Product Advertising API

Getting Started Guide

API Version 2013-08-01

amazon Product Advertising API

Product Advertising API: Getting Started Guide

Table of Contents

What Is Product Advertising API?

Topics

Amazon has developed a world-class web 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 a lot of the data used by Amazon including the items for sale, customer reviews, seller reviews, as well as most of the functionality you see on Amazon.com, such as finding 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. Build your own web store to sell Amazon items or your own items.

Best of all, Product Advertising API is free. By signing up to become a Product Advertising API developer, you join the tens of thousands of developers who are already realizing financial gains by creating Product Advertising API-driven applications and web stores.

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 as 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. 3) section describes how to become an Amazon Associate and get the identifiers you need to submit requests.

After you've read the Getting Started section, you'll 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. 13) section describes how to send a simple Product Advertising API request in multiple programming languages.
- **Processing Responses**—The Processing Responses (p. 20) 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 expectations. 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. For example, the Offer response group returns information about an item's offer, which is the 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. By default, responses 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 customers undertake is searching for items to buy. Customers enter search parameters for items they are looking for, such as a book title or an 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 must be familiar with one of these programming languages: Java, C#, PHP, and Perl.

Showing Your Preferred Programming Language

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

Getting Started with Product Advertising API

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

Getting Started with Product Advertising API

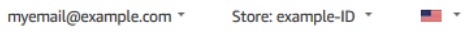
Becoming an Associate

Associates earn commissions by using their own websites to refer sales to Amazon.com. To get a commission, an Associate must have an Associate ID, also known as an Associate tag. The Associate ID is an automatically generated unique identifier that you will need to make requests through the Product Advertising API.

> **Note**
> You must register for Amazon Associates before you register as a Product Advertising API developer. If you make a request with an invalid Associate ID, the Product Advertising API returns an error.

To become an associate

1. Using the following table, choose the Amazon Associates URL for your locale.
2. Follow the instructions to create an Amazon Associates account.
3. When you register as an Amazon Associate, an Associate ID is sent you in email. When you sign in to Amazon Associates for your locale, the home page shows your email and ID in the corner.



> **Note**
> One of the requirements for becoming an Associate is that you provide the URL of your site. If your site is not yet public but you want to test against the API, you must still provide a URL during registration.
> Your Associate 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 separately for each locale.

Locale	URL
Brazil	https://associados.amazon.com.br
Canada	https://associates.amazon.ca

Locale	URL
China	https://associates.amazon.cn
France	https://partenaires.amazon.fr
Germany	http://partnernet.amazon.de
India	https://affiliate-program.amazon.in
Italy	https://programma-affiliazione.amazon.it
Japan	https://affiliate.amazon.co.jp
Mexico	https://afiliados.amazon.com.mx/gp/associates/join/landing/main.html
Spain	https://afiliados.amazon.es
United Kingdom	https://affiliate-program.amazon.co.uk
United States	https://affiliate-program.amazon.com/

Becoming a Product Advertising API Developer

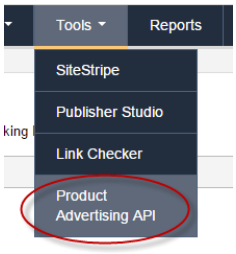
The Product Advertising API allows developers to advertise products from the following Amazon sites:

Locale	URL
Brazil	http://www.amazon.com.br
Canada	http://www.amazon.ca
China	http://www.amazon.cn
France	http://www.amazon.fr
Germany	http://www.amazon.de
India	http://www.amazon.in/
Italy	http://www.amazon.it
Japan	http://www.amazon.co.jp
Mexico	http://www.amazon.com.mx
Spain	http://www.amazon.es
United Kingdom	http://www.amazon.co.uk/
United States	http://www.amazon.com

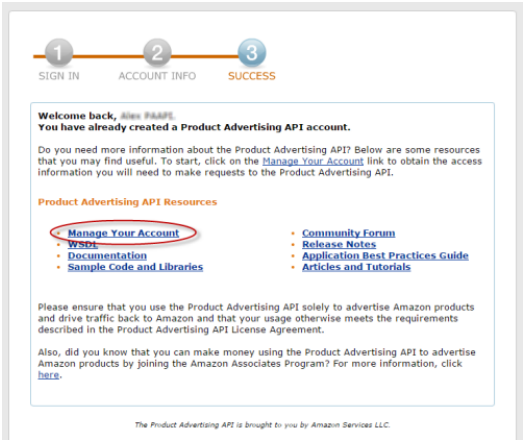
Note
Before you register for the Product Advertising API, register for Amazon Associates. See Becoming an Associate (p. 3).

To register as a Product Advertising API developer

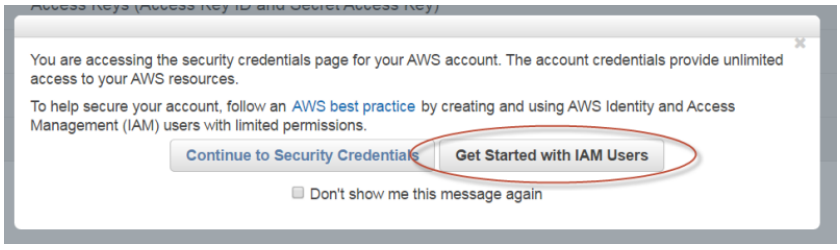
1. On the Amazon Associates page, under **Tools**, choose **Product Advertising API**. You can also use the table at the bottom of this page to locate the Product Advertising API URL for your locale.



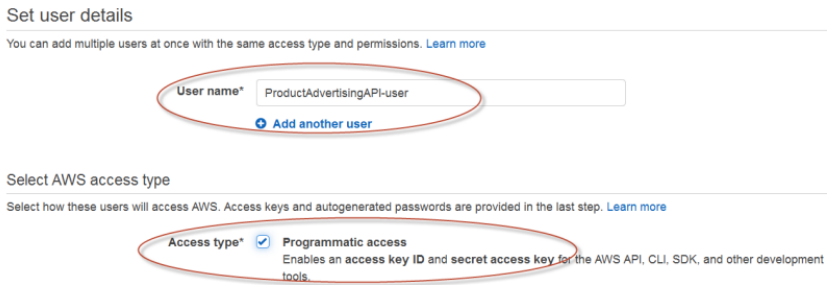
2. Choose **Sign Up Now**.
3. Use the primary or secondary email address in your Associates account to sign in.
4. Follow the steps to register.
5. On the **Success** page, choose **Manage Your Account**. We automatically create an AWS account for you. You will use your AWS account security credentials to make calls to the Product Advertising API.



6. On the **Manage Your Account** page, choose the **AWS Security Credentials Console** link. Sign in to your AWS account with the same email address and password you used to register for the Product Advertising API.
7. On the pop-up message, choose **Get Started with IAM Users**. In the following steps, you will create an IAM user and then give the user permission to access the Product Advertising API.



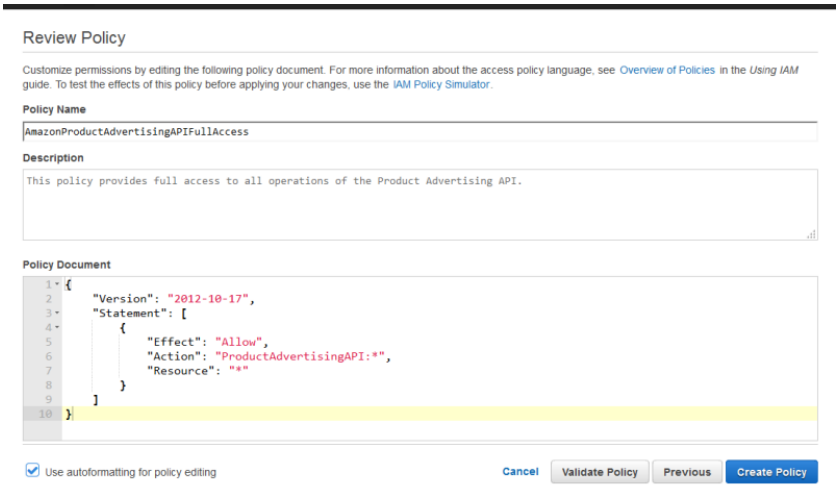
8. Choose **Add user**.
9. Type a user name such as **ProductAdvertisingAPI-user**. For **Access Type**, check **Programmatic access**.



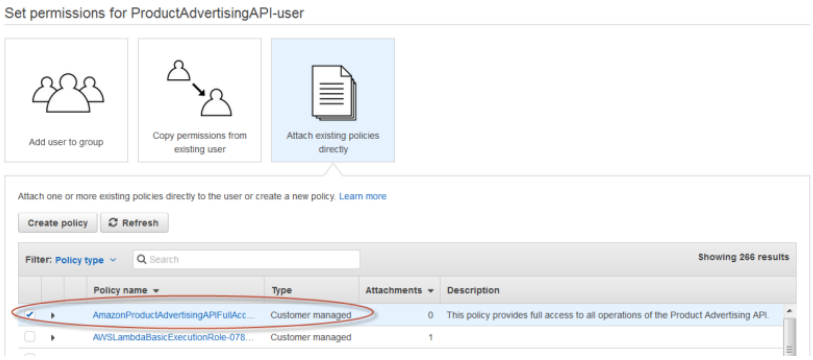
10. Choose **Next: Permissions**.
11. Choose **Attach existing policies directly**.
12. Choose **Create policy**. This opens a new page.
13. For **Create Your Own Policy**, choose **Select**.
14. On the **Review Policy** page, for **Policy Name**, type a name such as **AmazonProductAdvertisingAPIFullAccess**.
15. For **Description**, type a description such as: "This policy provides full access to all operations of the Product Advertising API."
16. For **Policy Document**, copy and paste the following policy in the field.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ProductAdvertisingAPI:*",
            "Resource": "*"
        }
    ]
}
```

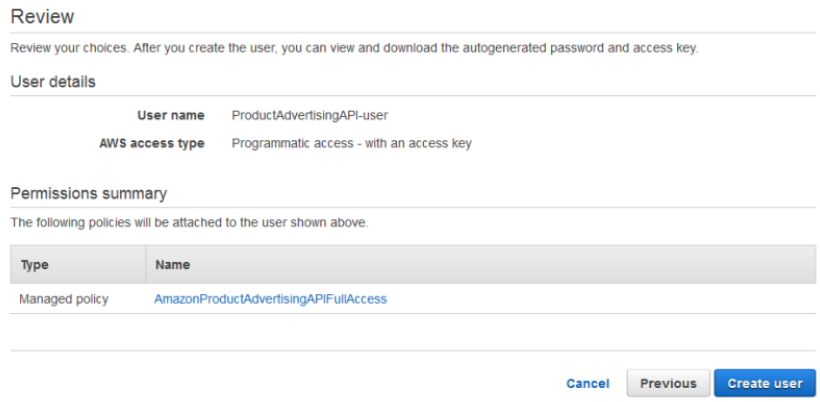
The following is an example Product Advertising API policy.



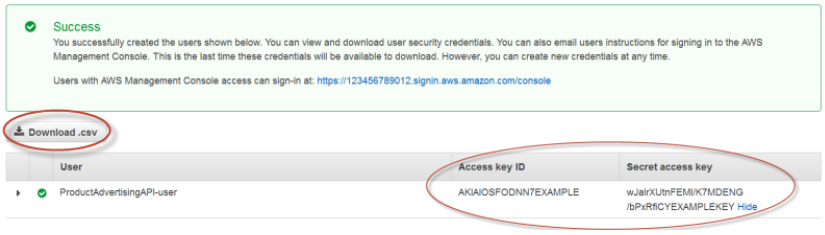
17. Choose **Validate Policy** and then choose **Create Policy**.
18. Return to the **Add user** page and for **Policy type**, search and choose the policy that you created. Choose **Refresh** if the policy doesn't appear.



19. Choose **Next: Review**.
20. Review your choices and then choose **Create user**.



21. Choose **Download.csv** to retrieve the credentials or choose **Show** to find the secret access key.



22. Save the access key information in a safe location. You will use these credentials to make calls to the Product Advertising API.

    Important
    You can access the secret access key only when you first create an access key pair. For security reasons, it cannot be retrieved at a later time. Make sure you save both the access key ID and its matching secret key. If you lose them, you must create a new access key pair.

23. When finished, choose **Close**. The user has the permissions required to access the Product Advertising API.

Note
As a best practice, use the IAM user credentials to access the Product Advertising API. You can continue using your root credentials, but root credentials provide unlimited access to your AWS resources. An IAM user has permission to access only the services you specify.

Locale	Product Advertising API URL
Brazil	https://associados.amazon.com.br/gp/associates/apply/main.html
Canada	https://associates.amazon.ca/gp/flex/advertising/api/sign-in.html
China	https://associates.amazon.cn/gp/advertising/api/detail/main.html
France	https://partenaires.amazon.fr/gp/flex/advertising/api/sign-in.html
Germany	https://partnernet.amazon.de/gp/flex/advertising/api/sign-in.html
India	https://affiliate-program.amazon.in/gp/associates/apply/main.html
Italy	https://programma-affiliazione.amazon.it/gp/advertising/api/detail/main.html
Japan	https://affiliate-program.amazon.com/gp/flex/advertising/api/sign-in-jp.html
Mexico	https://afiliados.amazon.com.mx/gp/advertising/api/detail/main.html
Spain	https://afiliados.amazon.es/gp/flex/advertising/api/sign-in.html
United Kingdom	https://affiliate-program.amazon.co.uk/gp/flex/advertising/api/sign-in.html
United States	https://affiliate-program.amazon.com/gp/flex/advertising/api/sign-in.html

Reading the Licensing Agreement

You will need to review and accept the terms and conditions of the license agreement to become a Product Advertising API developer. To read the Product Advertising API licensing agreement, go to the license agreement link for your locale:

Locale	License Agreement URL
Brazil	https://associados.amazon.com.br/gp/associates/agreement/
Canada	https://associates.amazon.ca/gp/advertising/api/detail/agreement.html
China	https://associates.amazon.cn/gp/advertising/api/detail/agreement.html
France	http://partenaires.amazon.fr/gp/advertising/api/detail/agreement.html
Germany	https://partnernet.amazon.de/gp/advertising/api/detail/agreement.html
India	http://affiliate-program.amazon.in/gp/advertising/api/detail/agreement.html
Italy	https://programma-affiliazione.amazon.it/gp/advertising/api/detail/agreement.html
Japan	https://affiliate.amazon.co.jp/gp/advertising/api/detail/agreement.html
Mexico	https://afiliados.amazon.com.mx/gp/advertising/api/detail/agreement.html
Spain	https://afiliados.amazon.es/gp/advertising/api/detail/agreement.html
United Kingdom	https://affiliate-program.amazon.co.uk/gp/advertising/api/detail/agreement.html
United States	https://affiliate-program.amazon.com/gp/advertising/api/detail/agreement.html

Note
If you plan to use the Product Advertising API to advertise Amazon products from a locale other than the one you signed up in, review the license agreement for that locale. The terms and conditions for each locale apply to any use of the Product Advertising API in that locale. Review the Amazon Product Advertising API Best Practices to make sure your application is compliant, scalable, and efficient.

Important
In addition to the Product Advertising API License Agreement, be sure to read your locale's Associates Program Operating Agreement for information on usage guidelines, policies, and requirements.

Get the Tools You Need

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

Language	API Style	Tools Used
Java	SOAP	• Java 6 or later Make sure the `PATH` environment variable points at the Java installation. • Eclipse 3.2 or later If you use Eclipse as your interactive development environment (IDE), you must use version 3.2 or later. You can, however, use other IDEs, such as NetBeans.
C#	SOAP	• Microsoft Visual Studio 2005 C# Express Edition • .NET Framework 2.0
Perl	REST (using HTTP POST)	To download the modules used in the following Perl example, go to CPAN website: • `Digest::HMAC_SHA256` • `MIME::Base64` • `LWP` • `XML::XPath` • `Date::Format`
PHP	REST (using HTTP GET)	The PHP example uses the base installation of PHP5. Because PHP configurations vary, we're using a command-line interface to run our example. You can also run the

Language	API Style	Tools Used
		example through a web server, but those details are not covered in this guide.

Product Advertising API Signed Requests Helper

All requests you send to Product Advertising API must be authenticated using a signed version of the request.

You can use the Product Advertising API Signed Requests Helper to generate this signed request. You can use this tool online or download it to your machine.

Product Advertising API Scratchpad

You can also use the Product Advertising API Scratchpad to generate sample requests and responses.

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

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 required because Eclipse 3.2 does not support wrapper-style generated code. If your IDE, such as NetBeans, supports wrapper-style generated code, 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 Microsoft Visual 2005 C# Express Edition.
2. Click **Help** > **About Microsoft Visual Studio**.

   The dialog box that opens 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 must 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 required Perl modules:

```
perl -MDigest::HMAC_SHA256 -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 images of items for sale, customer reviews, or price. 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.

Topics

Submitting Your First Request in Product Advertising API

To submit your first request to Product Advertising API

1. Go to the Product Advertising API Scratchpad.
2. Choose an operation, for example **ItemSearch**.
3. Under **Common parameters**, choose your marketplace and type your credentials.
4. Choose your request parameters. For example, for the **Keywords** parameter, type "Harry Potter" and then click **Run request**.
5. Under **Request URL**, your unsigned and request URL appear.
6. Under **Response**, locate your response examples.

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

Example unsigned request

```
http://webservices.amazon.com/onca/xml?Service=AWSECommerceService&
Operation=ItemSearch&
SubscriptionId=ExampleID&
AssociateTag=ExampleTag&
SearchIndex=Books&
Keywords=Harry Potter&
ResponseGroup=Images,ItemAttributes,Offers
```

Example signed request

```
http://webservices.amazon.com/onca/xml?Service=AWSECommerceService&
Operation=ItemSearch&
AWSAccessKeyId=ExampleID&
AssociateTag=ExampleTag&
Keywords=Harry%20Potter&
ResponseGroup=Images%2CItemAttributes%2COffers&
SearchIndex=Books&
Timestamp=2015-08-11T17%3A51%3A56.000Z&
Signature=oC%2Bv7Q33hROJDi2X79dYn%2BMzm9bRxDqYXk9qHTx3yEo%3D
```

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 and each locale has a slightly different endpoint. Although you can send requests to any locale, you typically send them to the locale where your customers reside. For more information about 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 preceding example.

```
<TotalResults>2427</TotalResults>
  <TotalPages>243</TotalPages>
  <Item>
    <ASIN>0545139708</ASIN>
    <DetailPageURL>https://www.amazon.com/Harry-Potter-Deathly-Hallows-Rowling/
dp/0545139708%3FSubscriptionId
      %3DAKIAIOSFODNN7EXAMPLE%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative
      %3D165953%26creativeASIN%3D0545139708</DetailPageURL>
    <ItemLinks>
      <ItemLink>
        <Description>Technical Details</Description>
        <URL>https://www.amazon.com/Harry-Potter-Deathly-Hallows-Rowling/dp/tech-
data/0545139708%3FSubscriptionId
          %3DAKIAIOSFODNN7EXAMPLE%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative
          %3D386001%26creativeASIN%3D0545139708</URL>
      </ItemLink>
      <ItemLink>
        <Description>Add To Baby Registry</Description>
        <URL>https://www.amazon.com/gp/registry/baby/add-item.html
%3Fasin.0%3D0545139708%26SubscriptionId
          %3DAKIAIOSFODNN7EXAMPLE%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative
          %3D386001%26creativeASIN%3D0545139708</URL>
      </ItemLink>
      <ItemLink>
        <Description>Add To Wedding Registry</Description>
        <URL>https://www.amazon.com/gp/registry/wedding/add-item.html
%3Fasin.0%3D0545139708%26SubscriptionId
          %3DAKIAIOSFODNN7EXAMPLE%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative
          %3D386001%26creativeASIN%3D0545139708</URL>
      </ItemLink>
      <ItemLink>
```

```
        <Description>Add To Wishlist</Description>
        <URL>https://www.amazon.com/gp/registry/wishlist/add-item.html
%3Fasin.0%3D0545139708%26SubscriptionId
          %3DAKIAIOSFODNN7EXAMPLE%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative
          %3D386001%26creativeASIN%3D0545139708</URL>
      </ItemLink>
      <ItemLink>
        <Description>Tell A Friend</Description>
        <URL>https://www.amazon.com/gp/pdp/taf/0545139708%3FSubscriptionId
%3DAKIAIOSFODNN7EXAMPLE
          %26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D386001%26creativeASIN
          %3D0545139708</URL>
      </ItemLink>
      <ItemLink>
        <Description>All Customer Reviews</Description>
        <URL>https://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>https://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." Item details are returned, including the name of the author, illustrator, book manufacturer, and product identifier (ASIN). If you copy the `DetailPageURL` into a browser, the product detail page for that item is displayed.

> **Tip**
> An Amazon Standard Item Number (ASIN) 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, but this practice isn't appropriate for customer applications. The remainder of this section describes how to programmatically issue Product Advertising API requests and process responses.

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. 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. These examples don't work as is because you must include valid request authentication. You might find the following links helpful as you learn more about Product Advertising API and making requests.

- Product Advertising API Scratchpad is a useful web-based application that enables you to quickly submit requests to the Product Advertising API. You can explore the operations and responses using REST queries.
- Authenticating requests — Request Authentication in the *Product Advertising API Developer Guide*
- Developer forum — Product Advertising API forum

> **Note**
> If you are viewing this document online, you can use the **Filter View** drop-down list in the top-right corner of the page to view the example code in your programming language only.

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. 10) 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 generated automatically 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 a `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, including the endpoint, the service name, the access key ID, Associate tag, Product Advertising API version number, and operation name, cannot be changed by the customer. The last two parameters, `SearchIndex` and `Keywords`, are values set by the customer through the user interface. 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 a value for `Keywords`. 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. 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

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 was executed successfully and you can display the information in the response.

If the value is **False**, 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. For example, if 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>
```

Although an `IsValid` value of True specifies that the request was valid and executed, it does not mean 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 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.getRequestProcessingTime());

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 was 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 was 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 was 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

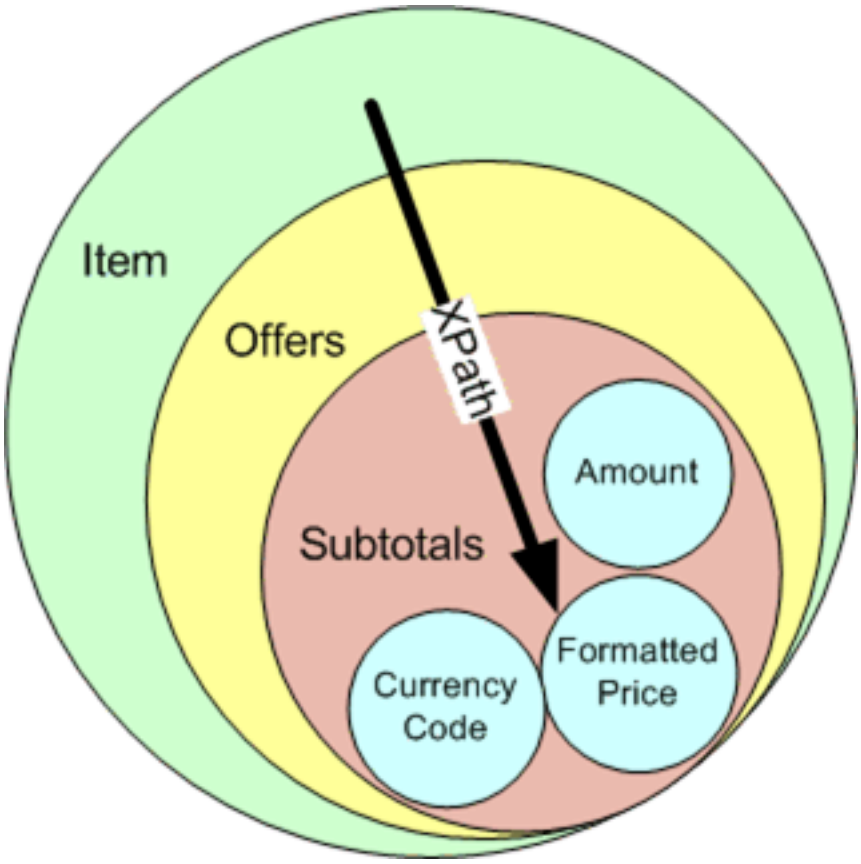
In the previous section, you saw that a Product Advertising API response is an XML document. The returned elements and their values depend on the data stored in Amazon's databases and the response groups specified in the request.

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 can potentially be returned by a response group .

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>
      <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($current->Offers->Offer->OfferListing->Price->FormattedPrice)){
    print("<br>Price:
    ".$current->Offers->Offer->OfferListing->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 same principles used in the example apply to more complex applications.

The Product Advertising API provides opportunities for developing new and innovative applications and websites. Previous sections covered 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. 27)
- Adding an Item to a Shopping Cart (p. 27)
- Purchasing the Item (p. 27)

All of these tasks are covered in more detail in the *Product Advertising API Developer Guide*.

Providing More Item Details

An `ItemSearch` request, which was discussed and implemented in the previous sections, often returns multiple items. Typically, a Product Advertising API application displays a small image of each item along with a short description. Customers often like to pick from the list one or more items that look interesting so they can learn more about them. It is possible to display extended information about each of the items returned by `ItemSearch`, and the length of the web page would grow substantially. For that reason, provide customers 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. For example, `ItemLookup` can return all of the physical details and pricing information of the item.

Adding an Item to a Shopping Cart

After 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. Use `CartAdd` if the customer has a shopping cart.

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 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 the sending of a `PurchaseURL` in a request to Amazon. Every cart operation returns the `PurchaseURL`. It contains all of the information required 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, the Associate will 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 or accessory 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 to purchase a wedding or baby shower gift.

For more information, see Resources (p. 29).

Resources

Use the following resources when working with the Product Advertising API.

Resource	Description
Product Advertising API Scratchpad	Use this tool to send requests and view sample code responses.
Product Advertising API Developer Guide	Read the complete documentation for the Product Advertising API.
Best Programming Practices	Follow this checklist of best practices.
Discussion Forums	Join the community of developers who are using our service.
Conditions of Use	Read detailed information about the copyright and trademark usage at Amazon.com and other topics.
Contact Us	Contact us for inquiries concerning billing, accounts, events, abuse, and more.

See the Product Advertising API website for your locale:

Locale	URL
Canada	https://associates.amazon.ca/gp/advertising/api/detail/main.html
China	https://associates.amazon.cn/gp/advertising/api/detail/main.html
France	http://partenaires.amazon.fr/gp/advertising/api/main.html
Germany	http://partnernet.amazon.de/gp/advertising/api/main.html
India	https://affiliate-program.amazon.in/gp/advertising/api/detail/main.html
Italy	https://programma-affiliazione.amazon.it/gp/advertising/api/detail/main.html
Japan	https://affiliate.amazon.co.jp/gp/advertising/api/detail/main.html
Mexico	https://afiliados.amazon.com.mx/gp/advertising/api/detail/main.html
Spain	https://afiliados.amazon.es/gp/advertising/api/detail/main.html
United Kingdom	https://affiliate-program.amazon.co.uk/gp/advertising/api/detail/main.html
United States	https://affiliate-program.amazon.com/gp/advertising/api/detail/main.html

Document History

See the most recent changes to the documentation.

API version: 2013-08-01

Latest documentation update: February 10, 2017

Change	Description	Release Date
Product Advertising API supports IAM user credentials	You can create an IAM user and attach a policy to allow your IAM users permission to use the Product Advertising API. For more information, see Becoming a Product Advertising API Developer (p. 4).	10 February 2017
Updated Scratchpad tool	Scratchpad now supports all operations and returns sample code responses. Use this tool to generate sample code and help debug your requests. See the updated Product Advertising API Scratchpad.	8 September 2015
New Marketplace	The MX (Mexico) marketplace was added.	25 August 2015
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
New Marketplace updates	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. 13).	