエストに重大な誤りがあり、解析できません。

処置 : リクエスト・ペイロードのエラーの有無をチェックし、リクエストが完全に送信された後でのみ通信チャネルを閉じてください。

エラー・コード : ICE-402

XML が正常に構成されませんでした

原因 : 送信されたリクエストは XML として認識できますが、XML 定義による適切な書式ではありません。

処置 : リクエスト・ペイロードのエラーの有無をチェックしてください。

エラー・コード : ICE-403

確認が失敗しました

原因 : リクエストを DTD に従って確認できませんでした。

処置 : リクエスト・ペイロードが Syndication Services により提供されているリクエスト DTD に準拠していることを確認してください。

エラー・コード : ICE-405

認識できない送信者です

原因 : 送信者には現行サーバーへのアクセス権が付与されていません。

処置 : Syndication Services 管理者に問い合せて、送信者情報を確認してください。

エラー・コード : ICE-406

サブスクリプションを認識できません

原因 : サブスクリプション ID が間違っているか、サブスクリプションが一貫した状態になっていません。

処置 : Syndication Services 管理者に問い合せて、送信者情報を確認してください。

エラー・コード : ICE-407

認識できない演算子です

原因 : リクエスト・タイプがサーバー側でサポートされていません。

処置 : 実行する演算子が Syndication Services でサポートされていることを確認してください。

エラー・コード : ICE-408

認識できない演算子の引数です

原因 : リクエスト引数がサーバー側でサポートされていません。

処置 : リクエスト・ペイロードを Syndication Services により提供されているリクエスト DTD と照らし合せてチェックしてください。エラーが解決されない場合は、オラクル社カスタマ・サポート・センターにお問い合せください。

エラー・コード : ICE-409

このサブスクリプションでは使用できません

原因 : リクエスタは、リクエストで参照されるサブスクリプションに該当しない情報を参照しています。

処置 : 条件を追加し、再度サブスクライブしてください。

エラー・コード : ICE-410

見つかりません

原因 : 検索対象が見つからないことを示す一般エラーです。

処置 : エラー・メッセージの詳細説明をチェックしてください。

エラー・コード : ICE-411

認識できないパッケージ順序の状態です

原因 : 送信者が指定したパッケージ順序の識別子を、受信者が認識できません。

処置 : ペイロードのエラーの有無をチェックしてください。エラーが解決されない場合は、オラクル社カスタマ・サポート・センターにお問い合せください。

エラー・コード : ICE-412

許可されていません

原因 : サブスクライバは、Syndication Services または一連の特定のコンテンツ・プロバイダへのアクセス権限を付与されていません。

処置 : Syndication Services 管理者にお問い合せください。

エラー・コード : ICE-413

禁止されています

原因 : サブスクライバのプロファイルによるアクセス違反です。

処置 : Syndication Services 管理者にお問い合せください。

エラー・コード : ICE-414

取引条件に違反します

原因 : 現行の操作または現行のリクエストの内容が、サブスクライバと Syndication Services 間の取引条件に準拠していません。

処置 : サブスクライバ・アカウントのビジネス規約を検討し、リクエストを調整してください。

エラー・コード : ICE-420

制約に失敗しました

原因 : Syndication Services から発生したコンテンツに対するサブスクライバのリクエストが、サブスクリプションに指定された制約に違反しているか、整合性がありません。

処置 : サブスクリプションの制約を検討し、サブスクライバのリクエスト・ペイロードを調整してください。

エラー・コード : ICE-422

スケジュール違反です。後で再試行してください

原因 : 不適切なタイミングでリクエストが発行されました。

処置 : 合意された時間枠内でパッケージ更新リクエストを発行してください。

エラー・コード : ICE-430

確認できません

原因 : 操作が確認されていないことを示す一般エラーです。

処置 : サブスクライバのローカル・システムの設定をチェックして確認を送信ください。エラーが解決されない場合は、オラクル社カスタマ・サポート・センターにお問い合せください。

エラー・コード : ICE-431

外部データのフェッチに失敗しました

原因 : 受信者が、送信者から与えられた外部参照（URL）に従っていません。

処置 : サブスクライバのローカル・システムの設定をチェックし、このアクションを使用可能にしてください。

エラー・コード : ICE-440

申し訳ありません

原因 : このレスポンスは、ネゴシエーション・プロセスの一部として、提案がなんらかの理由で受入れ不能であることを示します。

処置 : サブスクリプション・リクエストを検討し、必要な変更を行ってからリクエストを再送してください。

エラー・コード : ICE-441

代替案

原因 : このレスポンスは、ネゴシエーション・プロセスの一部として、更新されたオファーまたは代替案を示します。

処置 : 代替案のオファー・サブスクリプション・リクエストを検討し、必要な変更を行ってから応答を送信してください。

エラー・コード : ICE-442

ネゴシエーションが再び進行中です

原因 : このレスポンスは、ネゴシエーション・プロセスの一部として、再ネゴシエーションが処理中であることを示します。

処置 : ネゴシエーション・プロセスに進んでください。

エラー・コード : ICE-443

オファーが確認されましたが遅延されました

原因 : パラメータのネゴシエーション中に使用されます。オファーの受信者は、ユーザーの介入なしではオファーを評価できません。送信者は、提供されたパラメータおよび同じサブスクリプション ID を使用して、後で再試行できます。受信者は、後から（介入結果を反映して）新規オファーでレスポンスを返すことが期待されます。

処置 : 妥当な時間待機した後に応答がない場合は、Syndication Services 管理者に問い合せてネットワーク・エラーやシステム・エラーがあるかどうかを確認してください。これが、応答が受信されなかった理由である場合があります。明らかなネットワーク・エラーやシステム・エラーがなかった場合、受信者が送信者に直接連絡してなんらかの合意に達し、その時点で送信者が新規オファーを送信するか、送信者が前回のオファーを後でリセンドできます。

エラー・コード : ICE-450

範囲タイプが無効です

原因 : パラメータのネゴシエーション中に使用されます。数値、字句、時刻または列挙型のプロトコル固有の範囲が正しく指定されていません。

処置 : リクエスト・ペイロード内の対応するセクションを検討して訂正してください。

エラー・コード : ICE-451

スパン選択が範囲外です

原因 : パラメータのネゴシエーション中に使用されます。範囲内で選択された値が最小値より小さいか、最大値を超えています。

処置 : リクエスト・ペイロードを検討して訂正してください。

エラー・コード : ICE-452

選択エラー

原因 : パラメータのネゴシエーション中に使用されます。列挙内で選択された項目数が select 属性と一致しません。たとえば、select 属性では 1 つ以上の項目を選択するようにリクエストしているにもかかわらず、レスポンスでは項目が選択されていません。

処置 : リクエスト・ペイロードを検討して訂正してください。

エラー・コード : ICE-453

列挙の選択が空または無効です

原因 : パラメータのネゴシエーション中に使用されます。選択可能な ice-enum-item 要素の数が、select 属性と一致しません。たとえば、select 属性は ice-enum-item 要素を 1 つ以上選択する必要があることを示しているにもかかわらず、実際に使用可能な項目数が列挙内の ice-enum-item 要素の数より少なくなっています。

処置 : リクエスト・ペイロードを検討して訂正してください。

エラー・コード : ICE-454

スパンが無効です

原因 : パラメータのネゴシエーション中に使用されます。範囲の値が構造に違反しています。たとえば、スパン・ポイントの値には、スパンの最小値より小さい値や最大値より大きい値は指定できません。

処置 : リクエスト・ペイロードを検討して訂正してください。

エラー・コード : ICE-456

比較できない時刻書式です

原因 : パラメータのネゴシエーション中に使用されます。ネゴシエーション中に比較された 2 つの時間値のフォーマットに互換性がありません。たとえば、期間が ICE の日付または時刻と比較されています。

処置 : リクエスト・ペイロードを検討して訂正してください。

5*nn*: 実装エラーおよび操作エラーのメッセージ

エラー・コード : ICE-500

一般内部応答エラー

原因 : これは、一般的な問題に関する汎用的なレスポンスです。

処置 : Syndication Services 管理者にお問い合せください。

エラー・コード : ICE-501

一時的な応答問題です

原因 : 応答者がビジー状態であるか、更新の実行中などです。

処置 : 実際には、同じ再試行リクエストが成功する場合があります。

エラー・コード : ICE-503

実装されていません

原因 : リクエストされた操作がサービスに実装されていません。

処置 : 実行する操作が Syndication Services でサポートされていることを確認してください。

6*nn*: 保留状態のステータス・メッセージ

エラー・コード : ICE-601

非送信請求メッセージをただちに処理する必要があります

原因 : シンジケータにはサブスクライバに送信する非送信請求メッセージがありますが、サブスクライバはそれをまだリクエストしていません。

処置 : 保留中のパッケージに対するリクエストを Syndication Services に発行してください。

エラー・コード : ICE-602

未処理の確認が多すぎます

原因 : シンジケータはパッケージ配信の確認をリクエストしており、サブスクライバから確認（肯定または否定）が与えられるまでは追加の操作の実行を拒否します。

処置 : リクエストされたパッケージ配信に関する確認を発行してください。

エラー・コード : ICE-603

送信する確認はありません

原因 : サブスクライバがすでに送信したと想定するパッケージ配信確認を、Syndication Services がリクエストしました。

処置 : 措置は不要です。

エラー・コード : ICE-604

任意形のメッセージはありません

原因 : サブスクライバは `ice-unsolicited-now` メッセージを送信しましたが、シンジケータには送信する任意形のメッセージがありません。

処置 : サブスクライバのローカル・システムをチェックし、非一貫性状態が続く場合はオラクル社カスタマ・サポート・センターにお問い合せください。

B

Syndication Services のセキュリティ

この付録では、Syndication Services のセキュリティのアーキテクチャと構成について説明します。内容は次のとおりです。

- Syndication Services のセキュリティについて
- Syndication Services のセキュリティの構成

Syndication Services のセキュリティについて

Syndication Services のユーザー・サポートでは、LDAP ベースのプロバイダと OC4J Java Authentication and Authorization Service を使用しています。この項では、次の内容について説明します。

- Syndication Services のリソースの保護
- 保護されている Syndication Services リソースの管理と規定
- Oracle Application Server Security のサービスの使用

ユーザー管理の詳細は、2-4 ページの「ユーザー管理の理解」を参照してください。

Syndication Services のリソースの保護

Syndication Services のセキュリティにより、次のリソースが保護されます。

- データ ― Syndication Services リポジトリに格納されているデータに対する保護された書込みアクセス。これは、通常 Syndication Services のメタデータです。
- 機能 ― Syndication Services リポジトリに対する管理操作。
- パスワード ― パスワードおよび次を含むその他のセキュリティ関連データ。
  - コンテンツ・プロバイダのプロパティの値
  - プッシュ URL に対するユーザーの認証情報
  - Portal シンジケーション・ユーザーのパスワード

  Syndication Services のユーザー管理の詳細は、2-4 ページの「ユーザー管理の理解」を参照してください。Portal シンジケーション・ユーザーおよび Oracle Application Server Portal へのコンテンツのシンジケーションの詳細は、『Oracle Application Server Portal 構成ガイド』を参照してください。

保護されている Syndication Services リソースの管理と規定

JAZN（Java Authentication and Authorization Service（JAAS）の Oracle による実装）と Syndication Services アプリケーションは、Syndication Services リポジトリに格納されているデータへの書込みアクセスを管理および規定します。JAZN は、ユーザーの ID とセキュリティ・ロールを決定します。データの更新権限があるのは、所有者のみです。Syndication Services のユーザー管理の詳細は、2-4 ページの「ユーザー管理の理解」を参照してください。

管理操作に関して、Syndication Services は Oracle Enterprise Manager により管理されます。Oracle Enterprise Manager は、管理操作を提供するサーブレットを保護します。

パスワードおよびその他の機密性の高い情報はデータベース内に永続的に格納され、データベースの `DBMS_OBFUSCATION` PL/SQL パッケージによりさらに保護されます。

Oracle Application Server Security のサービスの使用

Syndication Services では JAZN のユーザー・レベル・セキュリティ機能を最適化し、Oracle Application Server の Infrastructure データベース・オプションにアクセスするために、サーバー側とクライアント側の両方で Secure Socket Layer（SSL）暗号化を使用します。

Syndication Services のセキュリティの構成

「Syndication Services のリポジトリ」および「Syndication Services のクライアント」では Syndication Services のセキュリティの構成について説明します。

Syndication Services のリポジトリ

Syndication Services とクライアント間での通信の機密性を保つ方法は、次のとおりです。

1. Oracle HTTP Server（OHS）/SSL リスナーを構成して、HTTPS アクセスを提供します。
2. OC4J を構成して HTTP アクセスを禁止します。

この構成は、機密性の高い Syndication Services エンド・ポイントのすべてに対して実施する必要があります。

Syndication Services のクライアント

Syndication Services リポジトリと通信するアプリケーションの開発に、提供されている Syndication Services クライアント・ライブラリを使用する場合は、HTTPS 接続で使用する資格証明や使用する HTTP プロキシなどの接続プロパティを設定できます（3-3 ページの「Syndication Services の接続のオープン」を参照）。また、Oracle Enterprise Manager のセキュリティ機能を使用して、HTTP トランスポートの様々なセキュリティ・プロパティを構成することもできます。

C

シンジケーション・クライアントのサンプル

この付録では、SampleSyndicationClient.java プログラムのリストを示します。「SampleSyndicationClient.java プログラムのリスト」を参照してください。

SampleSyndicationClient.java プログラムのリスト

SampleSyndicationClient.java プログラムのリストを次に示します。

```
import java.util.*;

import oracle.syndicate.*;
import oracle.syndicate.client.*;
import oracle.syndicate.client.handler.file.*;

import oracle.syndicate.message.*;
import oracle.syndicate.message.pkg.*;
import oracle.syndicate.message.offer.*;
import oracle.syndicate.message.response.*;
import oracle.syndicate.message.response.event.*;

/**
 *  @version $Header: SampleSyndicationClient.java 07-mar-2003.19:12:51 mcarrer Exp $
 *  @author  mcarrer
 *  @since   release specific (what release of product did this appear in)
 */
public class SampleSyndicationClient
{
  private SyndicateConnection _sc;

  public SampleSyndicationClient()
  {
  }

  public static void main(String args[])
    throws Exception
  {
    if (args.length < 3) {
      System.err.println("java SampleSyndicationClient ossurl username password");
      System.exit(0);
    }

    SampleSyndicationClient ssc = new SampleSyndicationClient();

    //
    // Step 0: Initialize syndication connection
    // and ping to the server to make sure it
    // up and running, and we can authenticate.
    //
    ssc.initConnection(args);
    ssc.ping();
```

```
  //
  // Step 1: Get the catalog of offers available
  // and select the first offer appearing in the catalog.
  //
  Catalog cat = ssc.getCatalog();
  Offer   off = ssc.getFirstOffer(cat);

  //
  // Step 2: Subscribe to the selected offer.
  //
  Subscription sbt = ssc.subscribe(off);

  //
  // Step 3: Get the content package associated
  // with the new subscription. If necessary,
  // confirm to the syndication server the
  // receipt of the content package.
  //
  //SyndicatePackage pkg = ssc.getPackage(sbt);
  //ssc.confirmPackage(pkg);

  //
  // Step 4: Verify the subscription status by
  // checking subscription expiration criteria,
  // such as the quantity remaining.
  //
  ssc.status(sbt);

  //
  // Step 5: Get session events.
  //
  ssc.events(sbt);

  //
  // Step 6: Cancel subscription.
  //
  ssc.cancel(sbt);
}


/**
 * Creates a SyndicateConnection using
 * an XML state handler storing client state
 * in the synd-client.xml file in the current directory.
 */
```

```
  private void initConnection(String argv[])
    throws SyndicateException
  {
    // Acquire a SyndicateConnectionFactory,
    // optionally set the connection parameters,
    // such as proxy info, timeout, and credentials.
    SyndicateConnectionFactory scf = SyndicateConnectionFactory.getInstance();
    // scf.setTimeout(1000);
    // scf.setProxy("myproxyhost", iMyProxtPort);

    // Create a default XML state handler storing subscription state in a file.
    SyndicateClientStateHandler scsh =
 scf.getDefaultSyndicateClientStateHandler("synd-client.xml");
    _sc = scf.createSyndicateConnection(argv[0],  // url of the oss server
                                        argv[1],  // username
                                        argv[2],  // password
                                        scsh);
  }

  private void ping()
    throws SyndicateException
  {
    Response resp = _sc.ping();
    System.out.println("ping: "+resp.getCode().getPhrase());
  }

  private Catalog getCatalog()
    throws SyndicateException
  {
    return _sc.getCatalog();
  }

  private Offer getFirstOffer(Catalog cat)
    throws SyndicateException
  {
    // Iterates through the catalog and returns the
    // first offer encountered.
    Offer   off = null;
    Iterator it = cat.getCatalogItems();
    while (it.hasNext() && (off == null)) {
      CatalogItem item = (CatalogItem)it.next();
      if (item.getCatalogItemType() == CatalogItem.OFFER) {

        off = (Offer) item;
      }
      else {
        // You got an offer group.
```

```
      off = getFirstOffer((OfferGroup) item);
    }
  }

  // Print a few offer details.
  System.out.println("offer: "+off.getID()+" - "+off.getDescription());
  DeliveryPolicy dp = off.getDeliveryPolicy();
  if (dp.getStartDate() != null) {
    System.out.println("¥t start date: "+dp.getStartDate());
  }
  if (dp.getStopDate() != null) {
    System.out.println("¥t  stop date: "+dp.getStopDate());
  }
  Iterator dlrs = dp.getDeliveryRules();
  while (dlrs.hasNext()) {
    DeliveryRule dlr = (DeliveryRule) dlrs.next();
    System.out.println("¥t  dlr mode: "+dlr.getMode());
  }
  return off;
}


private Offer getFirstOffer(OfferGroup offgrp)
  throws SyndicateException
{
  Offer   off = null;
  Iterator it = offgrp.getCatalogItems();
  while (it.hasNext() && (off == null)) {

    CatalogItem item = (CatalogItem)it.next();
    if (item.getCatalogItemType() == CatalogItem.OFFER) {

      off = (Offer) item;
    }
    else {
      // You got an offer group.
      off = getFirstOffer((OfferGroup) item);
    }
  }
  return off;
}

private Subscription subscribe(Offer off)
  throws SyndicateException
{
  // If the offer contains a push delivery rule,
  // you can use the offer APIs to set the destination URL.
```

```
  DeliveryPolicy dp = off.getDeliveryPolicy();
  Iterator   itdlrs = dp.getDeliveryRules();
  while (itdlrs.hasNext()) {

    DeliveryRule dlr = (DeliveryRule) itdlrs.next();
    if (DeliveryRule.DELIVERY_RULE_MODE_PUSH.equals(dlr.getMode())) {

      dlr.setURL("http://mysyndicationclient.com/syndclient/listener");
      // Optionaly set username/password.
      // dlr.setPushAuthUsername("me");
      // dlr.setPushAuthUsername("pwd");
      // If raw content is requested (for example, a mailto or
      // ftp url has been supplied), set the raw content flag.
      // dlr.setRawFormatSupport(true);
    }
  }

  Subscription sbt = _sc.subscribe(off);
  System.out.println("subscription: "+sbt.getSubscriptionID());
  return sbt;
}

private SyndicatePackage getPackage(Subscription sbt)
  throws SyndicateException
{
  // Use a FileSAXPackageHandler so that
  // syndicate content will be stored in
  // the local file system /tmp directory.
  FileSAXPackageHandler fsph = FileSAXPackageHandler.getInstance("/tmp/");

  // This will get content incremented since the last update.
  // The current subscriber state will be fetched from
  // the SyndicateClientStateHandler used by this connection.
  // To request a full update of the content versus an incremental
  // one, use the following syntax:
  // _sc.getPackage(sbt.getSubscriptionID(),
  //                SyndicateSubscription.STATE_ICE_INITIAL,
  //                null,
  //                fsph);
  SyndicatePackage sp = _sc.getPackage(sbt.getSubscriptionID(), null, fsph);
  return sp;
}
```

```
 private void confirmPackage(SyndicatePackage pkg)
   throws SyndicateException
 {
   // Check if the package requires confirmation.
   // If so, confirm it.
   if (pkg.hasConfirmation()) {

     Response resp = _sc.confirm(pkg.getID());
     System.out.println("confirmed "+pkg.getID()+":
"+resp.getCode().getPhrase());
   }
 }

 private void status(Subscription sbt)
   throws SyndicateException
 {
   Status sts = _sc.getStatus(sbt.getSubscriptionID());

   Subscription sbtNew = (Subscription) sts.getSubscriptions().next();
   System.out.println(" sbt "+sbtNew.getSubscriptionID()+":");
   System.out.println("     expiration priority: "+
                      Subscription.EXPIRATION_PRIORITIES[sbtNew.getExpirationPriority()]);
   System.out.println("     expiration date: "+sbtNew.getExpirationDate());
   System.out.println("     quantity remaining: "+sbtNew.getQuantityRemaining());
 }

 private void cancel(Subscription sbt)
   throws SyndicateException
 {
   Cancellation canc = _sc.cancelSubscription(sbt.getSubscriptionID(),
                                              "boring", "en_us");

   System.out.println("cancelled: "+canc.getCancellationID());
 }


 private void events(Subscription sbt)
   throws SyndicateException
 {
   Events evt = _sc.getEvents(null, null, sbt.getSubscriptionID());
   EventLog evtlog = evt.getEventLog();
   Iterator itEvts = evtlog.getEventItems();
   while (itEvts.hasNext()) {

     EventItem ei = (EventItem) itEvts.next();
     if (ei.getType() ==  EventItem.EVENT_TYPE_MSG) {
```

```
        EventMsg evtmsg = (EventMsg) ei;
        System.out.println(evtmsg.getRequestStart()+" "
                           +evtmsg.getRequest()+" "+
                           evtmsg.getResponseType());
      }
    }
  }
}
```

D

RSS コンテンツ・コネクタ（CPAdaptor）

この付録では、RSSCPAdaptor.java プログラムのリストを示します。
「RSSCPAdaptor.java プログラムのリスト」を参照してください。

RSSCPAdaptor.java プログラムのリスト

RSSCPAdaptor.java プログラムのリストを次に示します。

```
import java.io.*;
import java.net.*;
import java.text.*;
import java.util.*;

import org.w3c.dom.*;
import javax.xml.parsers.*;
import oracle.xml.parser.v2.*;

import oracle.syndicate.message.pkg.*;
import oracle.syndicate.server.cp.*;
import oracle.syndicate.server.cp.message.*;
import oracle.syndicate.server.cp.impl.exmsgs.*;

import oracle.syndicate.util.date.*;

// For testing only.
import oracle.syndicate.util.io.*;
import oracle.syndicate.server.cp.impl.message.*;

/**
 * Content provider adaptor (content connector) creates a
 * CPOffer for each channel defined in a given RSS feed. CPOffers will be
 * shown to Syndication Services administrators as resources, for which they can create offers.
 * RSS (Really Simple Syndication) is a mechanism for enabling
 * lightweight syndication of content.
 * More details on RSS can be found at: http://backend.userland.com/rss.
 * The adaptor exposes only one property:
 * the url of the RSS feed for which the offer has to be created.
 * Custom exception messages can be supplied by subclassing the CPException
 * class.
 * The adaptor code can be compiled as follows:
 *   javac -d demo/cp/rss/classes -classpath lib/syndserver.jar:../../xdk/lib/xmlparserv2.jar
demo/cp/rss/src/RSSCPAdaptor.java
 * Standalone unit tests can be run by:
 *   java -Dhttp.proxyHost=www-proxy.us.oracle.com -Dhttp.proxyPort=80 -classpath
lib/syndserver.jar:../../xdk/lib/xmlparserv2.jar:lib/redist/jazn/jazn.jar:demo/cp/rss/classes/
RSSCPAdaptor http://static.userland.com/gems/backend/rssTwoExample2.xml "Scripting News"
 *
 * To install:
 *   java -DORACLE_HOME=$OH -jar lib/syndserver.jar -installConnector -name "RSS Content Connector"
-description "Allows to create offers based on RSS feeds" -className RSSCPAdaptor
 *
 *  @version $Header: RSSCPAdaptor.java 03-apr-2003.13:23:05 mcarrer Exp $
```

```
 *  @author  mcarrer
 *  @since   release specific (what release of product did this appear in)
 */
public class RSSCPAdaptor
  implements CPAdaptor, OSSExceptionConstants
{
  private static final String LAST_BUILD_DATE_FORMAT = "EEE, dd MMM yyyy H:mm:ss z";

  private String    _rssurl;
  private CPContext _cpctx;

  public RSSCPAdaptor()
  {}

  /**
   * Returns the URL of the RSS feed.
   */
  public String getRSSURL()
  {
    return _rssurl;
  }

  /**
   * Sets the URL of the RSS feed.
   */
  public void setRSSURL(String rssurl)
  {
    _rssurl = rssurl;
  }

  /**
   * Receives and stores the CPContext from which
   * you can access the CPMessageFactory object.
   */

  public void init(CPContext cpctx)
    throws CPException
  {
    _cpctx = cpctx;
  }

  public boolean ping()
    throws CPException
  {
    // TODO
    return true;
  }
```

```
/**
 * Returns only one offer corresponding to the RSS content feed.
 */
public Iterator buildCatalog()
  throws CPException
{
  Document docrss = parseRSS();
  Element   elrss = docrss.getDocumentElement();
  NodeList  nlchs = elrss.getElementsByTagName("channel");

  // Loop over the channels defined in this RSS
  // and for each one, create an offer.
  CPMessageFactory cpmf = _cpctx.getCPMessageFactory();
  ArrayList      alOffs = new ArrayList();
  for (int i=0; i<nlchs.getLength(); i++) {

    // For each channel, you will create
    // a CPOffer. CPOffers are shown to
    // Syndication administrators as content
    // provider resources from which you can build offers.
    Element  elCh = (Element) nlchs.item(i);
    CPOffer cpoff = cpmf.createCPOffer();

    // Each offer must have a unique ID for its content provider.
    // Set the mandatory ID attribute and any other
    // optional attributes.
    String title = getChildElementValue(elCh, "title");
    if ((title == null) || (title.trim().length() == 0)) {
      title = "unknown";
    }

    // Set the offer properties.
    cpoff.setCPOfferID(title);
    cpoff.setName(title);
    cpoff.setDescription(getChildElementValue(elCh,  "description"));
    cpoff.setRightsHolder(getChildElementValue(elCh, "copyright"));
    cpoff.setProductName("OracleAS Syndication Services RSS Feed Import");
    cpoff.setAtomicUse(false);
    cpoff.setEditable(true);
    cpoff.setShowCredit(false);
    cpoff.setUsageRequired(false);

    alOffs.add(cpoff);
  }
```

```
  return alOffs.iterator();
}

public CPPackage buildPackage(CPPackageRequest req,
                              CPRequestContext ctx)
  throws CPException
{
  // Parse the RSS document.
  Document docrss = parseRSS();
  Element   elrss = docrss.getDocumentElement();
  Element    elCh = getChannelElement(elrss, req.getOfferID());

  // Create a new content package and initialize
  // its properties by using RSS feed values.
  CPMessageFactory cpmf = _cpctx.getCPMessageFactory();
  CPPackage  pkg = cpmf.createCPPackage();

  // Check if the RSS feed has been modified since the
  // the last content package request.
  String oldState = req.getState();
  String newState = getChildElementValue(elCh, "lastBuildDate");

  Date dNewState = getDate(newState);
  Date dOldState = null;
  if (!oldState.equals(INITIAL_STATE)) {

    dOldState = getDate(oldState);
    if ((dOldState != null) &&
        (dNewState != null) &&
        dOldState.equals(dNewState)) {

      // The RSS feed has not been updated
      // since the last visit. Throw an exception
      // notifying that the content package sequence is
      // up-to-date and no update needs to be reported.
      throw new CPException(CPExceptionConstants.CP_PACKAGE_UP_TO_DATE);
    }
    // This is an incremental update over
    // the previous update that was sent.
    // Set the full update flag accordingly.
    pkg.setFullUpdate(false);
  }
  else {

    // This is a request for a full new update.
    // Set the full update flag accordingly.
    pkg.setFullUpdate(true);
```

```
}
pkg.setOldState(oldState);
pkg.setNewState(newState);

// Scan through the RSS items, building
// corresponding CPItem objects.
NodeList nlItems = elCh.getElementsByTagName("item");
for (int i=0; i<nlItems.getLength(); i++) {

  Element elItem = (Element) nlItems.item(i);
  CPItem cpi = cpmf.createCPItem();

  // Sets item file name and content type.
  // Each RSS news item will have a dummy file name
  // and an XML code excerpt for the associated news.
  String itemFilename = "item"+i+".xml";
  cpi.setContentFilename(itemFilename);
  cpi.setContentType("text/xml");

  // Sets the item ID.
  String guid = getChildElementValue(elItem, "guid");
  if (guid != null) {
    cpi.setID(guid);
  }

  // Sets the item name.
  String title = getChildElementValue(elItem, "title");
  if (title != null) {
    cpi.setName(title);
  }
  else {
    // Item name is mandatory,
    // so you need to set it
    // to a value.
    cpi.setName(itemFilename);
  }

  // Sets the item activation date, if any.
  String pubDate = getChildElementValue(elItem, "pubDate");
  if (pubDate != null) {
    Date dPubDate = getDate(pubDate);
    cpi.setActivation( new ISODateTime(dPubDate));
  }

  // Set the item content to the RSS item XML element.
  try {
    StringWriter sw = new StringWriter();
```

```
      ((XMLElement) elItem).print( new PrintWriter(sw));
      ContentAccessor ca = cpmf.createContentAccessor(sw.toString());
      cpi.setContent(ca);
    }
    catch(Exception e) {
      throw new CPException(CPExceptionConstants.CP_GENERIC_ERROR, e, e);
    }

    // Add the item to the content package.
    pkg.addPackageItem(cpi);
  }

  return pkg;
}

public void closePackage(CPRequestContext ctx)
  throws CPException
{
}

private Document parseRSS()
  throws CPException
{
  Document doc = null;
  InputStream is = null;
  try {

    URL url = new URL(_rssurl);
    URLConnection urlc = url.openConnection();
    is = urlc.getInputStream();

    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
    javax.xml.parsers.DocumentBuilder db = dbf.newDocumentBuilder();
    doc = db.parse(is);
  }
  catch (Exception e) {
    throw new CPException(CPExceptionConstants.CP_GENERIC_ERROR, e, e);
  }
  finally {
    if (is != null) {
      try { is.close(); }
      catch (Exception e) {
        throw new CPException(CPExceptionConstants.CP_GENERIC_ERROR, e, e);
      }
    }
  }
```

```
  return doc;
}


private String getChildElementValue(Element el,
                                    String tagName)
  throws CPException
{
  String value = null;
  try {

    NodeList  nl = el.getElementsByTagName(tagName);
    if (nl.getLength() > 0) {

      Element elChild = (Element) nl.item(0);
      NodeList nl1 = elChild.getChildNodes();
      for(int i=0; i<nl1.getLength(); i++) {

        Node nChild = nl1.item(i);
        int  iNodeType = nChild.getNodeType();
        if (iNodeType == Node.TEXT_NODE) {
          value = nChild.getNodeValue();
          break;
        }
      }
    }
  }
  catch (Exception e) {
    throw new CPException(CPExceptionConstants.CP_GENERIC_ERROR, e, e);
  }

  return value;
}

private Element getChannelElement(Element el,
                                  String chName)
  throws CPException
{
  Element elch = null;
  try {

    NodeList nl = el.getElementsByTagName("channel");
    for (int i=0; i<nl.getLength(); i++) {

      Element elTemp = (Element) nl.item(i);
      String  chTitle = getChildElementValue(elTemp, "title");
      if (chTitle.equals(chName)) {
```

```
        elch = el;
        break;
      }
    }
  }
  catch (Exception e) {
    throw new CPException(CPExceptionConstants.CP_GENERIC_ERROR, e, e);
  }

  return elch;
}

private Date getDate(String sd)
  throws CPException
{
  Date d = null;
  try {

    if ((sd != null) && (sd.trim().length() != 0)) {
      SimpleDateFormat sdf = new SimpleDateFormat(LAST_BUILD_DATE_FORMAT);
      d = sdf.parse(sd);
    }
  }
  catch (Exception e) {
    throw new CPException(CPExceptionConstants.CP_GENERIC_ERROR, e, e);
  }

  return d;
}

public static void main(String argv[])
  throws Exception
{
  String rssurl = argv[0];
  String  offid = argv[1];

  RSSCPAdaptor rsscp = new RSSCPAdaptor();
  rsscp.init( new oracle.syndicate.server.cp.impl.CPContextImpl());
  rsscp.setRSSURL(rssurl);

  Iterator itOffs = rsscp.buildCatalog();
  while (itOffs.hasNext()) {

    CPOffer cpoff = (CPOffer) itOffs.next();
    System.out.println("id: "+cpoff.getCPOfferID());
    System.out.println("name: "+cpoff.getName());
    System.out.println("desc: "+cpoff.getDescription());
```

```
      System.out.println("rh: "+cpoff.getRightsHolder());
      System.out.println("pn: "+cpoff.getProductName());
    }

    CPPackage pkg = rsscp.buildPackage( new CPPackageRequestImpl(null, null,
                                                                 offid, null,
                                                                 null),
                                        null);
    Iterator itItems = pkg.getPackageItems();
    while (itItems.hasNext()) {

      CPItem cpi = (CPItem) itItems.next();
      System.out.println("id: "+cpi.getID());
      System.out.println("title: "+cpi.getName());
      System.out.println("actv: "+cpi.getActivation());

      ContentAccessor ca = cpi.getContent();
      InputStream cais = ca.openStream();
      ByteArrayOutputStream baos = new ByteArrayOutputStream();
      IOUtils.pipe(cais, baos);
      System.out.println(baos.toString());

      System.out.println("");
    }
  }
}
```

索引

く

け

こ

さ

し

と

は

ふ

ゆ

よ

り