
Product Advertising API

Getting Started Guide

API Version 2013-08-01

Product Advertising API: Getting Started Guide: (Formerly known as Amazon Associates Web Service)

Copyright © 2014 Amazon.com and its affiliates. All rights reserved.

What Is Product Advertising API?	1
Getting Set Up	4
Making a Request	10
Submitting Your First Request	10
Parts of a Request	13
Implementing a Request	13
Processing Responses	19
Checking Request Execution	19
Processing Overview	21
Processing Implementations	23
Next Steps	26
Document History	28

What Is Product Advertising API?

Topics

- [Key Concepts \(p. 2\)](#)
- [Required Knowledge and Skills \(p. 2\)](#)
- [Product Advertising API Resources \(p. 2\)](#)

Welcome to the *Product Advertising API Getting Started Guide*. Product Advertising API is a free service that gives you access to Amazon's world-class web service, a service that millions of customers use every day. As a developer, you can build Product Advertising API applications that leverage this robust, scalable, and reliable technology. You get access to much of the data that Amazon uses, including the items for sale, customer reviews, seller reviews, as well as most of the functionality that you see on www.amazon.com, such as finding items, finding similar items, displaying customer reviews, and product promotions. In short, Product Advertising API operations open the doors to Amazon's databases so that you can take advantage of Amazon's sophisticated e-commerce data and functionality. Using Product Advertising API, you can build your own web store to sell Amazon items or your own items, or you can build an application that helps others create their own e-commerce storefronts.

This guide is divided into several major sections that allow you to practice using Product Advertising API in a simple environment. Each of the sections listed here builds on the previous sections, so that if you read and work through the examples in sequence, you gain a basic understanding of the Product Advertising API.

- **What is Product Advertising API?**—The rest of this section describes the key concepts of Product Advertising API, the knowledge required to use it, and related resources that you can use to interact with the service.
- **Getting Started**—The [Getting Started \(p. 4\)](#) section describes how to become an Amazon Associate and get the identifiers you need to submit requests.

Beyond the getting started section, you'll probably want to learn more about Product Advertising API operations. The following sections provide detailed information about working with Product Advertising API requests and responses:

- **Making a Request**—The [Making a Request \(p. 10\)](#) section describes how to send a simple Product Advertising API request in multiple programming languages.
- **Processing Responses**—The [Processing Responses \(p. 19\)](#) section describes how to parse the response to your request.

Key Concepts

Narrowing search results with response groups—The `ItemSearch` function can take a variety of parameters that help narrow the list of items to only those that match the customer's wishes. Response groups are included in requests either by default or explicitly. Response groups select from all of the item data returned to determine which data to display. The Offer response group, for example, returns information about an item's offer, that is, its price and availability.

Narrowing search results with request parameters—Search indices are used to restrict a request to a certain portion of Amazon's database. The database, called the catalog, contains millions of items. Returning 100,000 items is not useful to the customer. So, request parameters, including the search index, are used to narrow the results to make them match the customer's expectation. For example, a Harry Potter book and the DVD of that book reside in different search indices. By specifying the search index, you return the information relevant to the customer's interest.

Structured requests and responses—Requests are structured. Each has an endpoint, which is the URL of the Product Advertising API, `webservices.amazon.com`. Responses, by default, are returned in XML, which makes parsing the response easier.

Making REST requests to search for items—One of the first and most common tasks a customer undertakes is searching for items to buy. They supply search parameters, such as whether they are looking for a book, DVD, or article of clothing. This guide shows how to make a REST request in multiple computer languages that searches through Amazon's catalog of items and selects those that are related to the keywords entered on the command line. The response processing code assumes the items returned are books. The response is parsed so that the title, author, and price of the item are displayed.

Required Knowledge and Skills

This guide assumes that you are familiar with XML syntax and structure and that you have a basic understanding of web services. For overviews of these topics, go to the following links:

- [W3 Schools XML Tutorial](#)
- [W3 Schools Web Services Tutorial](#)

In addition, you need to be familiar with one of these programming languages: Java, C#, PHP, and Perl.

Showing Your Preferred Programming Language

You can hide the sections of this guide that don't apply to the programming language you are using. There is a language selection menu in the upper-right corner of pages with language-specific text. Select your language to hide all others, or select **All** to show the examples in all available languages.

Product Advertising API Resources

The following table lists related resources that you'll find useful as you work with Product Advertising API.

Resource	Description
Product Advertising API Developer Guide	The Developer Guide provides a detailed discussion of the Product Advertising API. It includes an architectural overview, programming reference, and API reference.

Product Advertising API Getting Started Guide
Product Advertising API Resources

Resource	Description
Product Advertising API information	The primary web page for information about Product Advertising API.
Discussion Forums	A community-based forum for developers to discuss technical questions related to Amazon Web Services.
Contact Us	A central contact point for inquiries concerning AWS billing, account, events, abuse, etc.
Conditions of Use	Detailed information about the copyright and trademark usage at Amazon.com and other topics.

Getting Set Up with Product Advertising API

This section provides step-by-step instructions that you need to complete before you can submit your first Product Advertising API request.

Getting Set Up with Product Advertising API

Step 1: Become an Associate (p. 4)
Step 2: Get Your Security Credentials (p. 5)
Step 3: Get the Tools You Need (p. 5)
Step 4: Set Up Your Development Environment (p. 7)

Step 1: Become an Associate

Associates earn commissions by using their own website to refer sales to www.amazon.com. To get a commission, an Associate must have an Associate ID. This ID works only in the locale in which you register. If you want to be an Associate in more than one locale, you must register in all relevant locales.

- Sign up to become an Associate at <http://associates.amazon.com/exec/panama/associates/apply>.

For more information about becoming an Associate, refer to one of the following websites, depending on your locale.

Locale	URL
Brazil	https://associados.amazon.com.br/
Canada	https://associates.amazon.ca/
China	https://associates.amazon.cn/
France	https://partenaires.amazon.fr/

Locale	URL
Germany	http://partnernet.amazon.de
India	https://affiliate-program.amazon.in/
Italy	https://programma-affiliazione.amazon.it/
Japan	https://affiliate.amazon.co.jp/
Spain	https://afiliados.amazon.es/
United Kingdom	https://affiliate-program.amazon.co.uk
United States	http://affiliate-program.amazon.com/

Getting Your Associate Tag

When you register as an Amazon Associate, an Associate tag is sent to you by return email. An Associate tag, also known as an Associate ID, is an automatically generated unique identifier that you will need to make requests through the Product Advertising API. When you log in to the Amazon Associates website for your locale, the page that you are directed to includes a message that says "Signed in as" followed by your Associate tag.

Step 2: Get Your Security Credentials

You need security credentials to call the Product Advertising API, authenticate requests, and identify yourself as the sender of a request. Two types of credentials are available:

- AWS security credentials (access key ID and secret access key)
- X.509 certificates

Note

IAM roles are not currently supported. You must use the root account credentials or certificates. To generate and manage your credentials, go to the [AWS Security Credentials Console](#).

Step 3: Get the Tools You Need

Programming Language Tools

Product Advertising API requests can be integrated into applications using most modern programming languages. In the following table, click the tool you would like to use to implement Product Advertising API. The link takes you to the corresponding website where you can download and install the appropriate toolkit.

Language	API Style	Tools Used
Java	SOAP	<ul style="list-style-type: none">• Java 6 or higher <p>Make sure the <i>PATH</i> environment variable points at the Java installation.</p> <ul style="list-style-type: none">• Eclipse 3.2 or higher <p>If you use Eclipse as your interactive development environment (IDE), you must use version 3.2 or higher. You can, however, use other IDEs, such as NetBeans.</p>
C#	SOAP	<ul style="list-style-type: none">• Microsoft Visual Studio 2005 C# Express Edition• .NET Framework 2.0
Perl	REST (using HTTP POST)	<p>To download the modules used in the following Perl example, go to CPAN website:</p> <ul style="list-style-type: none">• <code>Digest::HMAC::SHA1</code>• <code>MIME::Base64</code>• <code>LWP</code>• <code>XML::XPath</code>• <code>Date::Format</code>
PHP	REST (using HTTP GET)	<p>The PHP example uses the base installation of PHP5.</p> <p>Because PHP configurations vary, we're using a command-line interface to run our example. You are also welcome to run the example through a web server, but those details are not covered in this guide.</p>

Product Advertising API Signed Requests Helper

All requests you send to Product Advertising API must be authenticated using a signed version of the request. We have a tool, the [Product Advertising API Signed Requests Helper](#), to generate this signed request. You can use this tool online or download it to your machine.

Step 4: Set Up Your Development Environment

This section helps you confirm that your development environment is set up correctly. Skip to the section that corresponds to the toolkit you downloaded. Or, if you are viewing this document online, you can view the setup steps for your preferred programming language by using the **Filter View** list on the top-right corner of the page.

Topics

- [Java Setup \(p. 7\)](#)
- [C# Setup \(p. 8\)](#)
- [Perl Setup \(p. 8\)](#)
- [PHP Setup \(p. 9\)](#)

Java Setup

You can implement Product Advertising API operations directly in Java. You can also generate and use the Product Advertising API Java Client Side library to simplify your Java implementations. This section explains how to generate the Product Advertising API Java Client-Side Library. The next section shows you how to use it to create a request.

Generating the Stubs

You use the `wsimport` utility in Java 6 to generate the stubs from the Product Advertising API WSDL, which is located at <http://webservices.amazon.com/AWSECommerceService/AWSECommerceService.wsdl>.

To generate the Product Advertising API client-side library stubs

1. Go to the directory where you want to generate the stubs and create a "build" directory and a "src" directory.
All of the generated source code goes under "src" folder.
2. If you are using Eclipse 3.2, create a custom binding to disable "Wrapper Style" code generation.

```
<jaxws:bindings wsdlLocation="http://webservices.amazon.com/AWSECommerceService/AWSECommerceService.wsdl" xmlns:jaxws="http://java.sun.com/xml/ns/jaxws">  
  
  <jaxws:enableWrapperStyle>>false</jaxws:enableWrapperStyle>  
</jaxws:bindings>
```

This step is necessary because Eclipse 3.2 does not support wrapper style generated code. However, if your IDE does support wrapper-style generated code, such as NetBeans, this step is not required.

3. Run the command:

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

You can find the generated stubs in the path, `com.ECS.client.jax`.

Generated File Types

Several file types are generated in the package, com.ECS.client.jax:

- **AWSECommerceService**—This file identifies the Product Advertising API service.
- **AWSECommerceServicePortType**—This file provides the port type that the client can listen on. This file also contains a list of all Product Advertising API operation signatures that can be used to build the client.

C# Setup

Product Advertising API requires that you have successfully installed Microsoft Visual Studio.

To confirm the installation

1. Open Visual Studio 2005 C# Express Edition.
2. Click **Help > About Microsoft Visual Studio**.
A dialog box opens. It should list Microsoft Visual Studio 2005 and version 2.0 of the .NET Framework.

Create the SOAP Proxy in Visual Studio

In your application, you need to add a web reference to the Product Advertising API WSDL you want to use.

To add a web reference

1. On the **Project** menu, click **Add Web Reference**.
2. In the **URL** box, type `http://webservices.amazon.com/AWSECommerceService/AWSECommerceService.wsdl`, and then click **Go**.
3. Click **Add Reference**.

A new **Web References** folder is added to the **Solution Explorer**.

You can now reference the SOAP proxy using your project namespaces. For example:

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

Perl Setup

Run the following commands to verify that you have installed all of the necessary Perl modules:

```
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
```

You should not receive any error messages.

PHP Setup

To verify your PHP installation

- Use a command-line interface to run the following command:

```
php -version
```

This command assumes you are either in your PHP installation directory or it is in your PATH system variable.

The response should be similar to the following:

```
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
```

Making a Request in Product Advertising API

A request is a way of asking Product Advertising API to do something for you. For example, you might ask Product Advertising API to return information about a product category or about a single item. You might request that Product Advertising API return pictures of items for sale, or what other customers thought of them, or how much they cost. Product Advertising API enables you to ask these questions by sending requests over the Internet using REST or SOAP. Product Advertising API answers your request by returning an XML document.

The following sections describe how to make a Product Advertising API request using multiple programming languages. For a refresher on getting your AWS Access Key and other getting started details, review [Getting an AWS Access Key ID \(p. 5\)](#).

Topics

- [Submitting Your First Request in Product Advertising API \(p. 10\)](#)
- [Parts of a Product Advertising API Request \(p. 13\)](#)
- [Implementing a Product Advertising API Request \(p. 13\)](#)

Submitting Your First Request in Product Advertising API

To submit your first request to Product Advertising API

1. Open the [Product Advertising API Signed Requests Helper](#).
2. Enter your AWS Access Key ID and your AWS Secret Access Key.

Signed Requests Helper

amazon
Product Advertising API

<input type="text" value="AWS Access Key ID"/>	<input type="text" value="AWS Secret Access Key"/>
--	--

Product Advertising API Getting Started Guide Submitting Your First Request

3. In the **Unsigned URL** text box, enter the following, replacing `YourAssociateTagHere` with your own Associate tag:

```
http://webservices.amazon.com/onca/xml?Service=AWSECommerceService
&Version=2013-08-01
&Operation=ItemSearch
&SearchIndex=Books
&Keywords=harry+potter
&AssociateTag=YourAssociateTagHere
```

4. Click the **Display Signed URL** button.
5. Copy the text in the **Signed URL** text box.

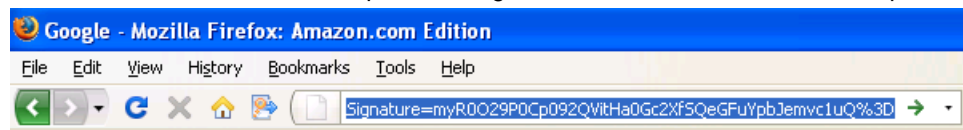
Signed URL:

```
http://webservices.amazon.com
/onca/xml?AWSAccessKeyId=
&AssociateTag=YourAssociateTagHere&Keywords=harry%20potter&
Operation=ItemSearch&SearchIndex=Books&
Service=AWSECommerceService&
Timestamp=2011-12-20T20%3A50%3A50.000Z&Version=2011-08-01&
Signature=708nHq5hd00U5v3Ho1GRy0S9iPyoldhyZEnKvkyoUTM%3D
```

Note

For your request to succeed, the *Timestamp* parameter can be no older than 15 minutes.

6. In a new browser window or tab, paste the signed URL into the address line, and press Enter.



Congratulations! You just made your first Product Advertising API request.

From this example, you can see that a REST request is a URL. Everything before the question mark (?) specifies the destination of the request. This destination is the same for every Product Advertising API request (sent to the same locale). Everything after the question mark is a parameter in the request. This request searches (*Operation=ItemSearch*) for all books (*SearchIndex=Books*) that have "Harry Potter" in the title (*Title=Harry%20Potter*).

Tip

Product Advertising API has a number of locales, including the US, JP (Japan), FR (France), DE (Germany), United Kingdom (UK), and Canada (CA). Each locale has a slightly different endpoint. For example, the JP endpoint is `http://ecs.amazonaws.jp`. You can send requests to any locale. But typically you send requests to the locale in which your customers' reside. For more information about all the locales and endpoints, see [Anatomy of a REST Request](#) in the *Product Advertising API Developer Guide*.

Product Advertising API responds to the request by returning an XML document. The following is a snippet of the response to the foregoing example.

```
<TotalResults>2427</TotalResults>
<TotalPages>243</TotalPages>
<Item>
  <ASIN>0545139708</ASIN>
  <DetailPageURL>http://www.amazon.com/Harry-Potter-Deathly-Hallows-Rowl
ing/dp/0545139708%3FSubscriptionId
%3DAKIAIOSFODNN7EXAMPLE%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26cre
ative
```

Product Advertising API Getting Started Guide
Submitting Your First Request

```
    %3D165953%26creativeASIN%3D0545139708</DetailPageURL>
  <ItemLinks>
    <ItemLink>
      <Description>Technical Details</Description>
      <URL>http://www.amazon.com/Harry-Potter-Deathly-Hallows-Rowling/dp/tech-
data/0545139708%3FSubscriptionId
      %3DAKIAIOSFODNN7EXAMPLE%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26cre
ative
      %3D386001%26creativeASIN%3D0545139708</URL>
    </ItemLink>
    <ItemLink>
      <Description>Add To Baby Registry</Description>
      <URL>http://www.amazon.com/gp/registry/baby/add-item.html%3Fas
in.0%3D0545139708%26SubscriptionId
      %3DAKIAIOSFODNN7EXAMPLE%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26cre
ative
      %3D386001%26creativeASIN%3D0545139708</URL>
    </ItemLink>
    <ItemLink>
      <Description>Add To Wedding Registry</Description>
      <URL>http://www.amazon.com/gp/registry/wedding/add-item.html%3Fas
in.0%3D0545139708%26SubscriptionId
      %3DAKIAIOSFODNN7EXAMPLE%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26cre
ative
      %3D386001%26creativeASIN%3D0545139708</URL>
    </ItemLink>
    <ItemLink>
      <Description>Add To Wishlist</Description>
      <URL>http://www.amazon.com/gp/registry/wishlist/add-item.html%3Fas
in.0%3D0545139708%26SubscriptionId
      %3DAKIAIOSFODNN7EXAMPLE%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26cre
ative
      %3D386001%26creativeASIN%3D0545139708</URL>
    </ItemLink>
    <ItemLink>
      <Description>Tell A Friend</Description>
      <URL>http://www.amazon.com/gp/pdp/taf/0545139708%3FSubscriptionId%3DAKI
AIOSFODNN7EXAMPLE
      %26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D386001%26cre
ativeASIN
      %3D0545139708</URL>
    </ItemLink>
    <ItemLink>
      <Description>All Customer Reviews</Description>
      <URL>http://www.amazon.com/review/product/0545139708%3FSubscriptionId
      %3DAKIAIOSFODNN7EXAMPLE%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025
      %26creative%3D386001%26creativeASIN%3D0545139708</URL>
    </ItemLink>
    <ItemLink>
      <Description>All Offers</Description>
      <URL>http://www.amazon.com/gp/offer-listing/0545139708%3FSubscriptionId

      %3DAKIAIOSFODNN7EXAMPLE%26tag%3Dws%26linkCode%3Dxm2%26camp
      %3D2025%26creative%3D386001%26creativeASIN%3D0545139708</URL>
    </ItemLink>
  </ItemLinks>
  <ItemAttributes>
    <Author>J.K. Rowling</Author>
```

```
<Manufacturer>Arthur A. Levine Books</Manufacturer>  
<ProductGroup>Book</ProductGroup>  
<Title>Harry Potter And The Deathly Hallows</Title>  
</ItemAttributes>
```

This snippet shows that 2427 items match the search criteria. The first item returned is, "Harry Potter and the Deathly Hallows." Many details about the book are returned, including the name of the author, illustrator, book manufacturer, and product identifier (ASIN). If you copy the *DetailPageURL* into a browser, a web page about this book is displayed.

Tip

An ASIN (Amazon Standard Item Number) is an alphanumeric token that uniquely identifies items for sale on Amazon.

Submitting URLs in a browser provides a good demonstration of how Product Advertising API requests and responses work. This practice, however, is not appropriate for customer applications. The remainder of this section describes how to issue Product Advertising API requests programmatically. The next section describes how to process the responses programmatically.

Parts of a Product Advertising API Request

Every programming language has its own style and requirements. For that reason, each implementation of submitting a Product Advertising API request is a little different. The following programmatic tasks, however, are shared across all programming languages for implementing a Product Advertising API request.

Programmatic tasks

1	Create a request object.
2	Add parameters and their values to the request.
3	Set up the request.
4	Send the request.

The following sections explain how to accomplish these tasks in different programming languages.

Implementing a Product Advertising API Request

This section shows you how to implement an `ItemSearch` request in various programming languages. It's important to note that these examples do not work as is since you must include valid request authentication. You may find the following links helpful as you learn more about Product Advertising API and making requests.

- Complete code examples—[Product Advertising API Sample Code & Libraries](#)
- Authenticating requests—[Request Authentication](#) in the *Product Advertising API Developer Guide*
- Timestamp and Signature—[Authentication Parameters](#) in the *Product Advertising API Developer Guide*.
- About Product Advertising API—<http://aws.amazon.com/associates/>

Another useful tool for trying out various requests is the [Product Advertising API Scratchpad](#). This tool is a web-based application that enables you to quickly submit requests to the Product Advertising API. You can explore basic `ItemLookup` and `ItemSearch` requests using REST queries.

Note

If you are viewing this document online, you can view the example code in only one programming language by clicking your preferred language in the **Filter View** drop-down on the top-right corner of the page.

Java

The following Java code implements an `ItemSearch` request in which the customer enters values for `SearchIndex` and `Keywords`. This example uses the Java client side library to simplify the implementation of the request. To download the client side library using `wsimport` and generate the stubs, see [Java Setup \(p. 7\)](#) and use the following code.

```
// 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("Books");
itemRequest.setKeywords("dog");
itemRequest.setVersion("2013-08-01");
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#

The following C# code implements an `ItemSearch` request in which the customer enters values for `SearchIndex` and `Keywords`. Comments are inline.

Note

The `GettingStartedGuideSample.com.amazonaws.ecs` package is auto-generated when you use the .NET "Add Web Reference..." dialog box.

```
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[] { "DVD", "Matrix" };
        }

        // 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 an ItemSearch wrapper
        ItemSearch search = new ItemSearch();
        search.AssociateTag = "[Your Associate ID]";
        search.AWSAccessKeyId = "[Your ID]";
        search.Version = "2013-08-01";

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

        // Fill the 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

The following Perl code implements an `Keywords` request in which the customer enters values for `SearchIndex` and `Keywords`. Comments are inline.

```
#!/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://webservices.amazon.com/onca/xml";
my $service = "AWSECommerceService";
my $accesskey = "[INSERT YOUR ACCESS KEY ID HERE]";
my $operation = "ItemSearch";
my $version = "2013-08-01";

# Assemble the REST request URL.
my $request =
    "$EndPoint?" .
    "Service=$service&" .
    "AWSAccessKeyId=$accesskey&" .
    "Operation=$operation&" .
    "Keywords=$keywords&" .
    "SearchIndex=$searchIndex&" .
    "Signature=[Request Signature]&" .
    "Version=$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

The following PHP code implements an `ItemSearch` request in which the customer enters values for `SearchIndex` and `Keywords`. Store this sample code in a file named `SimpleStore.php`. Comments are inline.

```
<?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 = "2013-08-01";
$ResponseGroup = "ItemAttributes,Offers";
//User interface provides values
//for $SearchIndex and $Keywords
```

```
//Define the request
$request=
    "http://webservices.amazon.com/onca/xml"
    . "?Service=AWSECommerceService"
    . "&AssociateTag=" . Associate_tag
    . "&AWSAccessKeyId=" . Access_Key_ID
    . "&Operation=" . $Operation
    . "&Version=" . $Version
    . "&SearchIndex=" . $SearchIndex
    . "&Keywords=" . $Keywords
    . "&Signature=" . [Request Signature]
    . "&ResponseGroup=" . $ResponseGroup;

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

The first part of this implementation constructs the `ItemSearch` request. The first parameters in the list are those that the customer cannot alter, including the endpoint, the service name, the Access Key ID, Associate Tag, Product Advertising API version number, and operation name.

The last two parameters, `SearchIndex` and `Keywords`, are values set by the customer through the user interface. A `SearchIndex` is similar to a product category, such as Books, Automobile, or Jewelry. `Keywords` is a word or phrase. The request selects items in the specified search index that have the `Keywords` value in their title or description.

The last two parameter values are entered by a customer using a web application, for example:

```
<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='Books'>Books</option>
                <option value='DVD'>DVD</option>
                <option value='Music'>Music</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>
```

This example uses a table to format a web page, which is composed of an HTML form. An HTML select statement provides a drop-down list of value choices for `SearchIndex`. An HTML input statement provides a text box for the customer to enter the `Keywords` value.

The request is sent using the PHP command, `file_get_contents`.

Processing Responses in Product Advertising API

Now that you've successfully sent a Product Advertising API request, you're ready to receive and process the response. This section discusses how to do that. The code examples in this section are continuations of the code examples presented in the previous section. For example, the variable names in this section match those in the previous section.

Topics

- [Checking Request Execution in Product Advertising API \(p. 19\)](#)
- [Product Advertising API Processing Overview \(p. 21\)](#)
- [Product Advertising API Processing Implementations \(p. 23\)](#)

Checking Request Execution in Product Advertising API

You can check the execution of a request first by examining the *IsValid* element in each response. If the element is set to True, the request executed successfully and you can display the information in the response. A value of False, however, indicates that there was an error in the request syntax. You can start troubleshooting the error in the request by viewing the errors returned in the response. The following example error statement shows that the request did not contain a required parameter, *ItemId*.

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

The *IsValid* element, however, is not always returned when a request fails. If, for example, you mistype the name of the operation, Product Advertising API returns the following message, which does not include the *IsValid* element :

```
<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, ItemLookup, SimilarityLookup,
SellerLookup, ItemSearch, BrowseNodeLookup, CartModify, CartClear, CartCreate,
CartAdd, SellerListingSearch.
  </Message>
</Error>
```

The *IsValid* value *True* specifies that the request was valid and was executed. That does not mean, however, that a result was obtained. There may not have been any items that satisfied the search criteria, for example. To check for this condition, either search for the presence of an *Error* element, or evaluate the value of the *TotalItems* element. If the value is zero, there are no results to display, as shown in the following example.

```
<IsValid>True</IsValid> ...
<Error>
  <Code>AWS.ECommerceService.NoExactMatches</Code>
  <Message>We did not find any matches for your request.</Message>
</Error> ...
<TotalResults>0</TotalResults>
<TotalPages>0</TotalPages>
```

Java

Errors can occur at many levels in the XML response. The following example determines if the response contains the element, *OperationRequest*. This response element is included in every response. If it is missing, the response is null. That might happen, for example, if the Product Advertising API web service times out the request. The second error check determines if there is an *Items* response element in the response.

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

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

To do a thorough job of error checking, you would have to evaluate all of the response elements returned to see if they were, in fact, returned. The preceding example provides a template for such code. Including all of that code here would complicate the example beyond the scope of this guide.

C#

The following code snippet verifies that the request executed successfully. The code checks for a null response.

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

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

Perl

The following code snippet verifies that the request executed successfully. The code checks for the presence of "Error" in the response.

```
#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

The following code snippet verifies that the request executed successfully. The code checks for an Error element in the XML response.

```
//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";
}
```

Product Advertising API Processing Overview

In the previous section, you saw that a Product Advertising API response is an XML document. The elements returned and their values depend on the data stored in Amazon's databases and the response groups specified in the request.

A response group tailors the information returned in a response. For example, the *Images* response group returns the images of items returned in the response. The *TopSellers* response group returns the top selling items in a search index. The *Product Advertising API Developer Guide* lists all of the elements that a response group can potentially return.

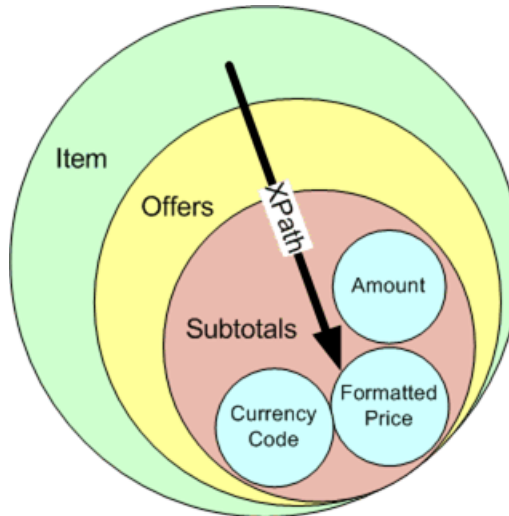
The elements returned by a response group are structured. For example, the *Item* element has several child elements, one of which is *Offers*, which has a child element, *Subtotal*, which itself has three child elements: *Amount*, *CurrencyCode*, and *FormattedPrice*, as shown in the following response snippet..

```
<Item>
  <Offers>
    <Subtotal>
```


Product Advertising API Getting Started Guide Processing Overview

```
<Amount>999</Amount>  
<CurrencyCode>US</CurrencyCode>  
<FormattedPrice>$9.99</FormattedPrice>
```

The parent-to-child succession of structured elements is called an *XPath*. To parse a result, the Product Advertising API response is turned into an object and then XPaths are used as an efficient means of finding elements and their values.



The following PHP example shows how an XPath is used to display the *FormattedPrice* value after first making sure there is a value.

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

The similar expression in C# is:

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

Returning additional values just requires the use of different XPaths.

Typically, responses return more than one item. For that reason, the parsing algorithm must iterate through all of the items returned in a response. For example, in PHP:

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

In C#:

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

Product Advertising API Processing Implementations

The following sections show these parsing principles applied more robustly across several programming languages.

Java

In the previous section, the request retrieved a response object. The Product Advertising API Client Side Library contains methods that can return a variety of values from that object. The following code retrieves from the response object the values of the following elements: items, item, item attributes, and title.

```
// 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("Book Name: " +
            item.getItemAttributes().getTitle());
    }
}
```

C#

The following C# code processes the response returned by Product Advertising API. This code is a continuation of the C# request sample code.

```
//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.Author + "\n"
+
            "Price: " + item.ItemAttributes.ListPrice.FormattedPrice + "\n"
        );
    }
}
//Catch and display any exceptions
catch (Exception ex)
{
    Console.WriteLine("An error occurred: " + ex.ToString());
}

Console.ReadLine();
}
```

This code uses a *for* statement to iterate through all of the Items in the response. The title, author, and price element values are displayed using *Console.WriteLine*.

Perl

The following Perl code processes the response returned by Product Advertising API. This code is a continuation of the Perl request sample code.

```
# 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 author names
my @authors;
for (my $j = 1;
     $j <= $xp->findvalue("count(/ItemSearchResponse/Items/Item[$i]/ItemAttributes/Author)");
     $j++)
{
  push @authors, $xp->findvalue("/ItemSearchResponse/Items/Item[$i]/ItemAttributes/Author[$j]");
}

# Find titles, prices, and display them with the authors
print "Title: ", $xp->findvalue("/ItemSearchResponse/Items/Item[$i]/ItemAttributes/Title"), "\n",
      "Author: ", join(", ", @authors), "\n",
      "Price: ", $xp->findvalue("/ItemSearchResponse/Items/Item[$i]/Offers/Offer/OfferListing/Price/FormattedPrice"), "\n\n";
}
}
```

This code uses a *for* statement to iterate through all of the Items in the response. The *title*, *author*, and *price* element values are displayed using *print*.

PHP

The following PHP code processes the response returned by Product Advertising API. This code is a continuation of the PHP request sample code.

```
<?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->Author)) {
print("<br>Author: ".$current->ItemAttributes->Author);
} elseif
(isset($current->Offers->Offer->Price->FormattedPrice)){
print("<br>Price:
".$current->Offers->Offer->Price->FormattedPrice);
}else{
print("<center>No matches found.</center>");
}
}
}
?>
```

The Product Advertising API response is put into an object, *\$parsed_xml*, using the PHP command, `simplexml_load_string`. The response is displayed using the function, `printSearchResults`, which is defined as:

The code first checks to see if any items were returned in the response:

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

The item attributes are located using the Xpath of the elements in the response: `Items->TotalResults`.

The code then parses the result object, *\$parsed_xml*, by iterating over each item returned in the response. The `Items->Item` Xpath is set to *\$current*.

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

\$current is used to access all of the item attributes in the response. For example, the following line displays the title:

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

The code only displays item attributes that are present in the response:

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

The dot in the `print` statement concatenates the display of the *Director* attribute to all of the previous attributes displayed.

Next Steps

Congratulations! Now that you have completed the basic example presented in this guide, you are ready to start designing your own Product Advertising API application. Although, most applications built on Product Advertising API are not as simple as the example in this guide, the principles used in the example readily apply to more complex applications.

The Product Advertising API provides a wealth of opportunities for developing new and innovative applications and websites. Previous sections covered in depth how to find an item for sale using `ItemSearch`. Finding an item is often the first task a Product Advertising API application implements. The tasks presented in this section are ordered in a use case scenario that is common for a customer using a Product Advertising API application.

Tasks in a Typical Use Case

Providing More Item Details (p. 26)
Adding an Item to a Shopping Cart (p. 27)
Purchasing the Item (p. 27)

All of these tasks are covered in greater depth in the *Product Advertising API Developer Guide*. Also, for a sample implementation of many of these tasks, see the `SimpleStore.php` application, which is available at [AWS Developer's Resource Center](#).

Providing More Item Details

An `ItemSearch` request, which we discussed and implemented in the previous sections, often returns multiple items. Typically, a Product Advertising API application displays a small image of each of those items along with a short description. A customer, however, often likes to pick from the list one or more items that look interesting so they can learn more about them. While it is possible to display extended information about each of the items returned by `ItemSearch`, the length of the web page would grow substantially. For that reason, we recommend that you provide a customer with extended information only when they show interest in a specific item.

Given the item identifier returned by `ItemSearch`, you can return extended information about any of the displayed items using the `ItemLookup` operation. `ItemLookup`, for example, can return all of the physical characteristics of the item along with pricing information.

Adding an Item to a Shopping Cart

Once a customer decides to purchase an item, he or she must be able to add it to a Product Advertising API remote shopping cart. Typically, you implement this with a user interface button labeled, for example, **Add to Cart**. The Product Advertising API operations that facilitate this functionality are `CartCreate` and `CartAdd`. Use `CartCreate` if the customer does not already have a shopping cart or `CartAdd` if the customer does.

Purchasing the Item

Now that the item is in the Product Advertising API remote shopping cart, the customer can purchase it. You can implement this task using a user interface button labeled, for example, **Proceed to Checkout**. The actual process of getting the customer's billing and shipping information, and method of payment is handled entirely by Amazon in what is called the Order Pipeline. The only task your application or website must implement is sending a `PurchaseURL` in a request to Amazon. Every cart operation returns the `PurchaseURL`. It contains all of the information necessary for Amazon to locate the customer's Product Advertising API remote shopping cart on its servers. The `PurchaseURL` also contains Associate information so that if an Associate brokered the sale, they receive a commission.

Where to Go from Here

There you have it: a complete shopping cycle, from finding an item to purchasing it. The use case scenario covers only a small slice of the functionality that Product Advertising API offers. For example, given a customer's demonstrated interest in an item, you might want to:

- Present similar items for sale to the customer.
- Present the top selling items in the same product category.
- Help the customer find a friend's wish list so that a wedding or baby shower gift can be purchased.

For more information about these options, see the [Product Advertising API Developer Guide](#). For more information about joining the developer forums, go to [Product Advertising API developer forum](#). There, you can ask questions and get answers from fellow developers and the staff at Amazon.

For general information about the Product Advertising API, as well as for code samples and other resources, go to the [Product Advertising API website](#).

Document History

This Document History describes the important changes to the documentation in this release of Product Advertising API.

Relevant Dates to this History:

- **Current product version**—2013-08-01
- **Latest product release**—August 2011
- **Last document update**—1 August 2012

Change	Description	Release Date
Guide updates	We reorganized the guide to make topics easier to discover and to remove redundancy.	1 August 2012
New Marketplace	The ES (Spain) marketplace was added.	20 September 2011
Marketplace update	This guide has been updated to add the CN (China) and IT (Italy) marketplaces.	1 August 2011
Version update	This guide has been updated to comply with the latest WSDL.	1 August 2011
Rebranding	This guide has been updated to reflect the change in the service's name.	
Enhancement	Added narrative about how to generate a signed request. For more information, see Submitting Your First Request in Product Advertising API (p. 10).	