Product Advertising API (Javari.jp)
クイックスタートガイド

API Version 2010-01-01

(Javari に関しては、2010/07/08 サポート開始)

Product Advertising API クイックスタートガイド (API Version 2010-01-01)

Copyright © 2007-2008 Amazon Web Services LLC or its affiliates. All rights reserved.

目次

ようこそ

トピック

- 対象読者
- 読者からのフィードバック
- 本ガイドの使用法
- 関連リソース

本ガイドは*『Product Advertising API (Javari.jp)クイックスタートガイド』*です。この項では、本ガイドの対象読者や構成、および Product Advertising API に関連した他のリソースについて説明します。

対象読者

本ガイドは、www.javari.jp において販売されている商品を紹介するストアフロントサイトを構築する開発者、またはストアフロントサイトの構築に役立つアプリケーションを開発する開発者を対象としています。

読者からのフィードバック

本 ガイドのオンラインバージョンには、本ガイドの読者からのフィードバックを入力できるリンクがあります。弊社では、可能な限り不備や誤りのない読みやすい ガイドの作成に努めています。これには、読者のみなさまからのフィードバックが非常に役に立ちます。たくさんのフィードバックをお待ちしています。

必要な知識とスキル

本ガイドの説明は、読者に以下の知識があることを前提としています。

- XML に関する知識（概要については、*『W3 Schools XML Tutorial』*を参照してください）
- ウェブサービスに関する基本的な知識（概要については、*『W3 Schools Web Services Tutorial』*を参照してください）

さらに、本ガイドの読者は次のいずれかのプログラミング言語の使用に習熟している必要があります。

- PHP

- C#
- Java
- Perl

本ガイドの使用法

本 ガイドは、全体的な説明とチュートリアルの提供を目的とした編成になっています。本ガイドでは、Product Advertising API を使用しながら簡 易環境で学習できるように、コード例をいくつかの項に分けて示します。各項の内容は、その前の項の内容を引き継いでいるため、コード例を順番に読み進める ことで、Product Advertising API に関する基礎知識が得られ、簡単な作業アプリケーションも作成できるようになります。

本ガイドの主な項目

- **準備作業** -- リクエスト送信に必要な ID の取得方法について説明します。
- **リクエストの作成** -- 複数のプログラミング言語による簡単な Product Advertising API リクエストの送信方法について説明します。
- **レスポンスの処理** -- リクエストに対するレスポンスの構文解析法について説明します。
- **サポート対象のオペレーションについて** - Javari 向け Product Advertising API においてサポートされているオペレーションについて説明しています。
- **次のステップ** -- Product Advertising API の理解を深めるための発展的な作業について説明します。

関連リソース

以下の表は、本サービスを利用する際に役立つ関連リソースをまとめたものです。

リソース	説明
『*Product Advertising API Developer Guide*』(http://docs.amazonwebservices.com/AWSECommerceService/latest/DG/)	この開発者用ガイドは本サービスを詳しく解説したものです。アーキテクチャの概要、プログラミングリファレンス、API リファレンスが収録されています。
『*Product Advertising API Release Notes*』(http://developer.amazonwebservices.com/connect/kbcategory.jspa?categoryID=17)	このリリースノートは最新リリースの概要を示したものです。とくに、新機能、訂正箇所、既知の問題点が記されています。
AWS Developer Resource Center ((http://developer.amazonwebservices.com/connect/kbcategory.jspa?categoryID=5)	ドキュメンテーション、コード例、リリースノートをはじめとする AWS ベースの革新的なアプリケーション開発に役立つさまざまな情報が収められたリソースセンターです。
Product Advertising API (https://affiliate.amazon.co.jp/gp/advertising/api/detail/main.html)	Product Advertising API についての主な情報源であるウェブページです。
ディスカッションフォーラム 日本語 (http://developer.amazonwebservices.com/connect/forum.jspa?forumID=19) 英語 (http://developer.amazonwebservices.com/connect/forum.jspa?forumID=19)	Amazon Web サービス全般に関する技術的な質疑応答の場である開発者向けのコミュニティです。
AWS アカウントに関連した質問の窓口となる E メールアドレス：webservices@amazon.com	この E メールアドレスは、アカウントに関する質問*専用*の窓口です。技術的な質問については、ディスカッションフォーラムをご利用ください。

Product Advertising API 入門

トピック

- Product Advertising API の概要
- Product Advertising API の主なコンセプト
- コード例の概要

この Product Advertising API 入門では、本サービスの概要を示しています。この項では、本ガイドに記載されているコード例を読むために必要な基本事項について説明します。

Product Advertising API の概要

Amazon では、10 年以上の歳月と数億ドルの資金を投じて世界規模のウェブサービスを開発し、毎日数百万の開発者の方々にご利用いただいております。堅牢でスケーラ ブル、信頼性も高いこの技術を利用して、開発者は Product Advertising API 用のアプリケーションを独自に開発できます。出品商品や、カスタ マーレビュー、出品者の評価といった Amazon のデータの大部分にアクセスできるだけでなく、商品の検索や、関連商品の検索、カスタマーレビューの表 示、商品プロモーションなど、www.amazon.co.jp の大部分の機能、および www.javari.jp の一部の機能を使用することができます。つまり、Product Advertising API を 利用すれば、Amazon および Javari のデータベースにアクセスでき、Amazon や Javari の提供する高性能な E コマースデータやオンライン機能が利用できるようになるという ことです。Amazon や Javari の商品や自分の売りたい品物を販売するオンラインストアを自分で構築できるようになります。

Product Advertising API は無料で利用できます。すでに数多くの Product Advertising API 利用者が、このサービスを利用したアプリケーションやウェブストアを開発し、アソシエイトとしての報酬を手にしています。Amazon アソシエイト・プログラムメンバーへの登録と併せて、Product Advertising API の利用登録を行うことで、誰でも Product Advertising API をによるアソシエイト報酬を獲得できます。全世界の Product Advertising API の開発者が 2006 年度に販売した商品の売り上げは、6 億ドルを大きく上回る額に達しています。この収益の一部を手にしてみませんか？

特徴

Product Advertising API の主な特徴は次のとおりです。

- **Amazon ならびに Javari の商品カタログを利用可能—**Amazon ならびに Javari の商品データベースにアクセスできます。
- **商品画像の表示—**www.amazon.co.jp および www.javari.jp にある商品画像を表示できます。
- **最新サービスを利用可能—**デジタルメディアなど、Amazon の最新サービスを利用できます。

本ガイドでは、検索条件を指定し、その条件に一致した商品を返す ItemSearch リクエストの簡単なコード例を示します。

Product Advertising API の主なコンセプト

ItemSearch リクエストでは、カスタマーの要望に合わせて商品リストを絞り込むためのさ まざまなパラメータを使用できます。レスポンスグループは、デフォルトで、または明示的な形でリクエストに含めることができます。レスポンスグループは、 問い合わせた商品に関して、どのような種類の情報を表示したいかを指定するものです。たとえば、Offer というレスポンスグループは、商品の販売情報（価格および商品在庫状況）を返します。

サーチインデックスは、リクエストを Amazon ならびに Javari のデータベースの特定のカテゴリに限定して実行するのに使用します。 カタログと呼ばれるデータベースには数百万件の商品データが収められています。商品の検索結果が 100,000 件もあったのでは、検索の意味がありません。そのため、利用者の要望に合わせて商品を絞り込む目的から、サーチインデックスなどのリクエストパラメータを使用します。たとえば、「ハリーポッ ター」の書籍と DVD は別々のサーチインデックスに分類されています。サーチインデックスを指定することで、検索の目的に合った情報を返すことができま す。

リクエストの構造は階層化されています。それぞれのリクエストには、エンドポイント、すなわち ecs.amazonaws.jp という Product Advertising API の URL を指定します。とくに指定しない限り、解析の容易な XML 形式でレスポンスを返します。

コード例の概要

通常、オンラインショッピングにおいてカスタマーが最初に行うのは、買いたい商品の検索です。その場合、書籍、DVD、衣料品など、探したい商品に応じた検索パラメータを指定しま す。本ガイドでは、複数のコンピュータ言語で REST リクエストの作成方法を示します。このリクエストは、Amazon ならびに Javari の商品カタログを検索し、コマンド ラインで入力されたキーワードに関連のある商品を選び出すものです。本ガイドでは、レスポンスの処理コードは、返される商品が書籍であることを前提としています。レスポンスを解析すると、商品のタイトル、著者、および価格を表示します。

準備作業

トピック

- AWS Access Key ID の取得
- アソシエイトへの登録
- 必要なツールの入手
- 開発環境の確認

Product Advertising API のリクエストの送信を開始するには、まずいくつかの登録・設定作業を完了しなければなりません。以下の各項では、これらの作業について説明します。

AWS Access Key ID の取得

Product Advertising API リクエストでは、AWS Access Key ID 値を入力しなければなりません。これは、Amazon の配布する半角英数字で構成された文字列で、リクエストの送信元を一意に識別する ID です。本 ガイドで読み始める前に、Product Advertising API の無料アカウントにサインアップして ID を取得しておく必要があります。この ID は、http://aws.amazon.com で取得してください。すでに Web サービスアカウントをお持ちの方は、以下の方法を使ってご自分の AWS Access Key ID を検索できます。

AWS Access Key ID を検索するには

1. AWS のウェブサイト、aws.amazon.com に進みます。
2. 「Your Web Services Account」ボタンにポインタを合わせます。

   ドロップダウンリストが表示されます。

3. 「View Access Key Identifiers」をクリックします。

> **注意**
>
> あ なたの AWS Access Key ID とシークレットアクセスコードがウェブページに表示されます。シークレットアクセスコードは自分だけが知っている秘密のコードです。決してリクエスト に含めて送信することのないようにしてください。開発者が AWS の有料サービスを使用した場合、このコードを使って開発者に料金が請求されます。シーク レットコードに漏洩の疑いがある場合には、ただちに Amazon に連絡してください。

アソシエイトへの登録

ア ソシエイトは、商品購入希望者を自分のウェブサイトから Amazon の各サイトへ誘導することによって Amazon から紹介料を受け取ります。紹介 料を受け取るためにはアソシエイト ID が必要です。この ID は、登録を行ったサイトでのみ有効です。複数のサイトでアソシエイトになりたい場合は、サイト ごとに登録を行う必要があります。

アソシエイトに登録するには

1. http://affiliate.amazon.co.jp/gp/associates/join から、アソシエイトの登録作業を行います。
2. 1.で作成済みのアカウントに http://affiliate.amazon.co.jp/ からログインして、支払情報の登録を行います。

   アソシエイトとしての報酬を Amazon から受け取るには、こちらの登録を済ませておく必要があります。

アソシエイト登録の詳細については､それぞれのサイトに応じて以下のウェブサイトをご覧ください。

- http://affiliate.amazon.co.jp/

必要なツールの入手

Product Advertising API リクエストをアプリケーションに統合する場合、ほぼすべてのプログラミング言語が利用できます。次の表の中から Product Advertising API の実装に使用するツールをクリックしてください。リンクをクリックすると、ツールキットをダウンロードしてインストールするため のウェブサイトに移動します。

言語	API の型	使用ツール
Java	SOAP	• Java 6 以降 *PATH* 環境変数に Java のインストール場所を指定してください。 • Eclipse 3.2 以降 対話型開発環境（IDE）として Eclipse を使用する場合には、バージョ

言語	API の型	使用ツール
		ン 3.2 以降を使用する必要があります。ただし、NetBeans など、他の IDE を使用することもできます。
C#	SOAP	• Microsoft Visual Studio 2005 C# Express Edition • .NET Framework2.0
Perl	REST（HTTP POST 使用時）	次の Perl コード例で使用したモジュールは、CPAN のウェブサイトでダウンロードできます。 • `Digest::HMAC_SHA1` • `MIME::Base64` • `LWP` • `XML::XPath` • `Date::Format`
PHP	REST（HTTP GET 使用時）	PHP のコード例では、PHP5 の基本インストールを使用しています。 PHP の設定は環境により異なるため、サンプルコードの実行にはコマンドラインインターフェースを使用しています。このサンプルコードはウェブサーバーから実行することもできます（本ガイドではその詳細については取り上げていません）。

署名認証を確認するためのツール

全ての Product Advertising API リクエストには署名認証を含めていただく必要があります。署名認証入りのリクエストを簡単に作成するには、Signed Request Helper (http://developer.amazonwebservices.com/connect/entry.jspa?externalID=2609&categoryID=14) をご利用ください。オンライン、またはダウンロード版にてご利用いただけます。

開発環境の確認

この項では、ご利用の開発環境が正しくセットアップされているかどうかを確認する方法について説明します。ダウンロードしたツールキットに対応する項に進んでください。

- Java のセットアップ
- C#のセットアップ
- Perl のセットアップ
- PHP のセットアップ

Java

Product Advertising API のオペレーションは Java で直接実装できます。Java の実装を簡略化するために、Product Advertising API の Java クライアントサイドライブラリを作成して利用することもできます。この項では、Product Advertising API の Java クライアント サイドライブラリを作成する方法について説明します。次の項では、ライブラリを使ってリクエストを作成する方法について説明します。

スタブの作成

Java 6 の wsimport ユーティリティを使用して、*http://ecs.amazonaws.com/AWSECommerceService/AWSECommerceService.wsdl* にある Product Advertising APIWSDL からスタブを作成します。

Product Advertising API のクライアントサイドライブラリのスタブを作成するには

1. スタブを作成したいディレクトリに移動して、「build」ディレクトリと「src」ディレクトリを新規作成します。

   生成されるソースコードはすべて「src」フォルダの中に格納されます。

2. Eclipse 3.2 を使用している場合には、「ラッパー型」のコード生成を禁止するカスタムバインディングを作成します。

   ```
   <jaxws:bindings
   wsdlLocation="http://ecs.amazonaws.com/AWSECommerceService/AWSEC
   ommerceService.wsdl"
   xmlns:jaxws="http://java.sun.com/xml/ns/jaxws">
   <jaxws:enableWrapperStyle>false</jaxws:enableWrapperStyle>
   </jaxws:bindings>
   ```

   これは、Eclipse 3.2 がラッパー型生成コードに対応していないために必要なステップです。ただし、ラッパー型生成コードに対応している IDE（NetBeans など）を使用している場合、このステップは不要です。

3. 次のコマンドを実行します。

```
wsimport -d ./build -s ./src  -p com.ECS.client.jax
http://ecs.amazonaws.com/AWSECommerceService/AWSECommerceService
.wsdl -b jaxws-custom.xml .
```

com.ECS.client.jax というパスでスタブが生成されます。

生成されるファイルの種類

com.ECS.client.jax というパッケージには、数種類のファイルが生成されます。

- AWSECommerceService － Product Advertising API のウェブサービスを識別します。
- AWSECommerceServicePortType － クライアントがリッスンできるポートのタイプを示します。

  このファイルには、クライアントの作成に使用できる Product Advertising API のオペレーションシグニチャの一覧も含まれています。

C#

Product Advertising API を使用するためには、Microsoft Visual Studio をインストールしておく必要があります。

インストールを確認するには

1. Visual Studio 2005 C# Express Edition を開きます。
2. 「ヘルプ」 > 「Microsoft Visual Studio のバージョン情報」をクリックします。

   ダイアログボックスが開き、Microsoft Visual Studio 2005 と.NET Framework（バージョン 2.0）が一覧表示されます。

Visual Studio での SOAP プロキシの作成

作成するアプリケーションには、使用する Product Advertising APIWSDL へのウェブ参照を挿入しておく必要があります。

ウェブ参照を挿入するには

1. 「プロジェクト」メニューから、「Web 参照の追加」を選択します。

   ダイアログボックスが開きます。

2. URL 入力ボックスに Product Advertising APIWSDL の URL を入力します。

   たとえば、「2008-06-28」というような値を入力します。

3. 「移動」をクリックします。

   API がダイアログボックスのメインペインに表示されます。

4. 「参照の追加」をクリックします。

   ソリューションエクスプローラに「Web 参照」というフォルダが新たに追加されます。

これで、プロジェクトのネームスペースを使用して SOAP プロキシを参照できるようになりました。

例）

```
using GettingStartedGuideSample.com.amazonaws.ecs;
```

Perl

以下のコマンドを実行して、必要な Perl モジュールがすべてインストールされているかどうか確認します。

```
perl -MDigest::HMAC_SHA1 -e 1
perl -MMIME::Base64 -e 1
perl -MLWP -e 1
perl -MXML::XPath -e 1
perl -MDate::Format -e 1
```

エラーメッセージが出ていないことを確認します。

PHP

PHP のインストールを確認するには

- コマンドラインインターフェースを使用して次のコマンドを実行します。

  ```
  php -version
  ```

このコマンドを実行するには、PHP のインストールディレクトリがカレントディレクトリであるか、もしくは PATH システム変数に指定されている必要があります。

次のようなレスポンスが返るのを確認してください。

```
PHP 5.1.2 (cli) (built: Jan 11 2006 16:40:00)
Copyright (c) 1997-2006 The PHP Group
Zend Engine v2.1.0, Copyright (c) 1998-2006 Zend Technologies
```

リクエストの作成

トピック

- 初めてのリクエストの送信
- リクエストの各部分について
- Product Advertising API リクエストの実装

リ クエストとは、Product Advertising API に処理を要求することです。たとえば、商品カテゴリーや個別の商品についての情報を返すよう Product Advertising API にリクエストすることができます。商品の写真や、商品に関するカスタマーの感想、あるいは商品の価格などを返すよう Product Advertising API に要求することもできます。Product Advertising API では、このような知りたい情報を、リクエスト という形で REST や SOAP を使用してインターネット経由で送信します。Product Advertising API は、XML ドキュメントの形でリクエス トに応答します。

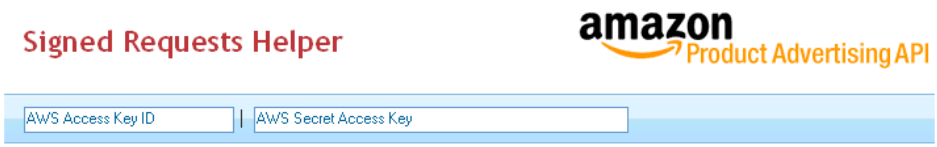
Note

レスポンスの処理については次の項で説明します。

以下の各項では、複数のプログラミング言語による Product Advertising API リクエストの作成方法について説明します。

初めてのリクエストの送信

1. Product Advertising API の雰囲気をつかむために、Signed Request Helper (http://associates-amazon.s3.amazonaws.com/signed-requests/helper/index.html) のページ上部にご自身の AWSAccessKeyId と秘密キーを入力します。



2. 以下の文字列を”Unsigned URL”の項目に入力します。

```
http://ecs.amazonaws.com/onca/xml?Service=AWSECommerceService&Operation=ItemSearch&SearchIndex=
```

```
Marketplace&MarketplaceDomain=www.javari.jp&MerchantId=All&Title
=Emilio%20Pucci&Version=2010-01-01
```

3. ”Display Signed URL”ボタンをクリックした後で”Signed URL”の項目に表示された URL をコピーしてブラウザに貼り付けて、結果を確認してみてください。

アカウント情報とリクエストの署名認証については、「AWS Access Key ID」の項ならびに「リクエストの署名認証について」(https://affiliate.amazon.co.jp/gp/associates/help/t126) を参照してください。

ヒント

URL にはスペースを入れないよう注意してください。スペースは、%20 として URL エンコードされます。パラメータの名前と値は、大文字と小文字を区別します。

おめでとうございます！これで、初めての Product Advertising API リクエストが完成しました。この例から、REST リクエストが URL である ことがわかります。クエスチョンマーク（?）よりも前の部分全体がリクエストの送信先です。この宛先は、どの Product Advertising API リクエ ストについても同じです（つまり、同じサイトへ送信されます）。クエスチョンマーク（?）の後に続く部分は、すべてこのリクエストに含まれるパラメータで す。このリクエストは、タイトルに「Emilio Pucci」という文字列（*Title*=Emilio%20Pucci）を含む、Javari.jp (MarketplaceDomain=www.javari.jp) で販売されている商品（*SearchIndex*=Marketplace）をすべて検索（*Operation*=ItemSearch）します。

ヒント

Product Advertising API には、US（アメリカ）、JP（日本）、FR（フランス）、DE（ドイツ）、UK（イギリス）、CA（カナダ）をはじめとする 数多くのサイトがあります。エンドポイントはサイトによってわずかに異なります。たとえば JP（日本）のエンドポイントは、 http://ecs.amazonaws.jp になります。リクエストは任意のサイトに送信できます。ただし、配送コスト削減などの理由からカスタマー の居住地域のサイトへ送信するのが一般的です。すべてのサイトおよびエンドポイントの一覧については、『Product Advertising API Developer Guide』(http://docs.amazonwebservices.com/AWSECommerceService/latest/DG/) を参照してください。

Product Advertising API は、XML ドキュメントを返すという形でリクエストに応答します。次に示すのは、レスポンスの一部です。

```
<TotalResults>39</TotalResults>
  <TotalPages>4</TotalPages>
  <Item>
    <ASIN> B0039RHLO4</ASIN>
    <DetailPageURL>
http://www.javari.jp/%E3%82%A8%E3%83%9F%E3%83%AA%E3%82%AA%E3%83%97%E3%
83%83%E3%83%81-EMILIO-PUCCI-%E3%83%97%E3%83%AA%E3%83%B3%E3%83%88%E3%83
%93%E3%83%BC%E3%83%81%E3%82%B5%E3%83%B3%E3%83%80%E3%83%AB/dp/B0039RHLO
4%3FSubscriptionId%3D[AWSAccessKeyId]%26tag%3Dws%26linkCode%3Dxm2%26ca
mp%3D2025%26creative%3D165953%26creativeASIN%3DB0039RHLO4
    </DetailPageURL>
    <ItemAttributes>
      <Manufacturer> EMILIO PUCCI(エミリオプッチ)</Manufacturer>
      <ProductGroup>Shoes</ProductGroup>
      <Title> [エミリオプッチ] EMILIO PUCCI プリントビーチサンダル
      </Title>
    </ItemAttributes>
  </Item>
```

この例では、検索条件に一致する商品が 39 件あることを示しています。1 件目の一致商品は『[エミリオプッチ] EMILIO PUCCI プリントビーチサンダル』です。メーカー名、また商品識別コード（ASIN）が B0039RHLO4 であるなど、この商品の詳細情報が返されます。 *DetailPageURL* をブラウザにコピーすると、この商品に関するウェブページが表示されます。

ヒント

ASIN（Amazon Standard Item Number）は、Amazon に出品されている商品を一意に識別する半角英数字で構成された文字列です。

ブ ラウザで URL を送信すると、Product Advertising API のリクエストとレスポンスがどのようにやり取りされているかがよくわかります。ただ しこの方法は、カスタマーアプリケーションにとってはあまり適したものではありません。この項の残りでは、Product Advertising API リクエ ストをプログラミングによって発行する方法について説明します。次の項では、レスポンスをプログラミングで処理する方法を説明します。

リクエストの各部分について

各 プログラミング言語には、独自の書式と要件が規定されています。このため、Product Advertising API へのリクエスト発行の実装には言語間で 多少の相違点があります。ただし、Product Advertising API リクエストを実装する際にプログラミングによって処理する以下のタスクは、すべ てのプログラミング言語に共通です。

プログラミングによって処理するタスク

1	リクエストオブジェクトを作成する。
2	リクエストにパラメータおよびパラメータ値を追加する。
3	リクエストをセットアップする。
4	リクエストを送信する。

以下の各項では、上記の各タスクの実行方法をプログラミング言語別に説明しています。

Product Advertising API リクエストの実装

この項では、さまざまなプログラミング言語による ItemSearch リクエストの実装方法を説明します。なお、コードサンプルに記載されたエンドポイントは、全て US のエンドポイントとなりますので、JP のサービスを呼び出したい場合は、JP 用のエンドポイントに変更の上ご利用ください。

Java

次の Java コードは、*SearchIndex* と *Keywords* の値をカスタマーが指定する ItemSearch リクエストを実装したものです。この例では、リクエストを簡略化するために Java クライアントサイドライブラリを使用しています。wsimport でクライアントサイドライブラリをダウンロードしてスタブを作成するには、「Product Advertising API の Java クライアントサイドライブラリ」の項を参照し、次のコードを使用してください。

```
// Set the service:
com.ECS.client.jax.AWSECommerceService service = new
com.ECS.client.jax.AWSECommerceService();

//Set the service port:
```

```
com.ECS.client.jax.AWSECommerceServicePortType port =
service.getAWSECommerceServicePort();

//Get the operation object:
com.ECS.client.jax.ItemSearchRequest itemRequest = new
com.ECS.client.jax.ItemSearchRequest();

//Fill in the request object:
itemRequest.setSearchIndex("Marketplace");
itemRequest.setMarketplaceDomain("www.javari.jp");
itemRequest.setKeywords("dog");
itemRequest.setVersion("2010-01-01");
com.ECS.client.jax.ItemSearch ItemElement= new
com.ECS.client.jax.ItemSearch();
ItemElement.setAWSAccessKeyId("[YOUR ID]");
ItemElement.getRequest().add(itemRequest);

//Call the Web service operation and store the response
//in the response object:
com.ECS.client.jax.ItemSearchResponse
    response = port.itemSearch(ItemElement);
```

C#

次の C#コードは、*SearchIndex* と *Keywords* の値をカスタマーが指定する ItemSearch リクエストを実装したものです。インラインでコメントが挿入されています。

Note

.NET の「Web 参照の追加...」ダイアログを使用すると、GettingStartedGuideSample.com.amazonaws.ecs パッケージが自動的に生成されます。

```
using System;
using System.Collections.Generic;
using System.Text;
using GettingStartedGuideSample.com.amazonaws.ecs;

namespace GettingStartedGuideSample
{
```

```
class Program
{
  static void Main(string[] args)
  {
    // Set default args if two are not supplied
    if (args.Length != 2)
    {
      args = new string[] { "Marketplace", "Pucci" };
    }

    // Get searchIndex and keywords from the command line
    string searchIndex = args[0];
    string keywords = args[1];

    // Create an instance of the Product Advertising API service
    AWSECommerceService ecs = new AWSECommerceService();

    // Create ItemSearch wrapper
    ItemSearch search = new ItemSearch();
    search.AssociateTag = "[Your Associate ID]";
    search.AWSAccessKeyId = "[Your ID]";
    search.Version = "2010-01-01";
    search.MarketplaceDomain = "www.javari.jp";

    // Create a request object
    ItemSearchRequest request = new ItemSearchRequest();

    // Fill request object with request parameters
    request.ResponseGroup = new string[] { "ItemAttributes" };

    // Set SearchIndex and Keywords
    request.SearchIndex = searchIndex;
    request.Keywords = keywords;

    // Set the request on the search wrapper
    search.Request = new ItemSearchRequest[] { request };
```

```
      try
      {
        //Send the request and store the response
        //in response
        ItemSearchResponse response =
          ecs.ItemSearch(search);
```

Perl

次の Perl コードは、SearchIndex と Keywords の値をカスタマーが指定する Keywords リクエストを実装したものです。インラインでコメントが挿入されています。

```
#!/usr/bin/perl

use strict;
use warnings;
use LWP::UserAgent qw($ua get);
use MIME::Base64;
use XML::XPath;
use Date::Format;

# Retrieve command line args for SearchIndex and Keywords
die "Usage: $0 <space-separated entry for Search Index and Keywords>¥n"
    unless @ARGV;
my $searchIndex = $ARGV[0];
my $keywords = $ARGV[1];

# Define the parameters in the REST request.
# Customer cannot change the following values.
my $EndPoint = "http://ecs.amazonaws.jp/onca/xml";
my $service = "AWSECommerceService";
my $accesskey = "[INSERT YOUR ACCESS KEY ID HERE]";
my $operation = "ItemSearch";
my $marketplacedomain = "MarketplaceDomain";
my $version = "2010-01-01";
```

```
# Assemble the REST request URL.
my $request =
    "$EndPoint?" .
    "Service=$service&" .
    "AWSAccessKeyId=$accesskey&" .
    "Operation=$operation&" .
    "Keywords=$keywords&" .
    "SearchIndex=$searchIndex&" .
    "MarketplaceDomain=$marketplacedomain&" .
    "Version=$version" ;

# Send the request using HTTP GET.
my $ua = new LWP::UserAgent;
$ua->timeout(30);
my $response = $ua->get($request);
my $xml = $response->content;
```

PHP

次の PHP コードは、*SearchIndex* と *Keywords* の値をカスタマーが指定する ItemSearch リクエストを実装したものです。このサンプルコードは、*SimpleStore.php* という名前のファイルに保存します。インラインでコメントが挿入されています。

```
<?php

//Enter your IDs
define("Access_Key_ID", "[Your Access Key ID]");
define("Associate_tag", "[Your Associate Tag ID]");

//Set up the operation in the request
function ItemSearch($SearchIndex, $Keywords){

//Set the values for some of the parameters.
$Operation = "ItemSearch";
$Version = "2010-01-01";
$ResponseGroup = "ItemAttributes,Offers";
//User interface provides values
```

```
//for $SearchIndex and $Keywords

//Define the request
$request=
    "http://ecs.amazonaws.jp/onca/xml"
  . "?Service=AWSECommerceService"
  . "&AssociateTag=" . Associate_tag
  . "&AWSAccessKeyId=" . Access_Key_ID
  . "&Operation=" . $Operation
  . "&Version=" . $Version
  . "&SearchIndex=" . $SearchIndex
  . "&MarketplaceDomain=" . $MarketplaceDomain
  . "&Keywords=" . $Keywords
  . "&ResponseGroup=" . $ResponseGroup;

//Catch the response in the $response object
$response = file_get_contents($request);
$parsed_xml = simplexml_load_string($response);
printSearchResults($parsed_xml, $SearchIndex);
}
?>
```

この実装の最初の部分が ItemSearch リクエストを構成します。リストの先頭部分は、エンドポイント、サービス名、Access Key ID、アソシエイトタグ、Product Advertising API のバージョン番号、およびオペレーション名など、カスタマーによる変更のできないパラメータです。

最後の 2 つのパラメータ（*SearchIndex* および *Keywords*）は、ユーザーインターフェースを介してカスタマーが設定します。*SearchIndex* は、「本」、「自動車」、「アクセサリー」といった商品カテゴリーに似たものです。*Keywords* は単語または語句です。このリクエストは、指定されたサーチインデックスに含まれる商品の中から、タイトルや説明に *Keywords* の値が含まれるものを選び出します。

最後の 2 つのパラメータ値は、カスタマーがウェブアプリケーションを使用して入力します。

例）

```
<table align='left'>
```

```
<?php
    print("
      <form name='SearchTerms' action=SimpleStore.php method='GET'>
      <tr><td valign='top'>
        <b>Choose a Category</b><br>
          <select name='SearchIndex'>
            <option value='All'>All</option>
            <option value='Javari'>Javari</option>
          </select>
      </td></tr>
      <tr><td><b>Enter Keywords</b><br><input type='text' name='Keywords'
size='40'/></td></tr>
      <input type='hidden' name='Action' value='Search'>
      <input type='hidden' name='CartId' value=$CartId>
      <input type='hidden' name='HMAC' value=$HMAC>
      <tr align='center'><td><input type='submit'/></td></tr>
      </form> ");
?>
</table>
```

この例ではテーブルを使用して、ひとつの HTML フォームからなるウェブページを生成しています。HTML の select 文は、*SearchIndex* の候補値をドロップダウンリストで表示します。HTML の input 文は、カスタマーが *Keywords* 値を入力するテキストボックスを表示します。

リクエストは、file_get_contents という PHP コマンドを使用して送信されます。

Product Advertising API のレスポンスの処理

トピック

- リクエストの実行の確認
- 処理の概要
- 処理の実装

Product Advertising API へのリクエスト送信は無事完了しました。次に、レスポンスの受信と処理に進みます。この項では、レスポンスを受信して処理する 方法について説明します。この項のコード例は、前の項で示したコード例のつづきです。たとえば、この項で示す変数名は前の項で示したものと同じ名前になっ ています。

リクエストの実行の確認

リクエストの実行は、まず各レスポンスの *IsValid* 要 素を調べることによって確認できます。この要素が True と返されている場合、リクエストの実行は無事完了しているため、レスポンスに含まれる情報を 表示することができます。しかし、False という値が返された場合、リクエストに構文エラーがあったことを示します。リクエストに返されたエラーを調べれば、そのリ クエストにあるエラーのトラブルシューティングを開始できます。次のエラー文のサンプルは、リクエストに必須パラメータ ItemId が含まれていなかったことを示しています。なお、以下のそれぞれのエラーサンプルは、US のエンドポイントに対してリクエストを送信した際に返されるメッセージの例となります。

```
<IsValid>False</IsValid>
...
<Error>
  <Code>AWS.MissingParameters</Code>
  <Message>Your request is missing required parameters. Required parameters
include ItemId.</Message>
</Error>
```

ただし、*IsValid* 要素は、リクエストの失敗時に必ず返されるというわけではありません。たとえばオペレーション名にタイプミスがあった場合、Product Advertising API は *IsValid* 要素の含まれていない次のようなメッセージを返します。

```
<Error>
```

```
  <Code>AWS.InvalidOperationParameter</Code>
  <Message>The Operation parameter is invalid. Please modify the Operation
parameter and retry. Valid values for the Operation parameter include
ListLookup, CartGet, SellerListingLookup, CustomerContentLookup,
ItemLookup, SimilarityLookup, SellerLookup, ItemSearch, BrowseNodeLookup,
CartModify, ListSearch, CartClear, CustomerContentSearch, CartCreate,
TransactionLookup, CartAdd, SellerListingSearch, Help.
  </Message>
</Error>
```

IsValid の 値が True である場合、リクエストは有効であり、実行されたことを意味します。ただしこれは、結果が得られたという意味ではありません。たとえば、検索 条件を満たす商品がひとつも存在しないという状況も考えられます。このような状況が起きていないかを確認するには、Error 要素の有無を調べるか、ある いは *TotalItems* 要素の値を調べます。この値がゼロである場合、次の例で示すように、表示する結果が 1 件もないということです。

```
<IsValid>True</IsValid> ...
<Error>
  <Code>AWS.ECommerceService.NoExactMatches</Code>
  <Message>リクエストに該当する結果がありません。</Message>
</Error> ...
<TotalResults>0</TotalResults>
<TotalPages>0</TotalPages>
```

Java

エラーは XML レスポンスのさまざまなレベルで発生する可能性があります。次の例は、レスポンスに *OperationRequest* 要 素が含まれているかどうかを調べるものです。このレスポンス要素はすべてのレスポンスに含まれています。これが欠落している場合、レスポンスは null に なります。この状況は、たとえば、Product Advertising API リクエストのウェブサービスがそのリクエストをタイムアウトした場合などに発生 します。第 2 のエラーチェックは、*Items* というレスポンス要素の有無を調べるものです。

```
assertNotNull("OperationRequest is null", operationRequest );
System.out.println("Result Time = " +
operationRequest.getRequestProcessingTime());
```

```
for (Items itemList : response.getItems()) {
  Request requestElement = itemList.getRequest();
  assertNotNull("Request Element is null", requestElement);
```

エラーチェック作業を徹底化するには、すべてのレスポンス要素を評価して各要素が実際に返されているかどうか確認していく必要がありま す。上の例は、そのような評価を実行するコードのテンプレートになります。ただし本ガイドでは、コード例が複雑になるのを避けるためコード全体は記載しま せん。

C#

次のコード例は、リクエストの実行が無事完了したことを確認するものです。このコードは、null レスポンスがないかどうかチェックします。

```
//Verify a successful request
ItemSearchResponse response = service.ItemSearch(itemSearch);

//Check for null response
if (response == null)
  throw new Exception("Server Error - no response recieved!");
ItemSearchResult[] itemsArray = response.GetItemSearchResult;
if (response.OperationRequest.Errors != null)
  throw new Exception(response.OperationRequest.Errors[0].Message);
```

Perl

次のコード例は、リクエストの実行が無事完了したことを確認するものです。このコードは、レスポンスに「Error」要素が含まれていないかどうかチェックします。

```
#See if "Error" is in the response.
if ( $xp->find("//Error") )
{
   print "There was an error processing your request:¥n",
    "  Error code: ", $xp->findvalue("//Error/Code"), "¥n",
    "  ", $xp->findvalue("//Error/Message"), "¥n¥n";
}
```

PHP

次のコード例は、リクエストの実行が無事完了したことを確認するものです。このコードは、XML レスポンスに Error 要素が含まれていないかどうかチェックします。

```
//Verify a successful request
foreach($parsed_xml->OperationRequest->Errors->Error as $error){
   echo "Error code: " . $error->Code . "¥r¥n";
   echo $error->Message . "¥r¥n";
   echo "¥r¥n";
}
```

処理の概要

前の項で述べたとおり、Product Advertising API のレスポンスは XML ドキュメントです。返される要素と各要素の値は、Amazon のデータベースに記録されているデータと、リクエストの中で指定されたレスポンスグループによって異なります。

レスポンスグループは、レスポンスとして返される情報の絞り込みを行います。たとえば、*Images* というレスポンスグループを指定すると、商品の画像情報がレスポンスとして返されます。TopSellers というレスポンスグループを指定すると、サーチインデックスにあるベストセラー商品が返されます。『*Product Advertising API Developer Guide*』(http://docs.amazonwebservices.com/AWSECommerceService/latest/DG/)では、レスポンスグループの指定によって絞り込めるすべての要素が一覧表示されています。

レスポンスグループによって返される要素は階層化されています。たとえば、次のレスポンス例に示す *Item* 要素にはいくつかの子要素があり、そのひとつが *Offers* 要素であり、この要素も *Subtotal* という子要素をもち、さらにこの子要素自体も *Amount*、*CurrencyCode*、*FormattedPrice* という 3 つの子要素をもっています。

```
<Item>
  <Offers>
    <Subtotal>
      <Amount>999</Amount>
      <CurrencyCode>US</CurrencyCode>
      <FormattedPrice>$9.99</FormattedPrice>
```

階層化した要素の親子関係を XPath と呼びます。結果を解析するときには Product Advertising API のレスポンスをオブジェクトに変換するため、XPath を使用すれば、要素や各要素の値を効率的に探し出せます。

次の PHP コード例では、最初に *FormattedPrice* 値の有無を確認したあとでその値を表示するために XPath を使用しています。

```
if(isset($current->Item->Offers->Subtotal->FormattedPrice)){   print("<
br>Price:"
    $current->Offers->Subtotal->FormattedPrice);
```

同じ処理を C#で書くと、次のようになります。

```
Label1.Text += "Price: " +
  Item.Offers.Subtotal.FormattedPrice + "<br />";
```

使用する XPath を変えるだけで、他の値を返すことができます。

通常、ひとつのレスポンスで複数の商品が返されてきます。このため、レスポンスで返されるすべての商品について解析アルゴリズムを繰り返す必要があります。たとえば、PHP で記述すると、次のようになります。

```
foreach($parsed_xml->Items->Item as $current){...}
```

C#では次のようになります。

```
foreach(Item item in response){...}
```

処理の実装

以下の各項では、いくつかのプログラミング言語を例にとって、これらの解析原理が一層厳格な形で守られていることを示します。

Java

前 の項では、リクエストでレスポンスオブジェクトを取得しました。Product Advertising API のクライアントサイドライブラリには、そのオブ ジェクトからさまざまな値を返すメソッドが含まれています。次のコードは、レスポンスオブジェクトから、items、item、item attributes、および title という要素の値を取得します。

```
// Get the Title names of all the books for all the items returned in the
response
    for (Items itemList : response.getItems()) {
        for (Item item : itemList.getItem()){
            System.out.println("Item Name: " +
            item.getItemAttributes().getTitle());
        }
    }
```

C#

次の C#コードは、Product Advertising API が返すレスポンスを処理します。このコードは C#リクエストサンプルコードのつづきです。

```
        //Go through the response and display the
        //title, author, and price
        foreach (Items items in response.Items)
        {
          foreach (Item item in items.Item)
          {
            //Output the results to the console
            Console.WriteLine(
              "Title: " + item.ItemAttributes.Title + "¥n"                  +
              "Author: " + item.ItemAttributes.Manufacturer + "¥n"
+
              "Price: " + item.ItemAttributes.ListPrice.FormattedPrice +
"¥n"
              );
          }
        }
```

```
      }
      //Catch and display any exceptions
      catch (Exception ex)
      {
        Console.WriteLine("An error occured: " + ex.ToString());
      }

      Console.ReadLine();
    }
  }
}
```

このコードでは、レスポンスに含まれる全商品について処理を繰り返すために for 文を使用しています。title、manufacturer、および price 要素の値は、Console.WriteLine を使って表示します。

Perl

次の Perl コードは、Product Advertising API が返すレスポンスを処理します。このコードは Perl リクエストサンプルコードのつづきです。

```
# Process XML response using XPath (xp)
my $xp = XML::XPath->new(xml => $response);

# Iterate through the items in the response
{
  for (my $i = 1; $i <= 10; $i++)
  {
  if ( ! $xp->find("/ItemSearchResponse/Items/Item[$i]") )
  {
    last;
  }

# Find manufacturer names
  my @manufacturer;
  for (my $j = 1;
```

```
    $j <=
$xp->findvalue("count(/ItemSearchResponse/Items/Item[$i]/ItemAttribute
s/Manufacturer)");
    $j++)
  {
   push @manufacturer,
$xp->findvalue("/ItemSearchResponse/Items/Item[$i]/ItemAttributes/Manu
facturer[$j]");
  }

# Find titles, prices, and display them with the manufacturers
  print "Title: ",
$xp->findvalue("/ItemSearchResponse/Items/Item[$i]/ItemAttributes/Titl
e"), "¥n",
  "Manufacturer: ", join(", ", @manufacturers), "¥n",
  "Price: ",
$xp->findvalue("/ItemSearchResponse/Items/Item[$i]/Offers/Offer/OfferL
isting/Price/FormattedPrice"), "¥n¥n";
  }
}
```

このコードでは、レスポンスに含まれる全商品について処理を繰り返すために for 文を使用しています。*title*、*manufacturer*、および *price* 要素の値は、print を使って表示します。

PHP

次の PHP コードは、Product Advertising API が返すレスポンスを処理します。このコードは PHP リクエストサンプルコードのつづきです。

```
<?php
function printSearchResults($parsed_xml, $SearchIndex){
  print("<table>");
  if($numOfItems>0){
  foreach($parsed_xml->Items->Item as $current){
```

```
    print("<td><font
size='-1'><b>".$current->ItemAttributes->Title."</b>");
    if (isset($current->ItemAttributes->Title)) {
    print("<br>Title: ".$current->ItemAttributes->Title);
  } elseif(isset($current->ItemAttributes->Manufacturer)) {
    print("<br>Manufacturer: ".$current->ItemAttributes->Manufacturer);
  } elseif
   (isset($current->Offers->Offer->Price->FormattedPrice)){
    print("<br>Price:
    ".$current->Offers->Offer->Price->FormattedPrice);
  }else{
  print("<center>No matches found.</center>");
   }
  }
 }
}
?>
```

Product Advertising API のレスポンスは、simplexml_load_string という PHP コマンドで *$parsed_xml* というオブジェクトに変換されます。このレスポンスは、次のように定義された printSearchResults という関数を使って表示されます。

このコードは、まずレスポンスに商品が含まれているかどうかをチェックします。

```
$numOfItems = $parsed_xml->Items->TotalResults;
if($numOfItems>0){
  ...
  }else{
  print("<center>No matches found.</center>");
```

商品属性の値は、レスポンス要素の XPath（Items->TotalResults）を使って見つけます。

レスポンスにより返される商品ごとにこのコードの処理を繰り返すことによって、結果オブジェクト *$parsed_xml* を解析します。Items->Item という Xpath を *$current* にセットします。

```
foreach($parsed_xml->Items->Item as $current){
```

$current を使用して、レスポンスに含まれるすべての商品属性にアクセスします。たとえば、次のコードはタイトルを表示します。

```
print("<td><font
size='-1'><b>".$current->ItemAttributes->Title."</b>");
```

このコードは、レスポンスの中にある商品属性を表示するだけです。

```
if(isset($current->ItemAttributes->Manufacturer)){
  print("<br>Manufacturer: ".$current->ItemAttributes->Manufacturer);
```

print 文にドットを付けると、それよりも前に列記されているすべての属性に *Manufacturer* 属性が連結されます。

次のステップ、ならびにサポート対象のオペレーション

トピック

- 商品の詳細情報の提供
- ItemSearch
- ItemLookup
- 共通のリクエストパラメータ

おめでとうございます！これで本ガイドにある基本例の学習はすべて完了しました。いつでも自分でアプリケーションのデザインを開始できます。Product Advertising API を利用するアプリケーションの大部分は、本ガイドのサンプルほど単純なものではありませんが、アプリケーションが複雑化しても、 サンプルで示した原理を簡単に応用してアプリケーションを作成することができます。

Product Advertising API は、革新的なアプリケーションやウェブサイトを生み出す豊かな可能性を秘めています。これまでの項では、ItemSearch を 使用して商品を検索する方法について説明しました。多くの場合、Product Advertising API アプリケーションで最初に実装されるタスク は、商品の検索です。この項では、Product Advertising API アプリケーションを使用するカスタマーにとって一般的な利用事例を想定し、Javari においてサポートされているオペレーションについて詳しく説明しながら、タスクの実行方法を説明します。

商品の詳細情報の提供

これまでの節で説明および実装した ItemSearch リ クエストは、多くの場合、複数の商品を返します。一般的に、Product Advertising API アプリケーションは、商品の小さな画像を簡潔な説明と ともに表示します。ただし、カスタマーの側から見ると、商品リストの中から興味ある商品をひとつまたは複数選び、それらの商品の詳細をチェックしたい場合 が多いでしょう。
ItemSearch が返した商品のすべてについて詳細な情報を表示することは可能ですが、その場合ウェブページが長大化してしまいます。このため、カスタマーが興味を示した商品についてのみ詳細情報を表示する方法をお勧めします。

ItemSearch が返した商品 ID に基づいて、任意の表示商品についての詳細を ItemLookup オペレーションによって表示することができます。ItemLookup は、価格情報とともに商品の詳細な情報をすべて返すことができます。

2010 年 7 月現在、Javari.jp 向け Product Advertising API でサポートされているオペレーションは、ItemSearch および ItemLookup となります。通常版 Product Advertising API においてサポートされているその他の機能については、現在ご利用いただけませんのでご注意ください。

ItemSearch

説明

ItemSearch オペレーションでは、1 つまたは複数のサーチインデックスに対して、検索条件を満たす商品の検索結果を、一度に最大で 10 個返します。

ItemSearch では、サーチインデックスの他に、1 つまたは複数のパラメータ値を指定する必要があります。追加パラメータ値で、指定したサーチインデックス内の商品を参照する必要もあります。例えば、Javari 商品を検索する際に使用する Marketplace のブラウズノード (BrowseNode は、 ItemSearch パラメータのひとつです) である、サンダル(53273051)を指定することが可能です。仮にサーチインデックスに Automotive を 指定し、ブラウズノードにサンダル (53273051)を指定しても、何も返されません。この場合、パラメータ値がサーチインデックスの値と正しく紐づいていないため、このような結果となります。

ItemPage パラメータを使用すると、指定した結果ページを返すことができます。指定できる *ItemPage* の最大値は 400 です。これよりも大きなページにアクセスしようとするとエラーが返されます。リクエストに *ItemPage* を指定しない場合は、デフォルトで最初のページが返されます。各ページには最大 10 個の商品が表示されます。

ItemSearch は、リクエストで最もよく使われるオペレーションです。通常、出品されている商品を検索する場合はこのオペレーションを使用します。

リクエストパラメータ

ItemSearch にはたくさんのパラメータがあります。ただし、全てのパラメータが全てのサーチインデックスで使用できるわけではありません。例えば、サーチインデックスが Apparel の場合、*Actor* パラメータを使用するのは適切ではありません。つまり、どのサーチインデックスも一部のパラメータしか使用できないということになります。

最も多くのサーチインデックスに適用されるパラメータを次の表に示します。

パラメータ	有効なサーチインデックス
BrowseNode	All、Blended を除く全て
Condition	All、Blended、Merchants を除く全て
Keywords	全て
MaximumPrice	All、Blended、Merchants を除く全て
MerchantId	All、Blended、Merchants を除く全て

パラメータ	有効なサーチインデックス
MinimumPrice	All、Blended、Merchants を除く全て
Title	All、Blended、Merchants を除く全て

ItemSearch では、サーチインデックスと、次のパラメータを 1 つまたは複数指定する必要があります。 以下は、Javari 商品の検索において有効なパラメータのリストとなります。

- Brand
- BrowseNode
- Keywords
- Manufacturer
- Title

名前	説明	必須
Brand	商品に関連したブランドの名前。名前の全部または一部を入力できます。 タイプ: 文字列（例: Adidas, Crocs, Nike） デフォルト: なし	いいえ
BrowseNode	ブラウズノードは、商品カタログを識別する正の整数です。例えば、サンダルは 53273051、スニーカーは 53276051、パンプスは 53271051 となります。 タイプ: 文字列 デフォルト: なし 有効な値: 正の整数	いいえ
ItemPage	レスポンスの全ての商品から特定のページの商品を取得します。1 ページには最大 10 個の商品が返されますが、*Condition* が "All" の場合のみ例外となります。この場合、 ItemSearch は、*Condition* ごとに最大で 3 件の結果を返します。例えば、新品/中古商品/新品同様の商品/コレクター商品をそれぞれ 3 件ずつ返します。また、コレクター商品や新品同様の商品に関する情報がない場合は、 ItemSearch は新品と中古品の出品情報をそれぞれ 3 件ずつ返します。指定できる *ItemPage* の最大値は 400 です。これよりも大きなページにアクセスしようとするとエラーが返されます。リクエストに *ItemPage* を指定しない場合は、デフォルトで最初のページが返されます。見つかった商品の合計ページ数は *TotalPages* レスポンスタグで返されます。 有効な値: 1～400 までの整数 タイプ: 文字列 デフォルト: なし	いいえ

名前	説明	必須
Keywords	商品に関連する単語または語句。これらの単語や語句は、商品に関するさまざまなフィールド（商品名、メーカー名、ブランド名など）で使用できます。語句を入力する場合、スペースを %20 に URL エンコードしてください。 タイプ: 文字列 デフォルト: なし	いいえ
Manufacturer	商品に関連したメーカーの名前。名前の全部または一部を入力できます。 タイプ: 文字列 デフォルト: なし	いいえ
MaximumPrice	商品の最高価格をレスポンスに指定します。価格は、最低通貨単位を基準としています。JP サイトの場合、3241 は 3241 円を表します。 タイプ: 文字列 デフォルト: なし 有効な値: 正の整数	いいえ
MerchantId	商品を出品しているマーチャントを指定します。*MerchantId* は Amazon がマーチャントに割り当てた半角英数字の ID です。 【注意】MerchantId のデフォルト値は "Amazon" になっているため、Javari 商品のリクエストを行う場合は、必ず値を "All" または、Javari のマーチャント ID である "A2EVFYIVZSJ3AR" に設定してください。 タイプ: 文字列 デフォルト: Amazon 有効な値: All--Amazon とその他全てのマーチャント | A2EVFYIVZSJ3AR—Javari.jp のマーチャント ID	はい
MinimumPrice	商品の最低価格をレスポンスに指定します。価格は、最低通貨単位を基準としています。JP サイトの場合、3241 は 3241 円を表します。 タイプ: 文字列 デフォルト: なし 有効な値: 正の整数	いいえ
ReviewSort	パラメータに基づいてレビューを並べ替えます。 タイプ: 文字列 デフォルト: なし 有効な値: -HelpfulVotes, HelpfulVotes, -OverallRating, OverallRating, Rank,	いいえ

名前	説明	必須
	-Rank, -SubmissionDate, SubmissionDate	
SearchIndex	検索対象の商品カテゴリ。多くの ItemSearch パラメータは、特定の *SearchIndex* でのみ有効です。 タイプ: 文字列 デフォルト: なし 有効な値: サーチインデックス名。Javari 商品の検索では、Marketplace のみが有効です。それ以外の値を指定した場合、Amazon 商品が結果として返されます。	いいえ
Sort	レスポンス内の商品を並べ替える方法。 タイプ: 文字列 デフォルト: なし 有効な値: Marketplace サーチインデックスの場合、以下の値。 ● -launch-date (発売日: 新しい日付～古い日付) ● price (価格: 低～高) ● -price (価格: 高～低) ● salesrank (売れている順番) ● titlerank (アルファベット順: A～Z) ● -titlerank (アルファベット順: Z～A)	いいえ
Title	商品名。商品名の全部または一部を入力できます。*Title* 検索は、*Keyword* 検索のうちの 1 つです。もし *Title* 検索によって十分な結果が得られない場合は、*Keywords* 検索をお試しください。 タイプ: 文字列 デフォルト: なし	いいえ
VariationPage	ItemSearch によって返されるバリエーションの特定ページを取得します。デフォルトでは, ItemSearch は全てのバリエーションを返します。スポンスのサブセクションを返すには、*VariationPage* を使用します。各ページに 10 個のバリエーションがあります。例えば、11～ 20 の出品情報を調べるには、*VariationPage* に 2 を設定します。見つかった商品の合計ページ数は *TotalPages* 要素で返されます。 タイプ: 文字列 デフォルト: なし 有効な値: 正の整数	いいえ
ResponseGroup	返される値の種類を指定します。1 つのリクエストに複数のレスポンスグル	い

名前	説明	必須
	ープをカンマで区切って指定できます。 タイプ: 文字列 デフォルト: Small 有効な値: BrowseNodes | EditorialReview | ItemAttributes | ItemIds | Large | Medium | OfferFull | Offers | OfferSummary | Reviews | SearchBins | Similarities | Variations | VariationSummary |	いえ

ItemSearch には、全てのオペレーションで使用できるパラメータを指定できます。詳細については、共通のリクエストパラメータをご参照ください。

レスポンス

名前	説明
ASIN	Amazon Standard Identification Number の略。Amazon によって割り当てられた、商品を一意に識別する半角英数字の文字列です。
DetailPageURL	商品の詳細ページの URL。商品名、在庫状況、類似品、機能、アクセサリ、商品の説明、商品のカスタマーレビュー、商品に関するニュース記事へのリンク、関 連するリストマニアのリスト、"So You'd Like To...." リスト、"Browse For Photo" リストが指定されます。
/Item	ASIN、DetailPageURL、ItemAttributes を含む、商品情報のコンテナ。
/ItemAttributes	Manufacturer、ProductGroup、Title を含む、商品についての情報のコンテナ。
Manufacturer	商品の製造元・メーカー名。
ProductGroup	商品カテゴリ。Javari 商品の場合、値は"Shoes"のみです。
Title	商品のタイトル。
TotalPages	レスポンスのページの合計数。各ページには 10 個の商品が表示されます。
TotalResults	商品の総数。

例

サーチインデックス Marketplace と *Keywords* パラメータを使用して、Javari に出品されているすべてのパンプス（Keywords=パンプス）に関する情報を返してみます。

```
http://ecs.amazonaws.com/onca/xml?
Service=AWSECommerceService&
AWSAccessKeyId=[AWS アクセスキー ID]&
Operation=ItemSearch&
Keywords=%E3%83%91%E3%83%B3%E3%83%97%E3%82%B9&
SearchIndex=Marketplace&
```

```
MarketplaceDomain=www.javari.jp&
MerchantId=All&
```

このリクエストに対するレスポンスを、以下のレスポンス例に示します。
商品のブラウズノードを検索するには、BrowseNodes レスポンスグループを使用します。

レスポンス例
次に示すのは、上の例の最初のリクエストで返されるレスポンスの一部です。

```
<TotalResults>6132</TotalResults>
<TotalPages>614</TotalPages>
<Item>
  <ASIN>B0014D6FAQ</ASIN>
  <DetailPageURL>
http://www.javari.jp/section-dor-compo-%E3%82%BB%E3%82%AF%E3%82%B7%E3%
83%A7%E3%83%B3%E3%83%89%E3%83%BC%E3%83%AB-%E3%83%91%E3%83%B3%E3%83%97%
E3%82%B9/dp/B0014D6FAQ%3FSubscriptionId%3D[AWS アクセスキー
ID]%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26cr
eativeASIN%3DB0014D6FAQ</DetailPageURL>
  <ItemAttributes>
    <Manufacturer>section d'or compo(セクションドール)</Manufacturer>
    <ProductGroup>Shoes</ProductGroup>
    <Title>[セクションドール] パンプス 3005</Title>
  </ItemAttributes>
  </Item>
  <Item>
  <ASIN>B0034KZ85A</ASIN>

<DetailPageURL>http://www.javari.jp/MELMO-%E3%83%A1%E3%83%AB%E3%83%A2-
6834-Melmo-%E3%82%AA%E3%83%BC%E3%83%97%E3%83%B3%E3%83%88%E3%82%A5%E3%8
3%91%E3%83%B3%E3%83%97%E3%82%B9/dp/B0034KZ85A%3FSubscriptionId%3D[AWS
アクセスキー
ID]%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26cr
eativeASIN%3DB0034KZ85A</DetailPageURL>
  <ItemAttributes>
  <Manufacturer>Melmo(メルモ)</Manufacturer>
```

```
  <ProductGroup>Shoes</ProductGroup>
  <Title>[メルモ] Melmo オープントゥパンプス</Title>
  </ItemAttributes>
</Item>
```

TotalResults および *TotalPages* タグは、見つかった商品数と、それらの商品のあるページ数を示します。表示する結果ページを選択するには、*TotalPages* を、*ReviewsPage* などの任意のページパラメータと一緒に使用します。通常、各ページに 10 件の結果が表示されます。

ItemLookup

説明

商品 ID を指定すると、 ItemLookup オペレーションは、リクエストで指定したレスポンスグループに従って、一部または全ての商品属性を返します。デフォルトでは、 ItemLookup は商品の *ASIN*, *DetailPageURL*, *Manufacturer*, *ProductGroup* および *Title* を返します。

ItemLookup は、多数のレスポンスグループをサポートしています。このため、商品レビュー、バリエーション、類似品、価格、在庫状況、商品画像など、商品属性と呼ばれるさまざまな情報を取得できます。

複数の商品を一度に検索するには、商品 ID をカンマで区切ります。

リクエストパラメータ

名前	説明	必須
Condition	Amazon 商品を指定する場合、商品のコンディションを指定するパラメータとして使用されています。 Condition が "All" に設定されている場合、Condition の有効な値に対して個々のレスポンスセットが返されます。デフォルト値は "New" です（ "All" ではありません）。 Javari 商品を指定する場合、デフォルト値の“New”以外の値を入力しても結果が返されません。これは、Javari では"New" の商品だけを販売しているためです。 タイプ: 文字列 デフォルト: New 有効な値 (Amazon 商品を指定した場合のみ): Used | Collectible |	いいえ

名前	説明	必須
	Refurbished | All	
IdType	商品の検索に使う商品 ID の種類。ASIN を除く全ての *IdType* では、併せて *SearchIndex* を指定する必要があります。 【注意】Javari 商品を指定する場合、IdType に EAN を使用する際は SearchIndex に All を指定する必要があります。それ以外の値を指定しても、正しい結果が返されませんのでご注意ください。 タイプ: 文字列 デフォルト: ASIN 有効な値: ASIN | EAN (UPC は JP ではサポート外)	いいえ
ItemId	商品を一意に識別する 1〜10 の整数。数値の意味は *IdType* で指定します。 つまり、*IdType* が ASIN の場合、*ItemId* の値は ASIN になります。もし *ItemId* が ASIN の場合、サーチインデックスをリクエストで指定することはできません。 タイプ: 文字列 デフォルト: なし 有効な値: 有効な商品 ID。カンマで区切ったリストを使用することで、最大 10 まで ID を指定できます。	はい
MarketplaceDomain	マーケットプレイスのドメインを指定します。 【注意】Javari 商品のリクエストを行う場合は、必ず”www.javari.jp”を指定してください。指定しない場合、デフォルト値である Amazon の結果が返されます。 タイプ: 文字列 デフォルト: www.amazon.co.jp 有効な値: www.amazon.co.jp, www.javari.jp (JP の場合)	はい
MerchantId	商品を出品しているマーチャントを指定します。*MerchantId* は Amazon がマーチャントに割り当てた半角英数字の ID です。 【注意】MerchantId のデフォルト値は "Amazon" になっているため、	はい

名前	説明	必須
	Javari 商品のリクエストを行う場合は、必ず値を "All" または、Javari のマーチャント ID である "A2EVFYIVZSJ3AR" に設定してください。 タイプ: 文字列 デフォルト: Amazon 有効な値: All--Amazon とその他全てのマーチャント | A2EVFYIVZSJ3AR—Javari.jp のマーチャント ID	
OfferPage	ItemLookup によって返される出品情報のページ。各ページに 10 点の出品商品があります。例えば、11～20 の出品情報を調べるには、*OfferPage* に 2 を設定します。 タイプ: 文字列 デフォルト: 1 有効な値: 1～ 100 までの整数	いいえ
ReviewPage	ItemLookup によって返されるレビューのページ番号。各ページに 5 件のレビューが表示されます。例えば、6～10 番目のレビューを読むには、*ReviewPage* に 2 を設定します。 タイプ: 文字列 デフォルト: 1 有効な値: 1～ 20 までの整数	いいえ
ReviewSort	返されるレビューの並べ替え順序を指定します。 タイプ: 文字列 デフォルト: -SubmissionDate 有効な値: -HelpfulVotes | HelpfulVotes | -OverallRating | OverallRating | SubmissionDate	いいえ
SearchIndex	検索対象の商品カテゴリ。 タイプ: 文字列 デフォルト: なし 有効な値: サーチインデックス名。Javari 商品において有効な値は、All, Marketplace です。 制約: *ItemId* が ASIN の場合、サーチインデックスをリクエストで指定することはできません。ItemId が ASIN 以外の場合は必須となります。	条件付き
VariationPage	ItemLookup によって返されるバリエーションのページ番号。 デフォルトでは、 ItemLookup は全てのバリエーションを返します。 レスポン	いい

名前	説明	必須
	スのサブセクションを返すには *VariationPage* を使用します。各ページに 10 個のバリエーションがあります。例えば、11〜20 の出品情報を調べるには、*VariationPage* に 2 を設定します。 タイプ: 文字列 デフォルト: All 有効な値: 1〜 150 までの整数	え
ResponseGroup	返される値の種類を指定します。1 つのリクエストに複数のレスポンスグループをカンマで区切って指定できます。 有効な値: BrowseNodes | EditorialReview | Images | ItemAttributes | ItemIds | Large | Medium | OfferFull | Offers | OfferSummary| Reviews | SalesRank | Similarities | Small | VariationImages | Variations | VariationSummary	いいえ

ItemLookup には、全てのオペレーションで使用できるパラメータを指定できます。詳細については、共通のリクエストパラメータをご参照ください。

レスポンス

名前	説明
ASIN	Amazon Standard Identification Number の略。Amazon/Javari によって割り当てられた、商品を一意に識別する半角英数字の文字列です。
Item	*DetailPageURL* および *ASIN*, *Title*, *ProductGroup*, および、*Manufacturer* を含む、商品についての情報のコンテナ。
ItemAttributes	商品に関する情報のコンテナ。*Title*, *ProductGroup*. および *Manufacturer* を含む、商品についての情報のコンテナ。
Items	1 つまたは複数の商品のコンテナ。
Manufacturer	商品を製造元・メーカー名。
ProductGroup	商品のカテゴリ。Javari 商品の場合、値は”Shoes”のみです。
Title	商品名。

例

次のリクエストは、*ItemId* B00008OE6I に関する情報を返します。

```
http://ecs.amazonaws.com/onca/xml?
Service=AWSECommerceService&
AWSAccessKeyId=[AWS アクセスキー ID]&
```

```
Operation=ItemLookup&
ItemId= B001AZJVSQ&
MarketplaceDomain=www.javari.jp&
MerchantId=All
```

このリクエストに対するレスポンスを、以下のレスポンス例に示します。

次のリクエストでは、*ItemId* が *EAN* であるため、*SearchIndex* および *ItemType*.も指定する必要があります。 例として、EAN に” 0883503091016”という値を入れています。

【注意】EAN による ItemLookup では、一部正しく返らない商品がありますので、予めご了承ください。

```
http://ecs.amazonaws.com/onca/xml?
Service=AWSECommerceService&
AWSAccessKeyId=[AWS アクセスキー ID]&
Operation=ItemLookup&
ItemId=0883503091016&
SearchIndex=All&
IdType=EAN&
MarketplaceDomain=www.javari.jp&
MerchantId=All
```

商品のブラウズノードを検索するには、BrowseNodes レスポンスグループを使用します。
商品のバリエーション情報を検索するには、Variations レスポンスグループを使用します。

レスポンス例
次のコードは最初のリクエストに対するレスポンスの一部です。デフォルトで返される全ての商品属性が示されています。

```
<Items>
  <Request>
  <IsValid>True</IsValid>
  <ItemLookupRequest>
    <MerchantId>All</MerchantId >
    <ItemId> B001Q3L9K0</ItemId>
  </ItemLookupRequest>
```

```
  </Request>
  <Item>
    <ASIN> B001Q3L9K0</ASIN>
<DetailPageURL>
http://www.javari.jp/%E3%82%B3%E3%83%AD%E3%83%BC%E3%83%AC%E3%83%9C%E3%
83%AB%E3%82%B5-Business-SC-001-zip-N-Black/dp/B001Q3L9K0%3FSubscriptio
nId%3D[AWS アクセスキー
ID]%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26cr
eativeASIN%3DB001Q3L9K0</DetailPageURL>
    <ItemAttributes>
    <Manufacturer>colore borsa(コローレボルサ)</Manufacturer>
    <ProductGroup>Shoes</ProductGroup>
    <Title>[コローレボルサ] Business tote bag SC-001 -zip-N Black</Title>
   </ItemAttributes>
  </Item>
</Items>
```

共通のリクエストパラメータ

次の表は、Product Advertising API の全てのオペレーションで使用できるリクエストパラメータを示しています。

パラメータ	定義	必須
AssociateTag	アソシエイトを一意に識別する半角英数字の文字列。この文字列は、Amazon が商品の販売に対する紹介料を加算するアソシエイトを識別するのに使われます。アソシエイトを識別せずにリクエストを実行しても、紹介料は支払われません。AssociateTag を CartCreate リクエストに指定した場合、CartCreate で返される PurchaseURL に AssociateTag の値が自動的に指定されます。 アソシエイトタグを取得するには、http://affiliate.amazon.co.jp をご参照ください。 有効な値: アソシエイトを一意に識別するために Amazon によって割り当てられた半角英数字の文字列。カスタマーが購入した場合に紹介料を受け取れるようにするには、全てのリクエストでこの値を指定します。	いいえ
AWSAccessKeyId	リクエストの送信者を一意に識別する半角英数字の文字列。*AWSAccessKeyId*	は

パラメータ	定義	必須
	を取得するには、http://aws.amazon.com をご参照ください。 有効な値: Amazon によって割り当てられたアクセスキー ID	い
ContentType	レスポンスのコンテンツ形式を指定します。通常、*ContentType* は、RES T リクエストで、Style パラメータを XSLT スタイルシートに設定した場合にだけ変更します。*例えば、Product Advertising API のレスポンスを HTML 形式に変換するには、ContentType* を text/html に設定します。詳細については、*Style* をご参照ください。 デフォルト: text/xml 有効な値: text/xml, text html	いいえ
MerchantId	商品を出品しているマーチャントを指定します。*MerchantId* は Amazon がマーチャントに割り当てた半角英数字の ID です。 【注意】MerchantId のデフォルト値は "Amazon" になっているため、Javari 商品のリクエストを行う場合は、必ず値を "All" または、Javari のマーチャント ID である "A2EVFYIVZSJ3AR" に設定してください。 タイプ: 文字列 デフォルト: Amazon • 有効な値: All--Amazon とその他全てのマーチャント | A2EVFYIVZSJ3AR--Javari.jp のマーチャント ID	いいえ
Operation	実行する Product Advertising API のオペレーションを指定します。 有効な値: ItemLookup および ItemSearch（Javari. jp の場合）	はい
Service	Product Advertising API のサービスを指定します。Product Advertising API の全てのオペレーションで、値は 1 つしかありません。 有効な値: AWSECommerceService	はい
Style	Product Advertising API のレスポンスで返されるデータの形式を制御します。*Style* は、 REST リクエストだけに関係します。このパラメータに "XML"（デフォルト）を設定すると、純粋な XML レスポンスが生成されま	いいえ

パラメータ	定義	必須
	す。このパラメータに XSLT スタイルシートの URL を指定すると、 XML レスポンスが変換されます。詳細については、以下をご参照ください。 *ContentType* デフォルト: XML 有効な値: XSLT スタイルシートの URL 。	
Validate	オペレーションを実行しないようにします。実際にリクエストを実行することなくテストを行うには、*Validate* パラメータを True に設定します。リクエストに *Validate* を 設定する場合、値を True にする必要があります。デフォルト値は False です。リクエストを実際に実行しないと（Validate=True)、リクエストに対する一部のエラーしか返されません。これは、リクエスト の実行時にしか発生しないエラー（例えば no_exact_matches）があるからです。 デフォルト: False 有効な値: True | False	いいえ
Version	使用する Product Advertising API ソフトウェアと WSDL のバージョンです。デフォルトでは、バージョン 2005-10-05 が使用されます。バージョンを指定する場合は、2009-11-02 などのバージョンを指定します。有効なバージョン番号の一覧については、Product Advertising API のリリースノートを参照してください。最新版の Product Advertising API は、デフォルトでは使われないので注意してください。 デフォルト: 2005-10-05 有効な値: 有効な WSDL バージョンの日付。例えば、2009-11-02	いいえ
XMLEscaping	レスポンスをシングルパスとダブルパスのどちらで XML エンコードするかを指定します。デフォルトでは, *XMLEscaping* は Single で、Product Advertising API のレスポンスは XML 内で 1 回だけエンコードされます。例えば、レスポンスデータにアンパサンド文字 (&) が含まれている場合、その文字は通常の XML エンコード (&) で返されます。もし、*XMLEscaping*	いいえ

パラメータ	定義	必須
	が Double の場合、同じアンパサンド文字が 2 回 XML エンコードされます（&）。この *XMLEscaping* の値 Double は、XML 要素内のテキストをデコードしない、PHP などのクライアントで便利です。 デフォルト: Single 有効な値: Single, Double	

ドキュメントの表記

ここでは、AWS の技術ドキュメントに適用される表記規則について説明します。

文字の表記

以下の表は、文字の表記規則をまとめたものです。

表記	説明／例
コールアウト	コールアウトは、本文のテキスト内に挿入したイラストによる数字です。コールアウトの付いたテキストは、ドキュメント内の別の場所でさらに詳しく説明されています。 このリソースは定期的に使用できます。 
テキスト内のコード	インラインコードサンプル（XML を含む）やコマンドは特殊なフォントで表記します。 `java -version` コマンドを使用できます。

コードブロック	サンプルコードのブロックは、本文との区別のため、全体を別のスタイルで表記します。
	
	`# ls -l /var/www/html/index.html`
	`-rw-rw-r--  1 root root 1872 Jun 21 09:33`
	`/var/www/html/index.html`
	`# date`
	`Wed Jun 21 09:33:42 EDT 2006`

強調	特別な意味を持つ語句や重要な語句は、特殊なフォントで表記します。 このサービスを利用するには、アカウントにサインアップする*必要があります*。
内部の相互参照	同じドキュメント内の参照箇所を示します。
論理値、定数、正規表現、アブストラクタ	コード以外でも、前後と区別して表記すべき値などは特殊なフォントで表記します。 値が `null` の場合、レスポンス `false` を返します。

表記	説明／例
製品名および機能名	AWS の製品名および機能名は、初出時のみ斜体で表記します。 *Amazon Machine Image*（AMI）を作成してください。
オペレーション	テキスト内のオペレーションの表記です。 GetHITResponse オペレーションを使用します。
パラメータ	テキスト内のパラメータの表記です。 このオペレーションは、パラメータ *AccountID* をとります。
レスポンス要素	テキスト内のレスポンスの表記です。 ひとつの CollectionParent、およびひとつまたは複数の CollectionItems 用コンテナ。
技術ドキュメントの参照	他の AWS ドキュメント名の表記です。ドキュメントの参照先がハイパーリンクされている場合には、下線付きで表記します。 コンセプトの詳細については、*『Amazon Mechanical Turk Developer Guide』*を参照してください。
ユーザー入力値	ユーザーがタイプ入力するテキストは、特殊なフォントで表記します。 パスワードの入力を求めるプロンプトが現れたら、**MyPassword** とタイプします。
ユーザーインターフェースのコントロールとラベル	UI の項目名は、区別しやすいフォントで表記します。 「File」メニューで「Properties」をクリックします。
変数	サンプルのテキストをコマンドラインにコピーする際は、この書式で表記されている値を変更する必要があります。 % ec2-register *<your-s3-bucket>*/image.manifest 次の「記号の表記」も参照してください。

記号の表記

以下の表は、記号の表記規則をまとめたものです。

表記	記号	説明／例

表記	記号	説明／例
相互排他的なパラメータ	括弧および縦棒 (|)	コード記述の中で、複数のオプションからひとつを選択しなければならない場合、各オプションを縦棒で区切って表記します。 % data = hdfread (start | stride | edge)
任意指定のパラメータ XML 変数テキスト	角括弧 []	コード記述の中で、任意で指定するコマンドまたはパラメータは角括弧で囲みます。 % sed [-n, -quiet] XML の例では、前後のタグと区別するため、変数テキストを角括弧で囲みます。 <CustomerId>[ID]</CustomerId>
変数	山括弧 <>	コードサンプルでは、有効な値に置き換える変数を山括弧で囲みます。 % ec2-register <your-s3-bucket>/image.manifest