Product Advertising API

Getting Started Guide

API Version 2009-07-01

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

Table of Contents

Welcome 1
What's New 4
Introduction to the Product Advertising API 5
Getting Set Up 7
- Getting an AWS Access Key ID 7
- Becoming an Associate 7
- Getting the Tools You Need 8
- Your Development Environment 9

Making Requests 12
- Submitting Your First Request 12
- Parts of a Request 15
- Implementing an Product Advertising API Request 15

Processing Responses 20
- Checking Request Execution 20
- Processing Overview 22
- Processing Implementations 23

Next Steps 27
- Providing More Item Details 27
- Adding An Item to a Shopping Cart 28
- Purchasing the Item 28
- Where to Go From Here 28

Document Conventions 29
Index 32

Welcome

Topics

- Audience (p. 1)
- Reader Feedback (p. 1)
- How to Use this Guide (p. 2)
- Product Advertising API Resources (p. 2)

This is the *Product Advertising API Getting Started Guide*. This section describes who should read this guide, how the guide is organized, and other resources related to Product Advertising API.

Audience

This guide is intended for developers who want to build an e-commerce storefront that sells items listed on www.amazon.com, or an application that helps others build e-commerce storefronts.

Required Knowledge and Skills

Use of this guide assumes you are familiar with the following:

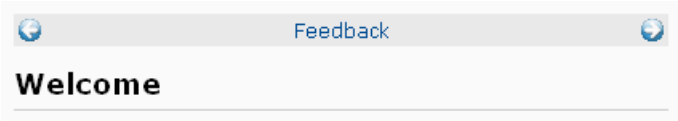
- XML (For an overview, go to the *W3 Schools XML Tutorial*)
- Basic understanding of web services (For an overview, go to the *W3 Schools Web Services Tutorial*)

In addition, for this guide, you need to be familiar with one of the following programming languages:

- PHP
- C#
- Java
- Perl

Reader Feedback

The online version of this guide provides a link that enables you to enter feedback about this guide. We strive to make our guides as complete, error free, and easy to read as possible. You can help by giving us feedback. Thank you in advance!



How to Use this Guide

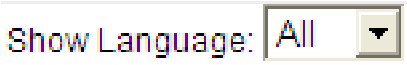
This guide is organized as a high-level introduction and tutorial. It is divided into several major sections that allow you to practice using Product Advertising API in a simple environment. Each section builds on the previous sections, so that if you read and work through the examples in sequence, you'll get a basic understanding of the Product Advertising API and also have a simple working application.

The major sections of this guide are:

- Getting Set Up (p. 7)--Describes the how to get the identifiers you need to submit requests.
- Making a Request (p. 12)--Describes how to send a simple Product Advertising API request in multiple programming languages.
- Processing Responses--Describes how to parse the response to your request.
- Next Steps (p. 27)--Describes what to do next in learning about Product Advertising API.

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 table below lists related resources that you'll find useful as you work with this service.

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 Release Notes	The Release Notes give a high-level overview of the current release. They specifically note any new features, corrections, and known issues.
Product Advertising API Developer Resource Center	A central starting point find documentation, code samples, release notes, and other information to help you build innovative applications with AWS.
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.

Resource	Description
Amazon Web Services Contact Form.	This HTML form is *only* for Amazon Web Services account questions. For technical questions, use the Discussion Forums.

What's New

This What's New is associated with the 2009-07-01 version of the Product Advertising API. This guide was last updated on 2009-08-14.

The following table describes the important changes since the last release of the *Product Advertising API Getting Started Guide*.

Change	Description
Enhancement	Added narrative about how to generate a signed request. For more information, see Submitting Your First Request (p. 12).
Rebranding	This guide has been updated to reflect the change in the service's name.
Version update	This guide has been updated to comply with the latest WSDL.

Introduction to the Product Advertising API

Topics

- Overview of Product Advertising API (p. 5)
- Key Concepts (p. 6)
- Overview of Examples (p. 6)

This introduction to Product Advertising API is intended to give you a high-level overview of this web service. After reading this section, you should understand the basics you need to work through the examples in this guide.

Overview of Product Advertising API

Amazon has spent over ten years and hundreds of millions of dollars developing 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 much of the data that is used by Amazon, 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. 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. In 2006, Product Advertising API developers sold well over $600 million worth of items. Would you like a percentage of that revenue?

Features

Product Advertising API provides the following major features:

- **Access to Amazon's product catalog—**Product Advertising API provides access to Amazon's product database

- **Access to customer and seller reviews—**Product Advertising API provides access to Amazon's customer and seller review database
- **Display product images—**Display product images used on www.amazon.com
- **Latest offerings—**Access the latest Amazon offerings, including digital media

This guide presents a simple example of an `ItemSearch` request in which search criteria are specified and items matching that criteria are returned.

Key Concepts

The `ItemSearch` function can take a wide variety of parameters that help narrow down 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 return which to display. The Offer response group, for example, returns information about an item's offer, that is, it's price and availability.

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.

Requests are structured. Each has an endpoint, which is the URL of the Product Advertising API, ecs.amazonaws.com. Responses, by default, are returned in XML, which makes parsing the reponse easier.

Overview of Examples

One of the first and most common tasks a customer undertakes is searching for items to buy. They supply search parameters, such as if 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.

Getting Set Up

Topics

- Getting an AWS Access Key ID (p. 7)
- Becoming an Associate (p. 7)
- Getting the Tools You Need (p. 8)
- Your Development Environment (p. 9)

There are a number of tasks that you must complete before you can submit your first Product Advertising API request. This section explains those tasks in the following sections.

Getting an AWS Access Key ID

Every Product Advertising API request requires that you include your AWS access identifiers (your AWS Access Key ID and your AWS Secret Access Key). These are alphanumeric tokens distributed by Amazon that uniquely identify the request sender. Before proceeding in this guide you must obtain your identifiers by signing up for a free Amazon Web Services account. To do so, go to http://aws.amazon.com. Once you have a Web Services account, you can retrieve your AWS Access Key ID:

To retrieve your access identifiers

1. Go to the AWS website, aws.amazon.com.
2. Point to the **Your Web Services Account** button.
   A drop down list displays.
3. Click **View Access Key Identifiers**.

Caution

The web page displays your AWS Access Key ID and Secret Access Key.

Becoming an Associate

An Associate earns commissions by using their own website to refer sales to www.amazon.com. To get the 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.

To become an Associate

1. Register with Amazon Payments at http://s1.amazon.com/exec/varzea/register/login/ .
   To receive money from Amazon for your work as an Associate, you must register with Amazon Payments.
2. 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
Canada	https://associates.amazon.ca/
France	https://partenaires.amazon.fr/
Germany	http://partnernet.amazon.de
Japan	https://affiliate.amazon.co.jp/
United Kingdom	https://affiliate-program.amazon.co.uk
United States	http://affiliate-program.amazon.com/

Getting the Tools You Need

Program 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 appropriate website where you can download and install the appropriate toolkit.

Language	API Style	Tools Used
Java	SOAP	• Java 6 or higher Make sure the *PATH* environment variable points at the Java installation. • Eclipse 3.2 or higher 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.
C#	SOAP	• Microsoft Visual Studio 2005 C# Express Edition • .NET Framework2.0

Language	API Style	Tools Used
Perl	REST (using HTTP POST)	To download the modules used in the following Perl example, go to CPAN website: • `Digest::HMAC_SHA1` • `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 are also welcome to run the 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. 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.

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:

- Java Setup (p. 9)
- C# Setup (p. 10)
- Perl Setup (p. 11)
- PHP Setup (p. 11)

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 Show Language list on the top-right corner of the page.

Java

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 will use the wsimport utility in Java 6 to generate the stubs from the Product Advertising API WSDL, which is located at *http://ecs.amazonaws.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 will go 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://ecs.amazonaws.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 you are an IDE that 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://
   ecs.amazonaws.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 webservice.
- 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#

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.The dialog box 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. From the **Project** menu, select **Add Web Reference**.

   A dialog box opens.
2. Enter the WSDL URL for Product Advertising API in the URL box.

   For example, 2008-06-28.
3. Click **Go**.

   The main pane in the dialog box shows the API.
4. 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

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

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 Requests

Topics

- Submitting Your First Request (p. 12)
- Parts of a Request (p. 15)
- Implementing an Product Advertising API Request (p. 15)

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.

Note

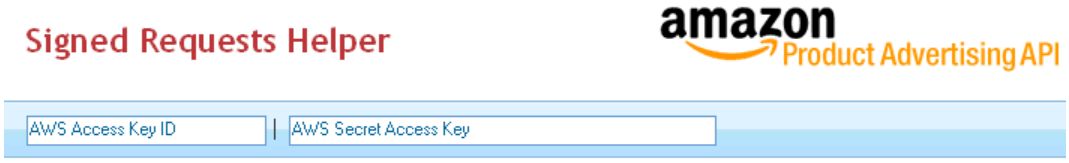
Processing responses is discussed in the next section.

The following sections describe how to make an Product Advertising API request using multiple programming languages:

Submitting Your First Request

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.



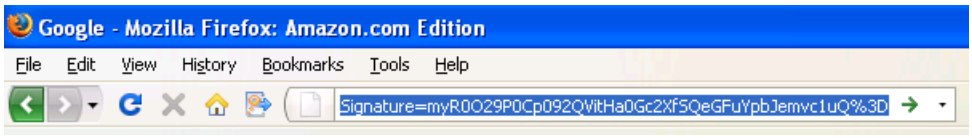
3. In the **Unsigned URL** text box, enter the following:

```
http://ecs.amazonaws.com/onca/xml?Service=AWSECommerceService
&Version=2009-03-31
&Operation=ItemSearch
```

```
&SearchIndex=Books
&Keywords=harry+potter
```

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

6. In a new browser window or tab, paste the signed url into the address line, and press the Enter button on your keyboard.



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 a list of all locales and endpoints, go to 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-
Rowling/dp/0545139708%3FSubscriptionId
      %3D1GPD3QT81GH6YJ5AV5G2%26tag%3Dws%26linkCode%3Dxm2%26camp
%3D2025%26creative
      %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
          %3D1GPD3QT81GH6YJ5AV5G2%26tag%3Dws%26linkCode%3Dxm2%26camp
%3D2025%26creative
          %3D386001%26creativeASIN%3D0545139708</URL>
      </ItemLink>
```

```
      <ItemLink>
        <Description>Add To Baby Registry</Description>
        <URL>http://www.amazon.com/gp/registry/baby/add-item.html
%3Fasin.0%3D0545139708%26SubscriptionId
          %3D1GPD3QT81GH6YJ5AV5G2%26tag%3Dws%26linkCode%3Dxm2%26camp
%3D2025%26creative
          %3D386001%26creativeASIN%3D0545139708</URL>
      </ItemLink>
      <ItemLink>
        <Description>Add To Wedding Registry</Description>
        <URL>http://www.amazon.com/gp/registry/wedding/add-item.html
%3Fasin.0%3D0545139708%26SubscriptionId
          %3D1GPD3QT81GH6YJ5AV5G2%26tag%3Dws%26linkCode%3Dxm2%26camp
%3D2025%26creative
          %3D386001%26creativeASIN%3D0545139708</URL>
      </ItemLink>
      <ItemLink>
        <Description>Add To Wishlist</Description>
        <URL>http://www.amazon.com/gp/registry/wishlist/add-item.html
%3Fasin.0%3D0545139708%26SubscriptionId
          %3D1GPD3QT81GH6YJ5AV5G2%26tag%3Dws%26linkCode%3Dxm2%26camp
%3D2025%26creative
          %3D386001%26creativeASIN%3D0545139708</URL>
      </ItemLink>
      <ItemLink>
        <Description>Tell A Friend</Description>
        <URL>http://www.amazon.com/gp/pdp/taf/0545139708%3FSubscriptionId
%3D1GPD3QT81GH6YJ5AV5G2
          %26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative
%3D386001%26creativeASIN
          %3D0545139708</URL>
      </ItemLink>
      <ItemLink>
        <Description>All Customer Reviews</Description>
        <URL>http://www.amazon.com/review/
product/0545139708%3FSubscriptionId
          %3D1GPD3QT81GH6YJ5AV5G2%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
          %3D1GPD3QT81GH6YJ5AV5G2%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 Request

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

This section shows you how to implement an `ItemSearch` request in various programming languages. For more complete code samples, go to the Product Advertising API Sample Code & Libraries page.

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 Show Language list on the top-right corner of the page.

Important

These examples do not include request authentication.

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 the Using the Product Advertising API Java Client Side Library (p. 10) section 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("2008-08-19");
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.

```
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 ItemSearch wrapper
ItemSearch search = new ItemSearch();
search.AssociateTag = "[Your Associate ID]";
search.AWSAccessKeyId = "[Your ID]";
search.Version = "2008-08-19";

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

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

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

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


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

Perl

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://ecs.amazonaws.com/onca/xml";
my $service = "AWSECommerceService";
my $accesskey = "[INSERT YOUR ACCESS KEY ID HERE]";
my $operation = "ItemSearch";
my $version = "2008-08-19";

# Assemble the REST request URL.
my $request =
```

```
    "$EndPoint?" .
    "Service=$service&" .
    "AWSAccessKeyId=$accesskey&" .
    "Operation=$operation&" .
    "Keywords=$keywords&" .
    "SearchIndex=$searchIndex&" .
    "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 = "2008-08-19";
$ResponseGroup = "ItemAttributes,Offers";
//User interface provides values
//for $SearchIndex and $Keywords

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

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

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

Topics

- Checking Request Execution (p. 20)
- Processing Overview (p. 22)
- Processing Implementations (p. 23)

Now that you've successfully sent an 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.

Checking Request Execution

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, CustomerContentLookup, ItemLookup,
 SimilarityLookup, SellerLookup, ItemSearch, BrowseNodeLookup, CartModify,
 ListSearch, CartClear, CustomerContentSearch, CartCreate, TransactionLookup,
 CartAdd, SellerListingSearch, Help.
  </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 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 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 recieved!");
```

```
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";
}
```

Processing Overview

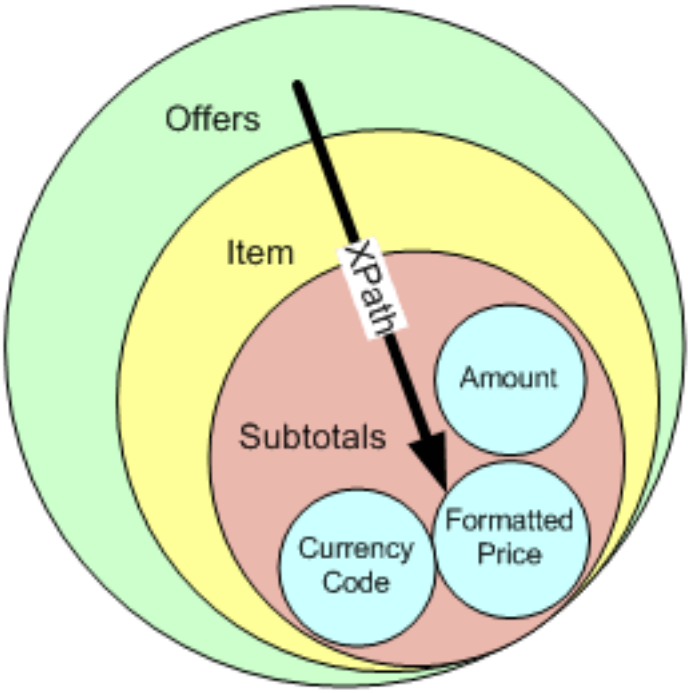
In the previous section, you saw that an Product Advertising API response is an XML document. The elements returned and their values depend on the data stored in Amazon's databases and the response groups specified in the request.

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>
      <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){...}
```

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 occured: " + 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

Topics

- Providing More Item Details (p. 27)
- Adding An Item to a Shopping Cart (p. 28)
- Purchasing the Item (p. 28)
- Where to Go From Here (p. 28)

Congratulations! Now that you have completed the basic example presented in this guide, you are ready to start designing your own 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 can be readily applied to more complex applications.

The Product Advertising API 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 an 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 an Product Advertising API application.

All of these tasks are covered in greater depth in the `Product Advertising API Developer Guide`.

Note

The sample application, SimpleStore.php, provides a sample implementation of many of these tasks. To see this application, go to 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, an 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 an 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 only covers 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.

To find out more about these options, go to 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 Conventions

This section lists the common typographical and symbol use conventions for AWS technical publications.

Typographical Conventions

This section describes common typographical use conventions.

Convention	Description/Example
Call-outs	A call-out is a number in the body text to give you a visual reference. The reference point is for further discussion elsewhere. You can use this resource regularly. 
Code in text	Inline code samples (including XML) and commands are identified with a special font. You can use the command `java -version`.

Code blocks	Blocks of sample code are set apart from the body and marked accordingly.
	
	`# ls -l /var/www/html/index.html`
	`-rw-rw-r--  1 root root 1872 Jun 21 09:33 /var/www/html/index.html`
	`# date`
	`Wed Jun 21 09:33:42 EDT 2006`

Emphasis	Unusual or important words and phrases are marked with a special font. You *must* sign up for an account before you can use the service.
Internal cross references	References to a section in the same document are marked. See Document Conventions (p. 29).
Logical values, constants, and regular expressions, abstracta	A special font is used for expressions that are important to identify, but are not code. If the value is `null`, the returned response will be `false`.

Convention	Description/Example
Product and feature names	Named AWS products and features are identified on first use. Create an *Amazon Machine Image* (AMI).
Operations	In-text references to operations. Use the `GetHITResponse` operation.
Parameters	In-text references to parameters. The operation accepts the parameter *`AccountID`*.
Response elements	In-text references to responses. A container for one `CollectionParent` and one or more `CollectionItems`.
Technical publication references	References to other AWS publications. If the reference is hyperlinked, it is also underscored. For detailed conceptual information, see the *Amazon Mechanical Turk Developer Guide*.
User entered values	A special font marks text that the user types. At the password prompt, type **`MyPassword`**.
User interface controls and labels	Denotes named items on the UI for easy identification. On the **File** menu, click **Properties**.
Variables	When you see this style, you must change the value of the content when you copy the text of a sample to a command line. % ec2-register <*`your-s3-bucket`*>/image.manifest See also the following symbol convention.

Symbol Conventions

This section describes the common use of symbols.

Convention	Symbol	Description/Example
Mutually exclusive parameters	(Parentheses | and | vertical | bars)	Within a code description, bar separators denote options from which one must be chosen. `% data = hdfread (start | stride | edge)`

Optional parameters
XML variable text
	[square brackets]	Within a code description, square brackets denote completely optional commands or parameters.
		`% sed [-n, -quiet]`
		Use square brackets in XML examples to differentiate them from tags.
		`<CustomerId>[ID]</CustomerId>`

Variables	<arrow brackets>	Within a code sample, arrow brackets denote a variable that must be replaced with a valid value. `% ec2-register <your-s3-bucket>/image.manifest`

Index

A

add item to cart, 28
Associate
  becoming one, 7
AWS Access Key ID, 7
AWS Secret Access Key, 7
AWSECommerceService, 10
AWSECommerceServicePortType, 10

C

C#, 8, 10, 16, 21, 24
  error, 21
cart, adding item to, 28

D

developer tools, 8

E

error, 20

G

generated file types, 10

I

IsValid, 20
item, purchasing , 28

J

Java, 8, 15, 21, 23
  error, 21
Java Client Side Library, 9

L

locale, 8, 13

O

Order Pipeline, 28

P

Perl, 9, 11, 17, 22, 24
  error, 22
PHP, 9, 11, 18, 22, 25
  error, 22
Proceed to Checkout, 28
providing item details, 27
purchase item, 28
PurchaseURL, 28

R

request
  C#, 16
  Java, 15, 23
  making a, 12
  parts of, 15
  Perl, 17
  PHP, 18
  processing, 20, 22
    with C#, 24
    with Java, 23
    with Perl, 24
    with PHP, 25
  submitting, 12
  validating execution, 20
response, 13
response group, 22

S

Secret Access Code, 7
shopping cycle, 28
SOAP proxy, 10
stubs, generating, 10

T

tools for developer, 8

W

Web Services account, 7

X

XML, 13
  document, 12
XPath , 22

Y

Your Web Services Account button, 7