Amazon Fulfillment Web Service

Getting Started Guide

Version 1.1

Amazon Fulfillment Web Service: Getting Started Guide

Table of Contents

Welcome

Topics

Welcome to the *Amazon Fulfillment Web Service* Getting Started Guide. Amazon Fulfillment Web Service is a set of operations designed to help you automate the process of shipping your inventory to an Amazon fulfillment center and—if you sell on a site other than Amazon—to ship your items from the fulfillment center to the seller. If you sell on Amazon, this last part is done automatically.

This guide is designed to help you understand how to make requests to the the Amazon Fulfillment Web Service, and present some ideas for using this service. This section describes who should read this guide, how the guide is organized, and other resources related to the Amazon Fulfillment Web Service.

The Amazon Fulfillment Web Service will occasionally be referred to within this guide as simply "Amazon FWS"; all copyrights and legal protections still apply.

Audience

This guide is intended for developers who want a quick introduction to Amazon FWS and who want to build fulfillment properties into an already-existing web site or web application, such as an inventory system.

Required Knowledge and Skills

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

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

In addition, you need to be familiar with one of the following programming languages, or be willing to work with a language-agnostic scratchpad tool.

- Java
- C#

Business Requirements

To use Amazon FWS, you need to create a merchant account with Amazon.com. This is a requirement because you have to list an item to sell in order to have Amazon fulfill it. Only merchants can list an item. The process of registering for a merchant account is documented in Registering for a Merchant Account (p. 11).

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 explains how to implement an Amazon FWS operation and process the response using multiple programming languages. You can use these examples as a model for implementing all of the other Amazon FWS operations in a more sophisticated web site or application.

The information in this guide is meant to be read sequentially. Each concept and implementation detail builds upon the previous ones. We recommend, therefore, that you read this short guide in the order presented. If you are already an AWS developer, we note steps that you can skip.

The major sections of this guide are:

"What's New" lists the changes in this guide since the last version.

"Introduction to Amazon FWS" introduces you to Amazon FWS and summarizes its main features.

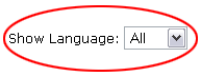
"Getting Set Up" describes what you need to do to prepare to send your first Amazon FWS request. It includes getting an AWS developer account, signing up for Amazon FWS, and getting the tools you need to run the samples.

"Making Requests to Amazon FWS" walks you through running the samples that we provide.

"Where to Go from Here" offers some ideas for you to learn more about Amazon FWS.

Showing Your Preferred Programming Language

In the HTML version of this document, 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.



Your selection applies to all of the pages in this guide.

Amazon FWS Resources

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

Resource	Description
Amazon Fulfillment Web Service Developer Guide	The Developer Guide provides a detailed discussion of the service. It includes an architectural overview and a programming reference.
Amazon FWS API Reference	The API Guide is a reference for the operations and associated parameters used in the inbound, outbound, and inventory APIs.
Amazon FWS Inbound Quick Reference Card	The inbound quick reference card gives a concise listing of parameters used for the inbound operations
Amazon FWS Outbound Quick Reference Card	The outbound quick reference card gives a concise listing of parameters used for the outbound operations
Amazon FWS Inventory Quick Reference Card	The inventory quick reference card gives a concise listing of parameters used for the inventory operations
Amazon FWS Release Notes	The Release Notes give a high-level overview of the current release. They specifically note any new features, corrections, and known issues.
Seller Central Help	The Seller Central Help page contains topical help and answers to common questions for tasks relating to selling on Amazon and using Amazon for fulfillment.
AWS Developer Resource Center	A central starting point to find documentation, code samples, release notes, and other information to help you build innovative applications with AWS.
Discussion Forums	A community-based forum for developers to discuss technical questions related to Amazon Web Services.
AWS Support Center	The home page for AWS Technical Support, including access to our Developer Forums, Technical FAQs, Service Status page, and (if you're subscribed to this program) AWS Premium Support.

Resource	Description
Product information about Amazon FWS	The primary web page for information about Amazon FWS.
Contact Us	A central contact point for inquiries concerning AWS billing, account, events, abuse, etc.
Conditions of Use	Detailed information about the copyright and trademark usage at Amazon.com and other topics.

What's New

This What's New is associated with the 1.1 release of the *Amazon Fulfillment Web Service*. This guide was last updated on August 09, 2010.

The following table describes the important changes since the last release of the Amazon Fulfillment Web Service Getting Started Guide.

Change	Description	Release Date
Update	Minor changes and edits.	04 June 2010
Update	Changed the structure of this guide to help developers who already use AWS services.	01 June 2009
First Release	This is the first release of the Amazon Fulfillment Web Service Getting Started Guide.	19 March 2008

Introduction to Amazon FWS

The Amazon Fulfillment Web Service provides a way for web service applications to use Amazon's fulfillment operations. Merchants send their inventory to Amazon fulfillment centers. When customers place orders, Amazon picks, packs, and ships the products for the merchants. Our tools give merchants the freedom and control they need over their inventory without their having to manage the process of fulfilling orders themselves.

Amazon FWS works for orders placed on Amazon.com or through other merchant sales channels. You can request Amazon to fulfill an order with inventory stored in our warehouse at any time, to any place. By combining Amazon FWS and a direct web site integration, you can sell products on your own web site or other sales channels, and automatically create fulfillment orders directly to our warehouse, thereby creating a nearly virtual business.

The following image displays the entire workflow of Fulfillment by Amazon.

How It Works

1	You send your new or used products to Amazon's fulfillment centers.
2	Amazon stores your products in its ready-to-ship inventory.
3	Amazon fulfills orders on your behalf.
4	Amazon picks your products from inventory and packages them.

5	Amazon ships the products to your customers from the network of Amazon fulfillment centers.

For more information about Fulfillment by Amazon, go to the Fulfillment by Amazon home page.

Three APIs

Amazon FWS consists of three APIs (inbound, outbound, and inventory) that handle different aspects of the fulfillment process.

The inbound API deals with the first two steps in the fulfillment process: sending products to Amazon, and having Amazon store those products.

The inbound API helps you get your inventory to one of Amazon's fulfillment centers. Once your items are in a fulfillment center, you can sell them on Amazon.com or you can sell them on another site. If you plan to sell only on Amazon.com, you only need to use the inbound API. Amazon.com handles all the outbound fulfillment for you. However, if you plan to sell on channels besides Amazon.com, you need both the inbound and the outbound APIs.

The outbound API deals with the last three steps in the fulfillment process: telling Amazon about the items you've sold and where to send the items.

The outbound API helps you integrate Fulfillment by Amazon with any payment processing application or inventory management system currently in use. Amazon FWS does not receive any payment information from your sales. Rather you use the outbound API to tell the Amazon Fulfillment Network to ship a set of items to a recipient. Use the outbound API if you sell on a channel other than Amazon.com or if you require fulfillment for something other than a sale (samples, gifts, or rewards).

Note

If you only sell on Amazon.com, you don't need to use the outbound API. When customers buy your items on Amazon.com, the Amazon Fulfillment Network automatically packs and ships the items for you.

Features

Amazon FWS provides the following major features:

- **Creating inbound shipments to the fulfillment center—**You can streamline the process for creating and labeling the items you ship to a fulfillment center
- **Checking status of inbound shipments—**You can check to see if your shipment has reached a fulfillment center and, if so, whether the shipment has been processed
- **Submitting fulfillment orders—**When you integrate with our API, your customers can submit orders at any time and won't experience lag time while waiting for you to process the orders
- **Tracking and managing outbound shipment requests—**You can track shipments and keep your customers aware of arrival times

About the Samples

The sample libraries available for Amazon FWS cover all possible operations available for the service. However, the samples included in this guide focus specifically on the core fulfillment operations:

- Checking the service's status
- Creating a shipment
- Creating a fulfillment order

For more information about the examples, see Making Requests to Amazon FWS (p. 20).

Getting Set Up

Topics

This section walks you through each of the tasks you must complete before you can submit a request to Amazon Fulfillment Web Service. They are presented in the best order to follow so that you can run the samples as quickly as possible. The following tables shows the sections you need to read if you're already an AWS user, or if you're brand new to AWS.

Existing AWS User	New AWS User
1. Signing Up for Amazon FWS (p. 10) 2. Registering for a Merchant Account (p. 11) 3. Getting the Tools You Need (p. 13) 4. Getting the Sample Code (p. 15)	1. Creating an AWS Account (p. 9) 2. Signing Up for Amazon FWS (p. 10) 3. Registering for a Merchant Account (p. 11) 4. Getting Your AWS Access Identifiers (p. 12) 5. Getting the Tools You Need (p. 13) 6. Getting the Sample Code (p. 15)

Creating an AWS Account

Tip

If you're already an AWS service user, you can skip directly to Signing Up for Amazon FWS (p. 10).

Amazon FWS is one of the products offered by Amazon Web Services (AWS). To access any product AWS offers, you must first create an AWS account at http://aws.amazon.com. An AWS account is simply

an Amazon.com account that is enabled to use AWS products. If you want, you can use an existing Amazon.com account login and password when creating the AWS account.

Important

To use the Amazon Fulfillment Web Service, you must use the same email address for both your AWS account and your merchant account. For more information about setting up a merchant account, see Registering for a Merchant Account (p. 11).

From your AWS account you can view your AWS account activity, view usage reports, and manage your AWS account access identifiers.

To set up a new account

1. Go to http://aws.amazon.com.
2. In the **Sign Up for AWS** box, click **Sign up now**.
   The **Sign In** page is displayed.
3. Enter your e-mail address and select the button for **No, I am a new customer**, and click **Continue**.

   Important

   If you already have a merchant account, use that address. If you don't, you need to use the same e-mail account for both AWS and the merchant account. You will sign up for one in a subsequent step.

   The next page asks for the name you want associated with the account (e.g., Joe Smith) and a password for the account. If you're using an e-mail address already in the Amazon.com system, the page indicates that this is an existing address, but still lets you create a new account with it.
4. Enter the name you want associated with the account and a password, and click **Continue**.
   The **Account Info** page is displayed.
5. Enter your contact information and select how you learned about AWS. Then read the AWS Customer Agreement, select the check box to indicate that you've read and agree to the terms of the customer agreement, and click **Continue**.
   The process is complete and you've created your new AWS account.

At this point, you have an AWS account, but you're not signed up to use Amazon FWS yet. For instructions, see Signing Up for Amazon FWS (p. 10).

Signing Up for Amazon FWS

Before you can use Amazon FWS, you must sign up to use the service. You must already have an AWS account (for more information, see Creating an AWS Account (p. 9)).

To sign up for Amazon FWS

1. Go to the Amazon FWS page.
2. Click **Sign Up for This Web Service** on the top right corner of the page.

AWS sends you a confirmation e-mail.

At this point, you have an AWS account and are signed up to use Amazon FWS. Next you need to register for a merchant account with Amazon.com. For instructions, see Registering for a Merchant Account (p. 11).

Registering for a Merchant Account

Amazon FWS is the API for Fulfillment By Amazon (FBA). Anyone using FBA in any form (API, web interface, or feeds) must have a merchant account with Amazon.com. There are two types of merchant accounts you can sign up for:

- **Media—**Sign up for this account if you want to sell books, music, DVDs, or video cassettes
- **Non-media—**Sign up for this account if your products fall into a variety of other categories, such as automotive, baby, software, sports, or video games (there is a complete list when you register)

For the purpose of the sample code in this guide, we'll register for a non-media merchant account.

Tip

If you already have a non-media merchant account and have signed up for FBA, you can skip this section entirely.

If you already have a merchant account, you'll need to sign up for FBA. If you don't already have a merchant account, skip to the next procedure.

To sign up for FBA using an existing merchant account

1. Go to the Amazon Services Register Now page.
2. Click the appropriate **add FBA to your account** link.
3. Sign in to your account.

If you don't have a merchant account yet, you can sign up for one together with FBA in the following procedure. If you already have a merchant account and have signed up for FBA using the previous procedure, you can skip this procedure entirely.

To sign up for a non-media merchant account and FBA

1. Go to the Amazon Services Register Now page.
2. Be sure that the **Selling on Amazon** checkbox is selected.
3. Select the checkbox for **Fulfillment by Amazon** to enable Amazon to fulfill your merchant items.

   Important

   Be sure to select this checkbox. If you don't select **Fulfillment by Amazon** , you will be signed up for a merchant account, but not for FBA.

4. Click the **Sign-up for Selected Services** button.

   The Amazon Services Seller Central page displays.
5. Fill out the fields on the page as instructed.

At this point, you have a merchant account and are signed up for FBA. Now your're ready to work with the sample code for Amazon FWS. However to do so, you'll need your AWS access identifiers. For more information, see Getting Your AWS Access Identifiers (p. 12).

Getting Your AWS Access Identifiers

Tip

If you already have your Secret Access Key and AWS Access Key ID, you can skip directly to Getting the Tools You Need (p. 13).

To run the examples used in this guide, you'll need your *AWS access key identifiers*, which AWS assigned you when you created your AWS account. The following are the AWS access key identifiers:

- Access Key ID (a 20-character, alphanumeric sequence)
  For example: 022QF06E7MXBSH9DHM02
- Secret Access Key (a 40-character sequence)
  For example: kWcrlUX5JEDGM/LtmEENI/aVmYvHNif5zB+d9+ct

Caution

Your Secret Access Key is a secret, which only you and AWS should know. It is important to keep it confidential to protect your account. Store it securely in a safe place. Never include it in your requests to AWS, and never e-mail it to anyone. Do not share it outside your organization, even if an inquiry appears to come from AWS or Amazon.com. No one who legitimately represents Amazon will ever ask you for your Secret Access Key.

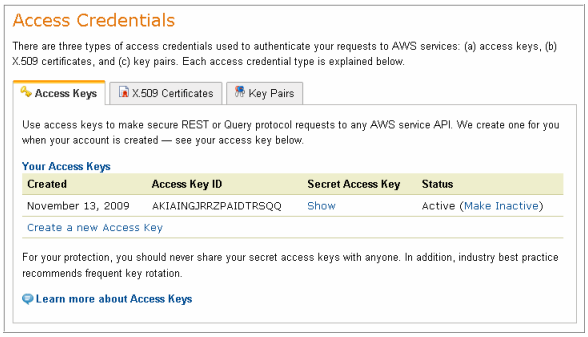
The Access Key ID is associated with your AWS account. You include it in AWS service requests to identify yourself as the sender of the request.

The Access Key ID is not a secret, and anyone could use your Access Key ID in requests to AWS. To provide proof that you truly are the sender of the request, you also include a digital signature calculated using your Secret Access Key. The sample code handles this for you.

Your Access Key ID and Secret Access Key are displayed to you when you create your AWS account. They are not e-mailed to you. If you need to see them again, you can view them at any time from your AWS account.

To get your AWS access key identifiers

1. Go to the Amazon Web Services web site at http://aws.amazon.com.
2. Point to **Your Account** and click **Security Credentials**.
3. Log in to your AWS account.
   The **Security Credentials** page is displayed.
4. Your Access Key ID is displayed in the **Access Identifiers** section of the page.
5. To display your Secret Access Key, click **Show** in the **Secret Access Key** column.



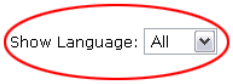
Getting the Tools You Need

Topics

If you want to use the sample code that goes with this guide, you must have one of the programming tools listed in this section for the language of your choice.

Tip

In the HTML version of this document, 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.



Your selection applies to all of the pages in this guide.

Java

To use the Java sample code, you must have the Java 2 Platform Standard Edition 5.0 Development Kit (JDK 5.0) or newer (for more information, go to http://java.sun.com/j2se/1.5.0/download.jsp). If your version of JDK is older than 5.0, you must install newer version.

To verify you have the correct JDK installed

- Open a console window and type the following

  ```
  java -version
  ```

  The response should be similar to the following.

  ```
  java version "1.5.0_05"
  Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_05-b05)
  Java HotSpot(TM) Client VM (build 1.5.0_05-b05, mixed mode, sharing)
  ```

To install the JDK

1. Follow the instructions on the Sun Java web site to install the JDK (for the instructions, go to http://java.sun.com/j2se/1.5.0/install.html).
2. Set your JAVA_HOME environment variable to point to your JDK installation location.

Additionally, you should have the Eclipse IDE for Java Developers. For more information, go to http://www.eclipse.org/.

C#

To use the C# sample code, you must have the following tools from Microsoft:

- Microsoft Visual Studio (for more information, go to http://msdn.microsoft.com/vstudio) or Microsoft Visual C# Express Edition (for more information, go to http://msdn2.microsoft.com/en-us/express/aa975050.asp)
- Windows Service Enhancements (WSE) 3.0 for Microsoft .NET (for more information, go to http://msdn2.microsoft.com/en-us/webservices/Aa740663.aspx). This guide assumes all options are installed.
- NET Framework 2.0 (for more information, go to http://msdn.microsoft.com/netframework)

When you download the tools, follow the instructions that Microsoft provides for installing them.

Scratchpad

To use the Scratchpad for the samples, you'll need to download the Scratchpad utility. For more information, go to Amazon FWS Community Code page and download both the inbound and the outbound Scratchpad utilities.

When you download the file, unzip the contents to a working directory. This file is a self-contained file directory, so you can unzip it anywhere. You do not need to compile anything to use the Scratchpad.

Tip

Internet Explorer 7 returns an error response to Scratchpad requests.

To make Internet Explorer 7 work with Scratchpad

1. Click **Tools > Internet Options**.
2. In the popup window click the **Advanced** tab.
3. In the **Settings** section, scroll down to the Browsing area and uncheck the **Show friendly HTTP error messages** box.
4. Click the **OK** button.

Now that you have the programming tools you need, let's get the sample code.

Getting the Sample Code

The sample code that goes with this guide is posted in the Amazon Fulfillment Web Service Community Code area of the AWS Resource Center. To use this code you'll also need to download the associated libraries for your programming language.

Java

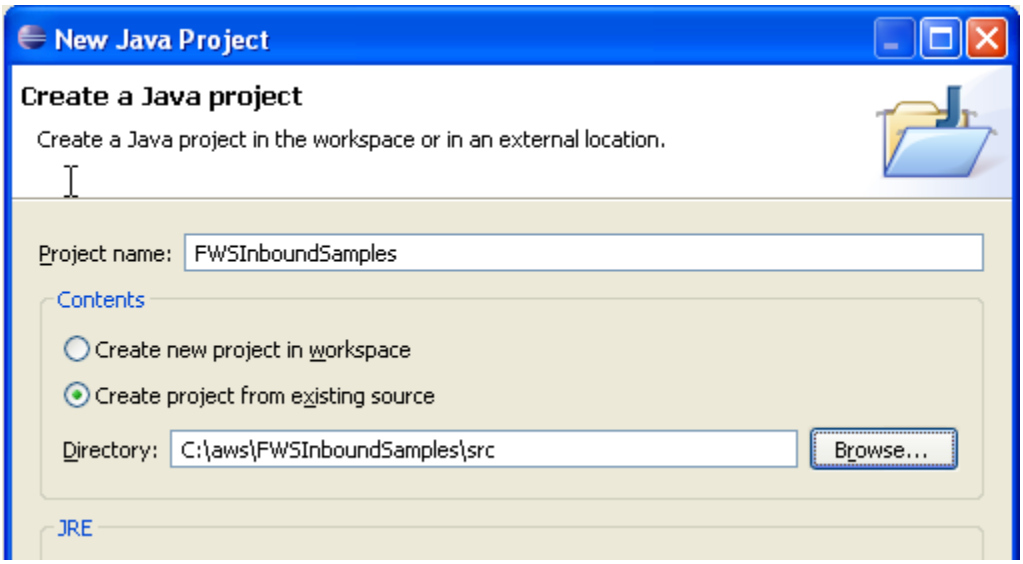
To download both the inbound and the outbound Java project files, go to Amazon FWS Community Code page and unzip the contents to a directory of your choice.

To download both the Java inbound AWS library and the Java outbound AWS library, go to the Amazon FWS Community Code page and unzip the contents to a directory of your choice.

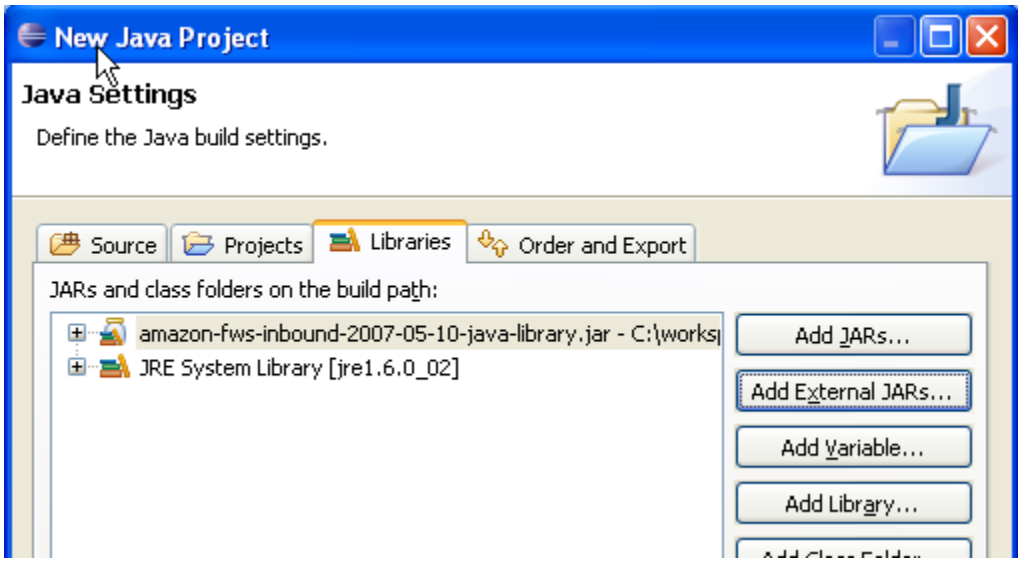
Once you have downloaded the project files and libraries, configure the projects for your environment.

To configure the inbound project in Eclipse

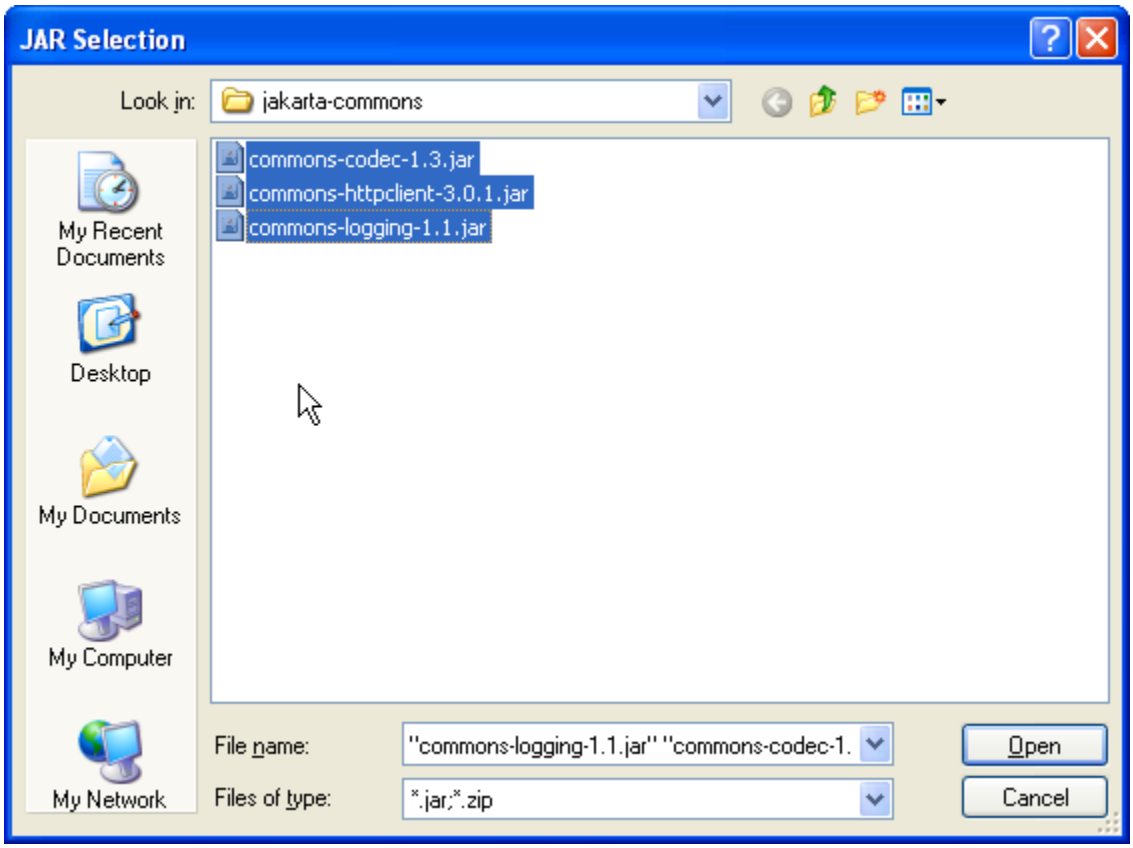
1. In Eclipse, create a new Java project from an existing source, using the inbound Java project folder `FWSInboundSamples\src`.



2. Add the inbound AWS library's .jar (`amazon-fws-inbound-2007-05-10-java-library.jar`) as an external JAR.

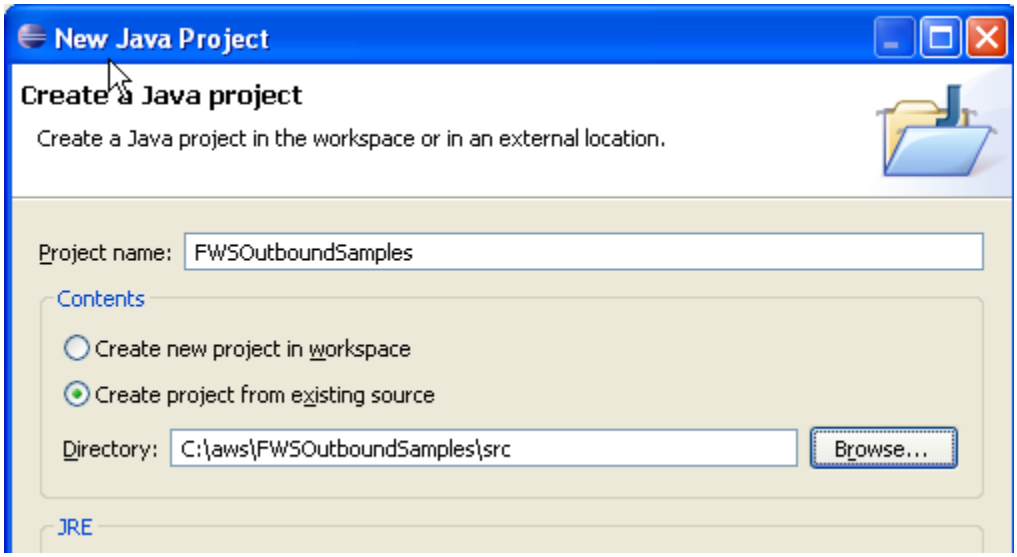


3. Add the three .jar files in the inbound AWS library's `amazon-fws-inbound-2007-05-10-java-library\third-party\jakarta-commons` folder as external JARs.

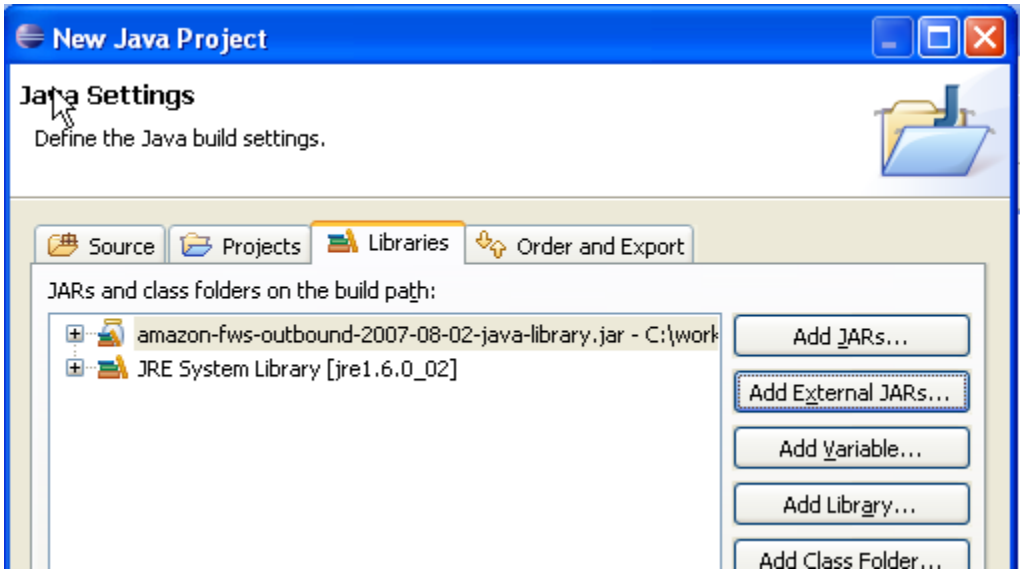


To configure the outbound project in Eclipse

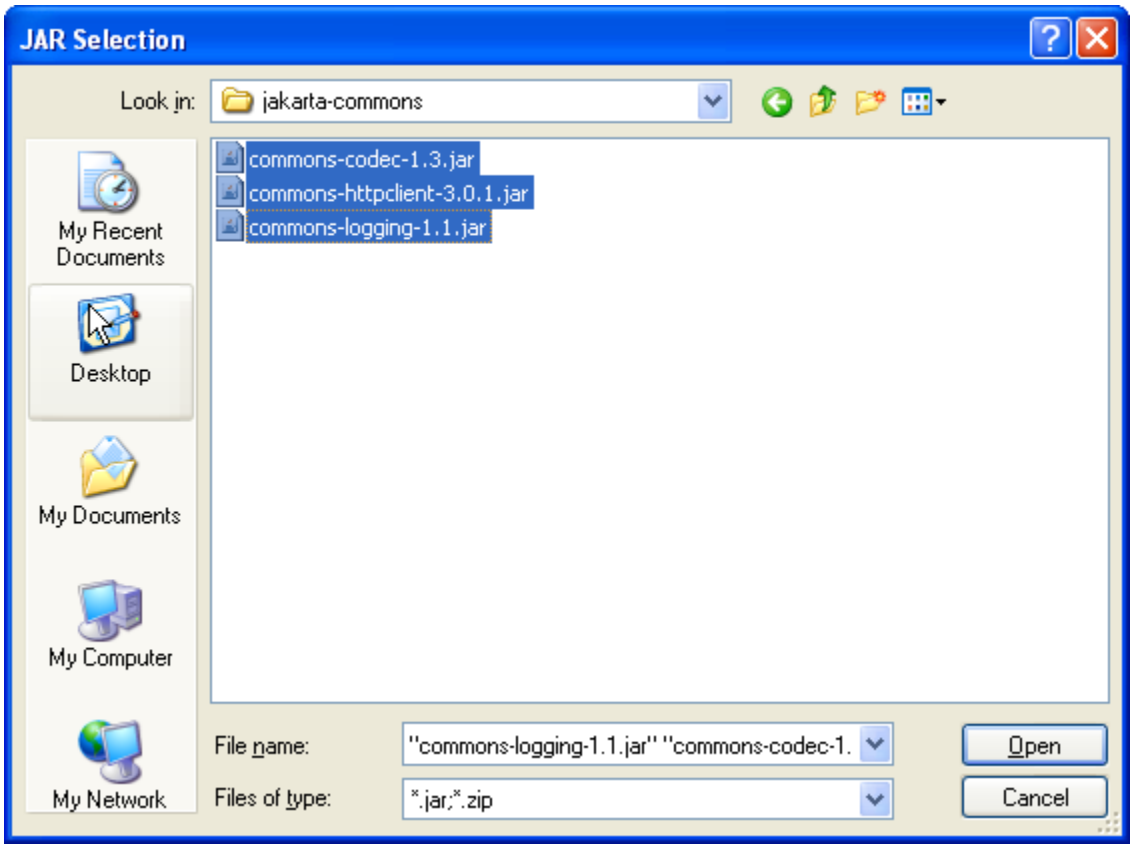
1. In Eclipse, create a new Java project from an existing source, using the outbound Java project folder `FWSOutboundSamples\src`.



2. Add the outbound AWS library's .jar (`amazon-fws-outbound-2007-08-02-java-library.jar`) as an external JAR.



3. Add the three .jar files in the inbound AWS library's `amazon-fws-outbound-2007-08-02-java-library\third-party\jakarta-commons` folder as external JARs.



C#

To download both the inbound and the outbound C# project files, go to Amazon FWS Community Code page and unzip the contents to a directory of your choice.

To download the C# inbound AWS library, go to Amazon FWS Community Code page and unzip the contents to a directory of your choice. To download the C# outbound AWS library, go to Amazon FWS Community Code page and unzip the contents to a directory of your choice.

Once you have downloaded the project files and libraries, configure the projects for your environment.

To configure the inbound projects for C#

1	Go to the folder where you downloaded the project files.
2	Open the `FWSInboundSamples/FWSInboundSamples.sln` file.
3	Add the library project (`Amazon.FWSInbound.csproj` as an existing project to FWSInboundSamples.
4	Make Amazon.FWSInbound a dependency of both FWSInboundGetStatus and FWSInboundGettingStarted.
5	In both FWSInboundGetStatus and FWSInboundGettingStarted, add a reference in the Project tab to Amazon.FWSInbound.

To configure the outbound projects for C#

1	Go to the folder where you downloaded the project files
2	Open the `FWSOutboundSamples/FWSOutboundSamples.sln` file
3	Add the library project (`Amazon.FWSOutbound.csproj` as an existing project to FWSOutboundSamples
4	Make Amazon.FWSOutbound a dependency of both FWSOutboundGetStatus and FWSOutboundGettingStarted
5	In both FWSOutboundGetStatus and FWSOutboundGettingStarted, add a reference in the Project tab to Amazon.FWSOutbound

Scratchpad

The Scratchpad utility is self-contained, so you don't have to download any sample code or libraries.

Now that you have the sample code, you're ready to make requests to Amazon FWS.

Making Requests to Amazon FWS

Topics

Once you're set up to use Amazon FWS, you can start making requests to the inbound, outbound, and inventory APIs. The following sections show sample code for making such requests. We recommend that you follow the samples in the order presented, because subsequent requests often depend on responses received by earlier requests.

The Java and C# samples in this section are from the project files and use the Amazon FWS libraries that AWS provides (to get the libraries, see Getting the Sample Code (p. 15)). To use the examples, you need to edit them to include your AWS access key identifiers, compile them, and then run them. To get your AWS access key identifiers, see Getting Your AWS Access Identifiers (p. 12).

Tip

In the HTML version of this document, 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.

Your selection applies to all of the pages in this guide.

Work flow for getting inventory in and shipping it out

1	List an item to sell. (p. 21)
2	Check the status of the inbound service. (p. 23)
3	Create a shipment and mark the shipment as shipped. (p. 26)

4	Check the status of the outbound service. (p. 33)
5	Create a fulfillment order. (p. 36)

Listing an Item to Sell

When you create a listing, you tell Amazon about the product you intend to sell. When you use Amazon for your fulfillment, you must create a listing regardless of whether you intend to sell the item on Amazon.com or on other sales channels (such as another web site). If you need additional help with creating a listing, go to the Help for Products & Listings.

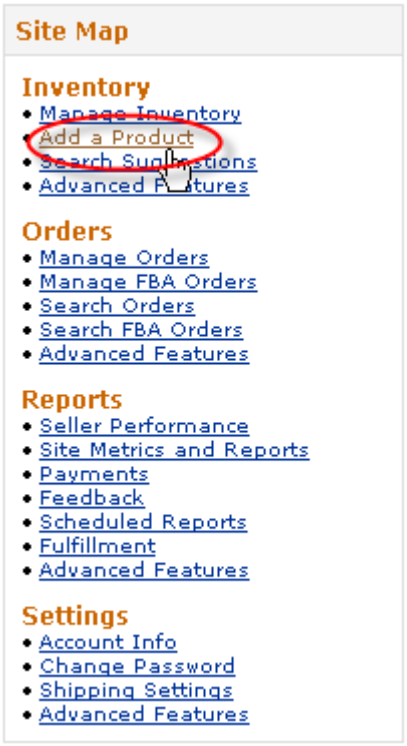
Important

There is no FWS API for this task. However, you can use Amazon Marketplace Web Service for this task. For more information, go to the SubmitFeed operation in the Amazon Marketplace Web Service Developer Guide.

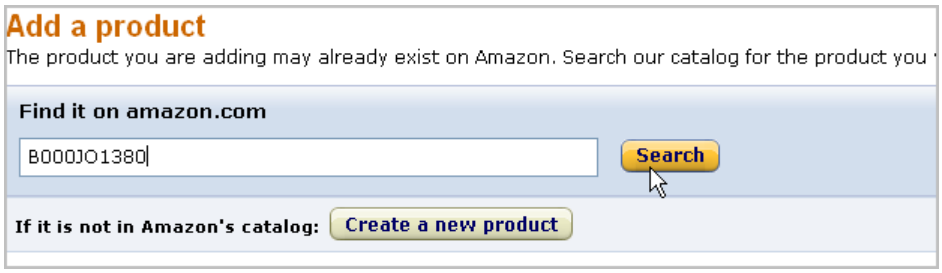
For the examples used in this guide, we'll create a listing for a new Apple 80 GB iPod. We'll keep you from having to sell an actual item by pricing it high enough that no buyer would want it.

To create a listing for a non-media item

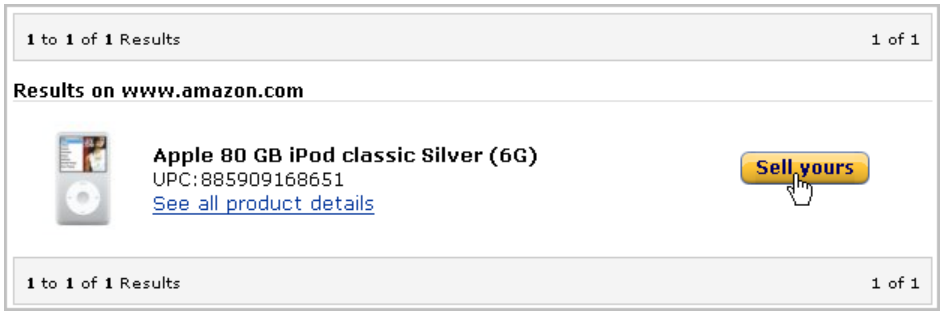
1. Go to Seller Central and log in to your merchant account.
2. Click **Add a Product** from the Site map on the left side of the screen.



3. In the **Add a product** page, enter **`B000JO1380`** in the **Find it on amazon.com** text box. This is the Amazon Standard Item Number (ASIN), a ten-digit Amazon.com product identifier, for the iPod.

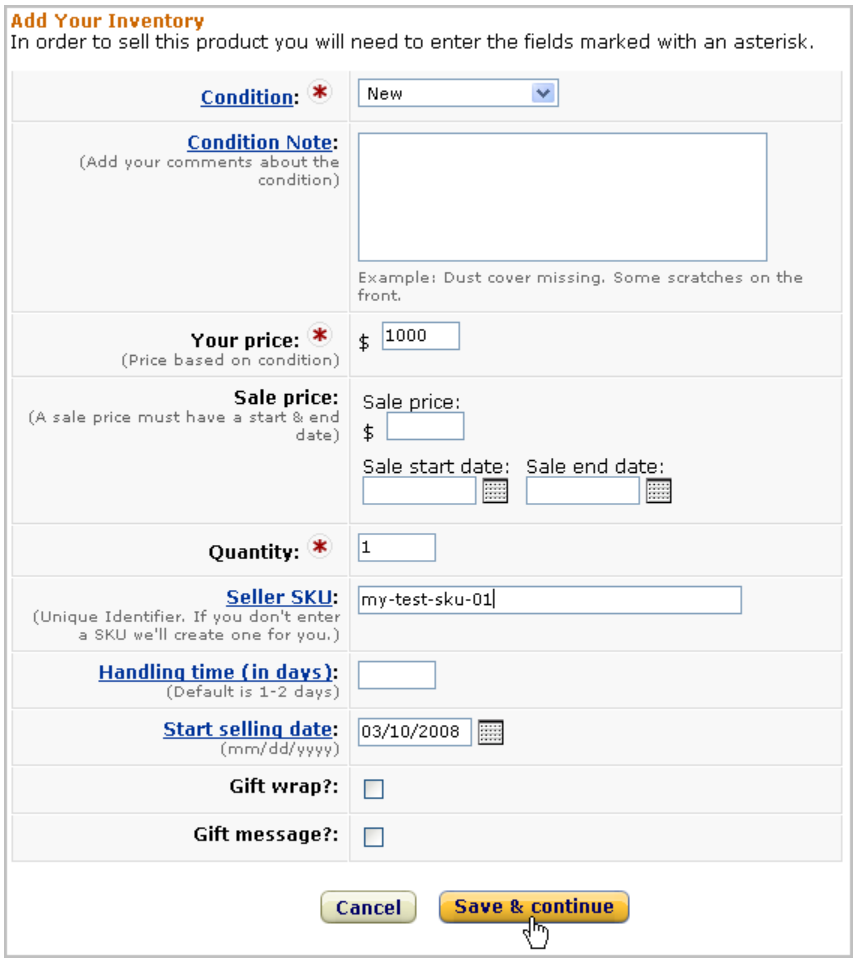


4. In the **Results** section of the page, click the **Sell Yours** button.



5. In the **Add Your Inventory** page, enter all the following information:

   - In the **Condition** field, select **New**.
   - In the **Your price** field, enter **`1000`**.
   - In the **Quantity** field, enter **`1`**.
   - In the **Seller SKU** field, enter **`my-test-sku-01`**
   - Click the **Save & continue** button.



Important

Be sure to mark your listing as AFN (for Amazon fulfilled). The inbound API does not mark the listing on Amazon.com as Amazon fulfilled. The inbound API only involves inventory and inbound shipments. To mark your item to be fulfilled by Amazon, go to Seller Central and click the **Manage Inventory** link. In the Manage Inventory page, select the checkbox next to the SKU you created (my-test-sku-01), then click the **Apply to Selected** option field and select **Convert to 'Fulfilled by Amazon'**. Click the **Go** button to make this change.

For more information about marking your listing as Amazon fulfilled, go to the Help for Convert Listings to 'Fulfilled by Amazon'.

You now have a listing with Amazon.com and can use your SKU to begin testing the operations of the inbound API. The first one we'll use the the `GetServiceStatus` operation.

Checking the Service Status (Inbound)

This sample checks to see if the service for the inbound API is running. This is the easiest to run because you only have to use your AWS access key identifiers: Access Key ID and Secret Access Key.

Scratchpad

The easiest way to request the service status is to use the Scratchpad. To get the Scratchpad, see Getting the Tools You Need (p. 13).

To check the inbound service status using Scratchpad

1. Open the Amazon FWS Inbound API Scratchpad utility (go to the directory you created for it, then go to `amazon-fwsinbound-2007-05-10-javascript-scratchpad/webapp/index.html`.
2. In the first text box, enter your AWS Access Key ID.
3. In the next text box, enter your AWS Secret Access Key.
4. In the **Explore API** option box, select **GetServiceStatus**.
5. Click the **Invoke Request** button to send this request to the service.

The operation returns a response similar to the following example.

```
<?xml version="1.0"?>

<ns1:GetServiceStatusResponse

    xmlns:ns1="http://fba-inbound.amazonaws.com/doc/2007-05-10/">

    <ns1:GetServiceStatusResult>

        <ns1:Status>2008-02-24 06:35:11.500 PST service responding.

            [Version: 2007-05-10]</ns1:Status>

    </ns1:GetServiceStatusResult>

    <ns1:ResponseMetadata>

        <ns1:RequestId>1f290cc3-3850-4b14-9225-9e28ae9a358e</ns1:RequestId>

    </ns1:ResponseMetadata>

</ns1:GetServiceStatusResponse>
```

If you receive an error message, be sure that you entered your AWS Access Key ID and AWS Secret Access Key correctly.

Tip

If you copy and paste your AWS Access Key ID, there might be trailing spaces. Delete these spaces and try again.

Java

To check the status of the service for the inbound API using Java

1. Open `InboundGetStatus.java` in the FWSInboundSamples project.
2. Make the changes indicated in the following example

```
package com.amazon.fws.webservices.samples;
```

```
import com.amazonaws.fws.inbound.AmazonFWSInbound;
import com.amazonaws.fws.inbound.AmazonFWSInboundClient;
import com.amazonaws.fws.inbound.AmazonFWSInboundException;
import com.amazonaws.fws.inbound.model.GetServiceStatus;
import com.amazonaws.fws.inbound.model.GetServiceStatusResponse;
import com.amazonaws.fws.inbound.model.GetServiceStatusResult;

public class InboundGetStatus {

        /**
         * @param args
         */
               public static void main(String[] args) {
               try {

               //Replace with your ACCESS_KEY and SECRET_ACCESS_KEY
               String AWS_ACCESS_KEY_ID = "";      (1)
               String AWS_SECRET_ACCESS_KEY = "";  (2)

              AmazonFWSInbound service = new AmazonFWSInboundClient(AWS_AC
CESS_KEY_ID, AWS_SECRET_ACCESS_KEY);
                GetServiceStatus request = new GetServiceStatus();
             GetServiceStatusResponse response = service.getServiceStatus(re
quest);
              GetServiceStatusResult result = response.getGetServiceStatus
Result();
                System.out.println("Status: " + result.getStatus());
          }
           catch (AmazonFWSInboundException e) {
               System.out.println("Error calling GetServiceStatus: " + e);
           }
     }
}
```

(1) Enter your Access Key ID in the quotes.

(2) Enter your Secret Access Key in the quotes.

3. Compile and run and the example. The service returns a response similar to the following example.

```
2008-03-04 16:38:52.289 PST service responding. [Version: 2007-05-10]
```

Tip

If you receive an error message, verify that you entered your Access Key ID and Secret Access Key correctly.

C#

To check the status of the outbound service using C#

1. Open the `InboundGetStatus.cs` in the FWSInboundGetStatus folder.
2. Make the edits noted in the following example.

```
using System;
using System.Collections.Generic;
using System.Text;
using Amazon.FWSInbound;
using Amazon.FWSInbound.Model;

namespace FWSGetServiceStatus
{
     class GetStatus
     {
          static void Main(string[] args)
          {
               string AWS_ACCESS_KEY_ID = "";
               string AWS_SECRET_ACCESS_KEY = "";

              AmazonFWSInbound service = new AmazonFWSInboundClient(AWS_AC
CESS_KEY_ID, AWS_SECRET_ACCESS_KEY);
                GetServiceStatus request = new GetServiceStatus();
             GetServiceStatusResponse response = service.GetServiceStatus(re
quest);
              GetServiceStatusResult result = response.GetServiceStatusRes
ult;
                Console.WriteLine("Inbound Status : " + result.Status);
          }
     }
}
```

1. Enter your Access Key ID in the quotes.
2. Enter your Secret Access Key in the quotes.

3. Build and run and the example. The service returns a response similar to the following example.

```
Status : 2008-03-04 16:38:52.289 PST service responding. [Version: 2007-05-
10]
```

Tip

If you receive an error message, verify that you entered your Access Key ID and Secret Access Key correctly.

Creating a Shipment and Marking it as Shipped

To create an inbound shipment, you'll use two operations from the inbound API: `GetShipmentPreview` and `PutInboundShipment`. The request parameters for `GetShipmentPreview` are `ShipFromAddress` (an aggregate of your address information) and `MerchantSKUQuantityItem` (your SKU and quantity of items). Most of the information is already entered, except where noted in the code comments.

The elements returned in the `GetShipmentPreview` response are the request parameters for the `PutInboundShipment` operation. In the Java and C# examples that follow, the code automates this process. That is, you don't have to set any parameters for the second operation.

Scratchpad

To create and mark an inbound shipment in the Scratchpad, you'll use three separate operations in the inbound API. Two operations, `GetInboundShipmentPreview` and `PutInboundShipment`, create the shipment. The `SetInboundShipmentStatus` operation sets the status of the shipment to *Shipped*.

To send a GetInboundShipmentPreview request

1. In the **Explore API** option box of the Amazon FWS Inbound API Scratchpad utility, select **GetInboundShipmentPreview**.
2. In the **ShipFromAddress** section, enter all the following address information:
   - In the **Name** field, enter your name or business name.
   - In the **Address Line 1** field, enter your or your business name's street address.
   - In the **Address Line 2** field, enter any additional address information, if needed.
   - In the **City** field, enter your city of origin.
   - In the **State Or Province Code** field, enter your state or province code.
   - In the **Country Code** field, enter the ISO 3166 country alpha-2 code (for example, *US*, *UK*, *CA*, *JP*).
   - In the **Postal Code** field, enter the postal code required for US shipments.
3. In the **MerchantSKUQuantityItem** section, enter all of the following item information:
   - In the **Merchant SKU** field, enter **`my-test-sku-01`**.
   - In the **Quantity** field, enter **`1`**.
4. Click the **Invoke Request** button to send this request to the service.

Copy the `ShipmentId` and `DestinationFulfillmentCenter` values in the response from the service. You will need these for the next step. You will also need the `ShipmentId` to mark the shipment as Shipped.

To send a PutInboundShipment request

1. In the **Explore API** option box, select **PutInboundShipment**.
2. In the **Shipment Id** field, enter the value from the `ShipmentId` element returned in the `GetInboundShipmentPreview` operation.
3. In the **Shipment Name** field, enter **`Test Shipment 1`**.
4. In the **Destination Fulfillment Center** field, enter the `DestinationFulfillmentCenter` value from the `GetInboundShipmentPreview` operation.
5. In the **ShipFromAddress** section, enter all the following address information:
   - In the **Name** field, enter your name or business name.
   - In the **Address Line 1** field, enter your or your business name's street address.
   - In the **Address Line 2** field, enter any additional address information, if needed.
   - In the **City** field, enter your city of origin.
   - In the **State Or Province Code** field, enter your state or province code.
   - In the **Country Code** field, enter the ISO 3166 country alpha-2 code (for example, *US*, *UK*, *CA*, *JP*).
   - In the **Postal Code** field, enter the postal code required for US shipments.
6. In the **MerchantSKUQuantityItem** section, enter all of the following item information:
   - In the **Merchant SKU** field, enter **`my-test-sku-01`**.
   - In the **Quantity** field, enter **`1`**.

The service returns no response for the `PutInboundService` operation.

To send a SetInboundShipmentStatus request

1. In the **Explore API** option box, select **SetInboundShipmentStatus**.
2. In the **Shipment Id** field, enter the value from the `ShipmentId` element returned in the `GetInboundShipmentPreview` operation.
3. In the **Shipment Status** option field, select **`Shipped`**.

The service returns no response for the `SetInboundShipmentStatus` operation.

You've now completed the inboud API tasks for sending an item into an Amazon fulfillment center. In the next section, we'll show an example of how to use the outbound API to send an item from the fulfillment center to a recipient.

Java

To create an inbound shipment and mark it as shipped in Java

1. Open `InboundGettingStarted.java` in the FWSInboundSamples project.
2. Make the changes indicated in the following example.

```
package com.amazon.fws.webservices.samples;

import com.amazonaws.fws.inbound.AmazonFWSInbound;
import com.amazonaws.fws.inbound.AmazonFWSInboundClient;
import com.amazonaws.fws.inbound.AmazonFWSInboundException;
import com.amazonaws.fws.inbound.model.MerchantSKUQuantityItem;
import com.amazonaws.fws.inbound.model.Address;
import com.amazonaws.fws.inbound.model.GetInboundShipmentPreview;
import com.amazonaws.fws.inbound.model.GetInboundShipmentPreviewResponse;
import com.amazonaws.fws.inbound.model.SetInboundShipmentStatus;
import com.amazonaws.fws.inbound.model.ShipmentPreview;
import com.amazonaws.fws.inbound.model.PutInboundShipment;
import com.amazonaws.fws.inbound.model.ShipmentStatus;

import java.util.List;

public class InboundGettingStarted {

     /**
      * @param args
      */
     public static void main(String[] args) {
          try {

               //Replace with your ACCESS_KEY and SECRET_ACCESS_KEY
               String AWS_ACCESS_KEY_ID = "";      (1)
               String AWS_SECRET_ACCESS_KEY = "";  (2)

             //Replace with an AFN SKU that has had an Offer Listing created
 by Merchant account for this
               //AWS account on http://www.sellercentral.com
               String merchantSKU = "";  (3)

             MerchantSKUQuantityItem mSkuQuantityItem = new MerchantSKUQuant
ityItem();
```

```
                mSkuQuantityItem.setMerchantSKU(merchantSKU);
                mSkuQuantityItem.setQuantity(1);

                Address shipFromAddress = new Address();
                shipFromAddress.setAddressLine1("605 5th Ave SE");
                shipFromAddress.setCity("Seattle");
                shipFromAddress.setCountryCode("US");
                shipFromAddress.setName("My Ship From Address");
                shipFromAddress.setPostalCode("98101");
                shipFromAddress.setStateOrProvinceCode("WA");


              AmazonFWSInbound service = new AmazonFWSInboundClient(AWS_AC
CESS_KEY_ID, AWS_SECRET_ACCESS_KEY);
               GetInboundShipmentPreview getInboundShipmentPreviewRequest
= new GetInboundShipmentPreview();
               getInboundShipmentPreviewRequest.getMerchantSKUQuantity
Item().add(mSkuQuantityItem);
              getInboundShipmentPreviewRequest.setShipFromAddress(shipFro
mAddress);

               // Determine which Amazon Fulfillment Center my item should
 go to
              GetInboundShipmentPreviewResponse resp = service.getInbound
ShipmentPreview(getInboundShipmentPreviewRequest);

               // Amazon Fulfillment may seperate items in a ShipmentPre
viewRequest into one or more shipmentPrevieiws for a variety of reasons:
             // balancing supply across the fulfillment network, to ensure
 products with special attributes go to the appropriate Fulfillment Center
 etc
              List<ShipmentPreview> shipmentPreviews = resp.getGetInbound
ShipmentPreviewResult().getShipmentPreview();

               // NOTE this is a bit simplistic, as there is a physical
reality here, if you have a shipment with multiple MerchantSKUQuantityItems

               // those items can be split across shipments, the client is
 responsible for ensuring that the right sku's go into the right physical
shipments
              // failure to do so will result in a delay in receive at the
 Amazon Fulfillment Center

               for (ShipmentPreview shipmentGroup : shipmentPreviews) {
                    String shipmentID = null;
                    String destinationFulfillmentCenter = null;
                    Address fulfillmentCenterAddress = null;

                    // Create a shipment for each shipment group --> SEE
NOTE ABOVE
                    PutInboundShipment putInboundShipmentRequest = new
PutInboundShipment();

                  //Get the FC name and Address to print on shipping label

                   destinationFulfillmentCenter = shipmentGroup.getDestin
ationFulfillmentCenter();
                    fulfillmentCenterAddress = shipmentGroup.getShipToAd
```

```
dress();

                    putInboundShipmentRequest.setDestinationFulfillmentCen
ter(destinationFulfillmentCenter);
                     putInboundShipmentRequest.getMerchantSKUQuantity
Item().add(mSkuQuantityItem);
                   putInboundShipmentRequest.setShipFromAddress(shipFromAd
dress);

                     //Retrieve shipment id from the preview
                     shipmentID = shipmentGroup.getShipmentId();

                     putInboundShipmentRequest.setShipmentId(shipmentID);
                     putInboundShipmentRequest.setShipmentName("My First
Shipment");

                     // Create the shipment
                     service.putInboundShipment(putInboundShipmentRequest);
                    System.out.println("Created shipment : " + shipmentID);


                     //After physically packing boxes and labeling shipment
 and dropping off at carrier
                     SetInboundShipmentStatus request = new SetInboundShip
mentStatus();
                     request.setShipmentId(shipmentID);
                     request.setShipmentStatus(ShipmentStatus.SHIPPED);
                     service.setInboundShipmentStatus(request);
                     System.out.println("Marked shipment : " + shipmentID +
 " as " + ShipmentStatus.SHIPPED);
                }

           }
           catch (AmazonFWSInboundException e) {
                System.out.println("Error calling GetServiceStatus: " + e);

           }
      }
}
```

1. Enter your Access Key ID in the quotes.
2. Enter your Secret Access Key in the quotes.
3. This is the SKU created when you listed your iPod.
4. The example is set to **1** because we created one item.
5. For testing, you can keep the example address as it is. This is the address the item is shipped from.

3. Compile and run and the example. The service returns a response indicating that your shipment is created and you shipment status is set to *Shipped*.

```
Created shipment : SS4D48DEZZDIQHL
Marked shipment : SS4D48DEZZDIQHL as Shipped
```

C#

To create an inbound shipment and mark it as shipped in C#

1. Open the `InboundGettingStarted.cs` file in the FWSInboundGettingStarted folder.
2. Make the edits indicated in the following example.

```
using System;
using System.Collections.Generic;
using System.Text;
using Amazon.FWSnbound;
using Amazon.FWSInbound.Model;

namespace FWSInboundGettingStarted
{
     class GettingStarted
     {
          static void Main(string[] args)
          {
               //Replace with your ACCESS_KEY and SECRET_ACCESS_KEY

               string AWS_ACCESS_KEY_ID = "";      1

               string AWS_SECRET_ACCESS_KEY = "";  2

                AmazonFWSInbound service = new AmazonFWSInboundClient(AWS_AC
CESS_KEY_ID, AWS_SECRET_ACCESS_KEY);

              //Replace with an AFN SKU that has had an Offer Listing created
 by Merchant account for this
               //AWS account on http://www.sellercentral.com

               string merchantSKU = "";            3

             MerchantSKUQuantityItem mSkuQuantityItem = new MerchantSKUQuant
ityItem();
               mSkuQuantityItem.WithMerchantSKU(merchantSKU);

               mSkuQuantityItem.WithQuantity(1);   4

               Address shipFromAddress = new Address();

               shipFromAddress.AddressLine1 = "605 5th Ave SE";      5
               shipFromAddress.City = "Seattle";
               shipFromAddress.CountryCode = "US";
               shipFromAddress.Name = "My Work Address";
               shipFromAddress.PostalCode = "98104";
               shipFromAddress.StateOrProvinceCode = "WA";

               GetInboundShipmentPreview getInboundShipmentPreviewRequest
= new GetInboundShipmentPreview();
               getInboundShipmentPreviewRequest.MerchantSKUQuantity
Item.Add(mSkuQuantityItem);
               getInboundShipmentPreviewRequest.ShipFromAddress = shipFro
mAddress;

               // Determine which Amazon Fulfillment Center my item should
 go to
               GetInboundShipmentPreviewResponse response = service.GetIn
boundShipmentPreview(getInboundShipmentPreviewRequest);
```

```
              // Amazon Fulfillment may seperate items in a ShipmentPre
viewRequest into one or more shipmentPrevieiws for a variety of reasons:
             // balancing supply across the fulfillment network, to ensure
 products with special attributes go to the appropriate Fulfillment Center
 etc
              List<ShipmentPreview> shipmentPreviews = (List<ShipmentPre
view>)response.GetInboundShipmentPreviewResult.ShipmentPreview;

              // NOTE this is a bit simplistic, as there is a physical
reality here, if you have a shipment with multiple MerchantSKUQuantityItems

              // those items can be split across shipments, the client is
 responsible for ensuring that the right sku's go into the right physical
shipments
             // failure to do so will result in a delay in receive at the
 Amazon Fulfillment Center
              string shipmentID = null;

              foreach (ShipmentPreview shipmentGroup in shipmentPreviews)
              {
                   string destinationFulfillmentCenterName = null;
                   Address fulfillmentCenterAddress = null;

                   // Create a shipment for each shipment group --> SEE
NOTE ABOVE
                   PutInboundShipment putInboundShipmentRequest = new
PutInboundShipment();

                  //Get the FC name and Address to print on shipping label

                 destinationFulfillmentCenterName = shipmentGroup.Destin
ationFulfillmentCenter;
                  fulfillmentCenterAddress = shipmentGroup.ShipToAddress;


                   //Again this is simplistic for demonstration purposes,
 we are only using a single item
                   //Re-add the set of items that preview told us are ok
to go together in a shipment
                   //You aren't bound by what preview told you were the
items in a particular group
                   //but you could get Exceptions if you add additional
items....
                   putInboundShipmentRequest.DestinationFulfillmentCenter
 = destinationFulfillmentCenterName;
                   putInboundShipmentRequest.MerchantSKUQuantity
Item.Add(mSkuQuantityItem);
                  putInboundShipmentRequest.ShipFromAddress = shipFromAd
dress;

                   //Retrieve shipment id from the preview
                   shipmentID = shipmentGroup.ShipmentId;

                   putInboundShipmentRequest.ShipmentId = shipmentID;
                   putInboundShipmentRequest.ShipmentName = "My First
Shipment";

                   //Create the shipment
```

```
                    service.PutInboundShipment(putInboundShipmentRequest);
                    Console.WriteLine("Created shipment : " + shipmentID);

                    //After physically packing boxes and labeling shipment
 and dropping off at carrier
                    //Mark the Shipment as shipped
                    SetInboundShipmentStatus request = new SetInboundShip
mentStatus();
                    request.ShipmentId = shipmentID;
                    request.ShipmentStatus = ShipmentStatus.Shipped;
                    service.SetInboundShipmentStatus(request);
                    Console.WriteLine("Marked shipment : " + shipmentID +
" as " + ShipmentStatus.Shipped);

                }
            }
        }
}
```

1. Enter your Access Key ID in the quotes.
2. Enter your Secret Access Key in the quotes.
3. This is the SKU created when you listed your iPod.
4. The example is set to **1** because we created one item.
5. For testing, you can keep the example address as it is. This is the address the item is shipped from.

3. Compile and run and the example.The service returns a response indicating that your shipment is created and you shipment status is set to *Shipped*.

```
Created shipment : SS4D48DEZZDIQHL
Marked shipment : SS4D48DEZZDIQHL as Shipped
```

Checking the Service Status (Outbound)

We now move from the inbound API to the outbound API. This first operation we'll use is one that checks to see if the service for the outbound API is running. Like the inbound service check, this operation takes no parameters other than your AWS access key identifiers: Access Key ID and Secret Access Key.

Scratchpad

To check the outbound service status using Scratchpad

1. Open the Amazon FWS Outbound API Scratchpad utility (`amazon-fwsoutbound-2007-08-02-javascript-scratchpad/webapp/index.html`).
2. In the first text box, enter your AWS Access Key ID.
3. In the next text box, enter your AWS Secret Access Key.
4. In the **Explore API** option box, select **GetServiceStatus**.
5. Click the **Invoke Request** button to send this request to the service.

Build and run and the example. The service returns a response similar to the following example.

```
Status : 2008-03-17T04:23:50Z service available [Version: 2007-08-02]
```

If you receive an error message, verify that you entered your AWS Access Key ID and AWS Secret Access Key correctly.

Tip

If you copy and paste your AWS Access Key ID, there might be trailing spaces. Delete these spaces and try again.

Java

To check the status of the outbound service using Java

1. Open `OutboundGetStatus.java` in the FWSOutboundSamples project.
2. Make the edits indicated in the following example.

```
package com.amazon.fws.webservices.samples;

import com.amazonaws.fws.outbound.AmazonFWSOutbound;
import com.amazonaws.fws.outbound.AmazonFWSOutboundClient;
import com.amazonaws.fws.outbound.AmazonFWSOutboundException;
import com.amazonaws.fws.outbound.model.GetServiceStatus;
import com.amazonaws.fws.outbound.model.GetServiceStatusResponse;
import com.amazonaws.fws.outbound.model.GetServiceStatusResult;

public class OutboundGetStatus {

     /**
      * @param args
      */
     public static void main(String[] args) {
          try {

               //Replace with your ACCESS_KEY and SECRET_ACCESS_KEY
               String AWS_ACCESS_KEY_ID = "";      (1)
               String AWS_SECRET_ACCESS_KEY = "";  (2)

             AmazonFWSOutbound service = new AmazonFWSOutboundClient(AWS_AC
CESS_KEY_ID, AWS_SECRET_ACCESS_KEY);
               GetServiceStatus request = new GetServiceStatus();
             GetServiceStatusResponse response = service.getServiceStatus(re
quest);
              GetServiceStatusResult result = response.getGetServiceStatus
Result();
               System.out.println("Status: " + result.getStatus());
          }
          catch (AmazonFWSOutboundException e) {
               System.out.println("Error calling GetServiceStatus: " + e);
          }

     }

}
```

(1) Enter your Access Key ID in the quotes.

(2) Enter your Secret Access Key in the quotes.

3. Compile and run and the example. The service returns a response similar to the following example.

```
Status : 2008-03-04 16:38:52.289 PST service responding. [Version: 2007-08-
02]
```

Tip

If you receive an error message, verify that you entered your Access Key ID and Secret Access Key correctly.

C#

To check the status of the outbound service using C#

1. Open the `OutboundGetStatus.cs` in the FWSOutboundGetStatus folder.
2. Make the edits indicated in the following example.

```
using System;
using System.Collections.Generic;
using System.Text;
using Amazon.FWSOutbound;
using Amazon.FWSOutbound.Model;


namespace FWSOutboundGetStatus
{
     class OutboundGetStatus
     {
          static void Main(string[] args)
          {
               //Replace with your ACCESS_KEY and SECRET_ACCESS_KEY
               string AWS_ACCESS_KEY_ID = "";      (1)
               string AWS_SECRET_ACCESS_KEY = "";  (2)

             AmazonFWSOutbound service = new AmazonFWSOutboundClient(AWS_AC
CESS_KEY_ID, AWS_SECRET_ACCESS_KEY);
               GetServiceStatus request = new GetServiceStatus();
             GetServiceStatusResponse response = service.GetServiceStatus(re
quest);
              GetServiceStatusResult result = response.GetServiceStatusRes
ult;
               Console.WriteLine("Status : " + result.Status);
          }
     }
     }
```

(1) Enter your Access Key ID in the quotes.

(2) Enter your Secret Access Key in the quotes.

3. Build and run and the example. The service returns a response similar to the following example.

```
Status : 2008-03-04 16:38:52.289 PST service responding. [Version: 2007-08-
02]
```

Tip

If you receive an error message, verify that you entered your Access Key ID and Secret Access Key correctly.

Creating a Fulfillment Order

The `CreateFulfillmentOrder` operation generates a request for the Amazon fulfillment center to ship your inventory to a recipient. This section shows this request in Java, C#, and Scratchpad.

Any order generated during this example process will be accepted by the system. However, since you have no physical inventory in a fulfillment center, such orders will display as unfulfillable.

Tip

You only need to use this operation or any operations in the outbound API if you are selling from a channel other than Amazon.com. If you're only selling on Amazon.com, the packaging and shipping process is done automatically.

Scratchpad

To create a fulfillment order in Scratchpad

1. In the **Explore API** option box, select **CreateFulfillmentOrder**.
2. In the **Merchant Fulfillment Order Id** field, enter **`TEST-00000001`**.
3. In the **Displayable Order Id** field, enter **`TEST-00000001`**.
4. In the **Displayable Order Comment** field, enter a message to display on the packing slip.
5. In the **Shipping Speed Category** enter **`Standard`**.
6. In the **DestinationAddress** section, enter all the following address information for the recipient. You can use the test address information in the Java and C# examples, or make it all up yourself.

   - In the **Name** field, enter the recipient's name.
   - In the **Line 1** field, enter the recipient's street address
   - In the **Line 2** field, enter any additional address information, if needed
   - In the **Line 3** field, enter any additional address information, if needed
   - In the **City** field, enter the recipient's city of origin
   - In the **State Or Province Code** field, enter the recipient's state or province code
   - In the **Country Code** field, enter the ISO 3166 country alpha-2 code (for example, *US*, *UK*, *CA*, *JP*).
   - In the **Postal Code** field, enter the postal code required for US shipments
   - In the **Phone Number** field, enter a ten-digit phone number.

7. In the **Item** section, enter your item information:

   - In the **Merchant SKU** field, enter **`my-test-sku-01`**.
   - In the **Merchant Fulfillment Order Item Id** field, enter **`TEST-00000001-1`**.
   - In the **Quantity** field, enter **`1`**.

8. Click the **Invoke Request** button to send this request to the service.

   If sucessful, the operation only returns a request ID. To check the status of your fulfillment order, go to Seller Central.

Java

To create a fulfillment order in Java

1. Open `OutboundGettingStarted.java` in the FWSOutboundSamples project.
2. Make the changes indicated in the following example.

```
package com.amazon.fws.webservices.samples;

import java.util.ArrayList;
import java.util.List;

import com.amazonaws.fws.outbound.AmazonFWSOutbound;
import com.amazonaws.fws.outbound.AmazonFWSOutboundClient;
import com.amazonaws.fws.outbound.AmazonFWSOutboundException;
import com.amazonaws.fws.outbound.model.Address;
import com.amazonaws.fws.outbound.model.CreateFulfillmentOrder;
import com.amazonaws.fws.outbound.model.CreateFulfillmentOrderItem;
import com.amazonaws.fws.outbound.model.CreateFulfillmentOrderResponse;
import com.amazonaws.fws.outbound.model.GetServiceStatus;
import com.amazonaws.fws.outbound.model.GetServiceStatusResponse;
import com.amazonaws.fws.outbound.model.GetServiceStatusResult;

public class OutboundGettingStarted {

     /**
      * @param args
      */
     public static void main(String[] args) {
          try {

               //Replace with your ACCESS_KEY and SECRET_ACCESS_KEY

               String AWS_ACCESS_KEY_ID = "";      (1)

               String AWS_SECRET_ACCESS_KEY = "";  (2)

             AmazonFWSOutbound service = new AmazonFWSOutboundClient(AWS_AC
CESS_KEY_ID, AWS_SECRET_ACCESS_KEY);
             CreateFulfillmentOrder request = new CreateFulfillmentOrder();


               request.setMerchantFulfillmentOrderId("TEST-0000001");
               request.setDisplayableOrderId("TEST-00000001");
              request.setDisplayableOrderDateTime("2007-08-02T00:00:00Z");

               request.setDisplayableOrderComment("Thank you for your or
der!");
               request.setShippingSpeedCategory("Standard");

                Address address = new Address();  (3)
               address.setName("Joe Smith");
```

```
                address.setLine1("605 5th Ave. S");
                address.setLine2("C/O Amazon.com");
                address.setCity("Seattle");
                address.setStateOrProvinceCode("WA");
                address.setPostalCode("98104");
                address.setCountryCode("US");
                address.setPhoneNumber("206-266-1000");
                request.setDestinationAddress(address);

                List<CreateFulfillmentOrderItem> items = new ArrayList<Cre
ateFulfillmentOrderItem>(1);

                CreateFulfillmentOrderItem item1 = new CreateFulfillmentOr
derItem();
                //replace with a valid sku that has inventory
                //failure to do so will create an order successfully but its
 status will be unfulfillable

                item1.setMerchantSKU("");
                String myOrderID = "TEST-00000001-1";
                item1.setMerchantFulfillmentOrderItemId(myOrderID);
                item1.setQuantity(2);
                items.add(item1);

                request.setItem(items);

               CreateFulfillmentOrderResponse response = service.createFul
fillmentOrder(request);
                System.out.println("Order : " + myOrderID + " sent. You can
 check status in http://www.sellercentral.com");

          }
          catch (AmazonFWSOutboundException e) {
                System.out.println("Error calling GetServiceStatus: " + e);
          }

     }

}
```

1. Enter your Access Key ID in the quotes.
2. Enter your Secret Access Key in the quotes.
3. Because this fulfillment order isn't for an actual inventory item in the fulfillment center, you can keep the fake address and order information and address included in this example.
4. Enter the merchant SKU for this item, *my-test-sku-01*.

3. Compile the project and run the executable. The service responds with a message similar to the following example.

```
Order : TEST-00000001-1 sent. You can check status in http://www.sellercent
ral.com
```

C#

To create a fulfillment order using C#

1. Open the `OutboundGettingStarted.cs` in the FWSOutboundGettingStarted folder.
2. Make the edits indicated in the following example.

```
using System;
using System.Collections.Generic;
using System.Text;
using Amazon.FWSOutbound;
using Amazon.FWSOutbound.Model;

namespace FWSOutboundGettingStarted
{
     class InboundGettingStarted
     {
          static void Main(string[] args)
          {
               //Replace with your ACCESS_KEY and SECRET_ACCESS_KEY

               string AWS_ACCESS_KEY_ID = "";      (1)

               string AWS_SECRET_ACCESS_KEY = "";  (2)

             AmazonFWSOutbound service = new AmazonFWSOutboundClient(AWS_AC
CESS_KEY_ID, AWS_SECRET_ACCESS_KEY);

             CreateFulfillmentOrder request = new CreateFulfillmentOrder();


               request.MerchantFulfillmentOrderId = "TEST-00000001";
               request.DisplayableOrderId = "TEST-00000001";
               request.DisplayableOrderDateTime = "2008-03-14T00:00:00Z";
            request.DisplayableOrderComment = "Thank you for your order!";

               request.ShippingSpeedCategory = "Standard";


               Address address = new Address();  (3)
               address.Name = "Joe Smith";
               address.Line1 = "605 5th Ave. S";
               address.Line2 = "C/O Amazon.com";
               address.City = "Seattle";
               address.StateOrProvinceCode = "WA";
               address.PostalCode = "98104";
               address.CountryCode = "US";
               address.PhoneNumber = "206-266-1000";
               request.DestinationAddress = address;

              List<CreateFulfillmentOrderItem> items = new List<CreateFul
fillmentOrderItem>(1);

             CreateFulfillmentOrderItem item1 = new CreateFulfillmentOrder
Item();
              //replace with a valid sku that has inventory
              //failure to do so will create an order successfully but its
 status will be unfulfillable
```

```
                item1.MerchantSKU = "";
                string myOrderID = "TEST-00000001-1";
                item1.MerchantFulfillmentOrderItemId = myOrderID;
                item1.Quantity = 1;
                items.Add(item1);

                request.Item = items;

                CreateFulfillmentOrderResponse response = service.CreateFul
fillmentOrder(request);
                Console.WriteLine("Order : " + myOrderID + " sent. You can
check status in http://www.sellercentral.com");
            }
        }
        }
```

1. Enter your Access Key ID in the quotes.
2. Enter your Secret Access Key in the quotes.
3. Because this fulfillment order isn't for an actual inventory item in the fulfillment center, you can keep the fake address and order information and address included in this example.
4. Enter the merchant SKU for this item, *my-test-sku-01*.

3. Build the project and run the executable. The service responds with a message similar to the following example.

```
Order : TEST-00000001-1 sent. You can check status in http://www.sellercent
ral.com
```

You're Finished!

Congratulations! You successfully listed an item, checked the inbound service status, created a shipment item, checked the outbound service status, and created a fulfillment order. For information on ideas that were not covered in this guide and how to continue, see Where to Go from Here (p. 41).

Please Give Us Your Feedback

Your input is important to us to help make our documentation helpful and easy to use. Please take a minute to give us your feedback on how well we were able to help you get started with Amazon FWS. Just click this Feedback Link link. Thank you.

Where to Go from Here

Now that you have completed the basic example presented in this guide, you are ready to start designing your own application. The Amazon FWS APIs (inbound, outbound, and inventory) include operations that let you accomplish the following tasks:

- Add items to an existing shipment (for more information, go to the PutInboundShipmentItems section in the Amazon FWS Developer Guide.
- Change shipment data in an existing shipment (for more information, go to the PutInboundShipmentData section in the Amazon FWS Developer Guide.
- List all of your listed fulfillment items (for more information, go to the ListAllFulfillmentItems section in the Amazon FWS Developer Guide.
- Cancel an existing fulfillment order (for more information, go to the CancelFulfillmentOrder section in the Amazon FWS Developer Guide.

For further assistance with building your application, we suggest you use the other available resources for Amazon FWS. For more information, see Amazon FWS Resources (p. 3).

Read the Forum

We recommend you look at the Amazon Fulfillment Web Service forum to get an idea of what other users are doing and questions they've had. This will help you further understand what you can and can't do with Amazon FWS.

Look at Other Available Sample Code

You're already aware of the sample code that goes with this guide (for more information, see Getting the Sample Code (p. 15)). You can look at any other sample code that's available from the Sample Code and Libraries page.

How to Cancel Your Registration

You can cancel your registration at any time.

To cancel your registration

1. Go to aws.amazon.com
2. Point to **Your Account** and click **Account Activity**.

   The **Account Activity** page displays.
3. Click the **View/Edit Service** link under the service name.

   The **View/Edit Service** page displays.
4. Click the link, **cancel this service**.
   This link is typically in the last sentence of the opening paragraph.

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. 43).

Convention	Description/Example
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`.
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 symbol convention below.

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`