Amazon Fulfillment Web Service

Developer Guide

Version 1.1

Amazon Fulfillment Web Service: Developer Guide

Table of Contents

Welcome

Topics

This is the *Amazon Fulfillment Web Service* Developer Guide. 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 are building web-enabled applications that will use Amazon FWS to integrate listing, payment, or reconciliation tools with Amazon either to send inventory to Amazon fulfillment centers or to request the centers to ship inventory to recipients.

To use the Amazon FWS, you must have an Amazon AWS account and be signed up to use Amazon FWS. See the *Amazon Fulfillment Web Service Getting Started Guide* for instructions.

Required Knowledge and Skills

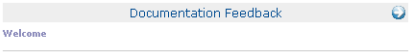
Use of this guide assumes you are familiar with the following:

- 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*)
- A programming language for consuming a web service and any related tools

You should also have read the *Amazon Fulfillment Web Service Getting Started Guide*, which includes a tutorial showing you how to set up the service and perform a basic request to get the service status.

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 This Guide Is Organized

This guide is organized into several major sections described in the following table.

Information	Relevant Sections
General information about Amazon FWS	Introduction to Amazon FWS (p. 6)
Information about requests and responses, including code samples for common use cases	Programming Guide (p. 9)
Typographic and symbol conventions	Document Conventions (p. 52)
Appendix: Label, Pallet, and Font Specifications	Appendix (p. 55)
Glossary	Glossary (p. 62)
Index	Index (p. 66)

Amazon FWS Resources

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

Resource	Description
Amazon FWS Getting Started Guide	The Getting Started Guide provides a quick tutorial of the service based on a simple use case. Examples and instructions for Java, C#, and PHP are included.
Amazon FWS API Guide	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

Resource	Description
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.
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 Developer Guide.

Change	Description	Release Date
New Document	Made the API reference section its own document. For API information, go to the Amazon Fulfillment Web Service API Reference document.	04 June 2010
Update	Updated the item label's barcode specifications for the narrow barcode element. For more information, see the narrow barcode element (p. 57) description.	3 August 2009
Update	New form for submitting item label, shipping label, and packing slip designs for FBA approval. For more information, see Label and Slip Approval (p. 60).	13 August 2008
Update	Updated information about item label, shipping label, packing slip, pallet, and font specifications. For more information, see Appendix (p. 55).	1 July 2008

Change	Description	Release Date
Update	Updated information about the response elements for the GetFulfillmentOrder operation. This information is not a change to the API, but rather an addition to the documentation for the API. The FulfillmentOrderItem aggregate now contains documentation for the elements `CancelledQuantity`, `UnfulfillableQuantity`, `EstimatedShipDateTime`, and `EstimatedArrivalDateTime`. The FulfillmentShipment aggregate now contains documentation for the `EstimatedArrivalDateTime` element. For more information, go the `GetFulfillmentOrder` operation in the Amazon Fulfillment Web Service API Reference document.	1 July 2008
First Release	This is the first release of the Amazon Fulfillment Web Service Developer Guide.	19 March 2008

Introduction to Amazon FWS

Topics

This introduction to Amazon Fulfillment Web Service is intended to give you a detailed summary of the web service. After reading this section, you should have a good idea of what Amazon FWS offers and how it can fit in with your business. For further information about the fulfillment process, go to the Fulfillment by Amazon web site.

Overview

The Fulfillment by Amazon (FBA) program allows you to use Amazon's world-class fulfillment operations. Amazon FWS is a set of APIs that takes advantage of FBA in a programmatic function. These APIs allow you to use the Amazon Fulfillment Network.

Using Amazon FWS you can integrate FBA functionality into an existing application by including specific functions to create inbound shipments to Amazon from your software. You can also create tools or software that connect shopping cart or ordering software directly to FBA. By linking to FBA, orders that you generate can also generate fulfillment requests directly to the Amazon fulfillment center. This means you can sell products on your own site and automatically generate fulfillment orders by sending addresses and shipping information to the Amazon fulfillment center.

Fulfillment by Amazon supports multi-channel selling by allowing you to request fulfillment shipments directly using your own shopping cart or order management systems. If you sell products on your own site, you generate fulfillment orders with the addresses and shipping information to an Amazon fulfillment center. At the same time, you can list your products for sale on Amazon.com and have Amazon fulfill from both your web site and Amazon.com.

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

How it works

1	You send your new or used products to Amazon.com's fulfillment centers.
2	Amazon stores your products in our ready-to-ship inventory.
3	Amazon fulfills orders on your behalf.
4	Fulfillment by Amazon picks your products from inventory and packages them.
5	Amazon ships the products to your customers from our network of fulfillment centers.

Three APIs

Amazon Fulfillment Web Service consists of three APIs—an inbound API, an outbound API, and an inventory API— 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 is designed to help you get your inventory to one of Amazon's fulfillment centers. Once they are in a fulfillment center, you can sell your items 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 tasks in the fulfillment process: telling Amazon about the items you've sold and where to send the items.

The outbound API is designed to help 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 your items are sold on Amazon.com, the Amazon Fulfillment Network automatically packs and ships the items to your customers.

The inventory API lets you see what is available in your inventory. It's a real-time reporting mechanism that returns your current and/or recently changed inventory supply in the Amazon fulfillment network.

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—**By integrating with our API, you can enable your customers to submit orders any time. There's no lag time while waiting for the merchant to process or batch orders.
- **Tracking and managing outbound shipment requests—**The integration continues as orders leave our fulfillment centers. You can track shipments and keep your customers aware of arrival times.

Programming Guide

Topics

This section explains the request and response structure for SOAP and Query, and provides Java code examples for common use cases. The sample program snippets used in this guide are based on the AWS libraries. To use the samples in this be sure to download the libraries.

About Requests and Responses

Topics

This section addresses how you formulate API requests. After reading this section, you should be acquainted with the basic differences between SOAP and Query, understand the components of a request, know how to authenticate a request, and understand the content of responses.

SOAP Requests

The Amazon Fulfillment Web Service supports the SOAP message protocol for calling service actions over an HTTP connection. The easiest way to use the SOAP interface with your application is to use a SOAP toolkit appropriate for your programming platform. SOAP toolkits are available for most popular programming languages and platforms.

The service's Web Services Description Language (WSDL) files (inbound WSDL and outbound WSDL) describe the operations along with the format and data types of the operations' requests and responses. Your SOAP toolkit interprets the WSDL files to provide your application access to the operations. For most toolkits, your application calls a service action using routines and classes provided or generated by the toolkit.

Structure of a SOAP Request

A SOAP request is an XML data structure that your SOAP toolkit generates and sends to the service. As described by the service WSDLs, the root element of this structure is named after the operations. You include the parameters for the request inside the root element, according to the FWS schema contained in each WSDL.

All SOAP operations include the authentication information in the `AWSAccessKeyId`, `Timestamp`, and `Signature` parameters inside the header element.

For more information about these parameters, see Amazon FWS SOAP Authentication (p. 17).

Example SOAP Request

The following example shows the XML for a SOAP message that calls the `GetServiceStatus` operation. Although you probably won't build the SOAP message for a service request manually, it is useful to see what your SOAP toolkit tries to produce when provided with the appropriate values.

The `GetServiceStatus` operation contains the header parameters common to all requests:

```
<?xml version="1.0" encoding="UTF-8"?>

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"

    xmlns:xsd="http://www.w3.org/2001/XMLSchema"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <soapenv:Header>

        <aws:AWSAccessKeyId

            soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"

            soapenv:mustUnderstand="0"

            xmlns:aws="http://security.amazonaws.com/doc/2007-01-01/"

            >0QY05JR56ZA8E56XPG82</aws:AWSAccessKeyId>

        <aws:Signature

            soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"

            soapenv:mustUnderstand="0"

            xmlns:aws="http://security.amazonaws.com/doc/2007-01-01/"

            >FvUH0tcso/u1P4vqJt7cXLPGrV8=</aws:Signature>

        <aws:Timestamp

            soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"

            soapenv:mustUnderstand="0"

            xmlns:aws="http://security.amazonaws.com/doc/2007-01-01/"
```

```
            >2009-01-23T00:43:12Z</aws:Timestamp>

    </soapenv:Header>

    <soapenv:Body>

        <GetServiceStatus

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

    </soapenv:Body>

</soapenv:Envelope>
```

Query Requests

Amazon FWS supports Query requests for calling service actions. Query requests are simple HTTP requests, using the GET or POST method with query parameters in the URL. Query requests must contain an `Action` parameter to indicate the action to be performed. The response is an XML document that conforms to a schema. You might use Query requests when a SOAP toolkit is not available for your platform or when Query requests are easier to make than a heavier SOAP equivalent.

Structure of a Query Request

Amazon FWS Query requests are URLs. The URL consists of

- **Endpoint—**The resource the request is acting on. For example, `https://fba-inbound.amazonaws.com/`.
- **Action—**The action you want to perform on the endpoint. For example, `GetServiceStatus`.
- **Parameters—**Any request parameters.

Example Query Request

The following is an example Query request to determine if the service is available:

```
https://fba-inbound.amazonaws.com/?Action=GetServiceStatus&Version=2007-05-
10&AWSAccessKeyId=1GPD3QT81GH6YJ5AV5G2&Timestamp=2008-01-21T21:18:00Z&Signa
ture=0FZXhSrODCShcfZs7AWmu35KsSg=
```

Because this format is hard to read, all Query examples in this guide are presented in the following parsed format:

```
https://fba-inbound.amazonaws.com/
?Action=GetServiceStatus
&Version=2007-05-10
&AWSAccessKeyId=1GPD3QT81GH6YJ5AV5G2
&Timestamp=2008-01-21T21:18:00Z
&Signature=0FZXhSrODCShcfZs7AWmu35KsSg=
```

The first line represents the *endpoint* of the request. This is the resource the request acts on. After the endpoint is a question mark (?), which separates the endpoint from the parameters. Each parameter is separated by an ampersand (&).

The `Action` parameter indicates the action to perform on the requested endpoint. The actions correspond to API operations.

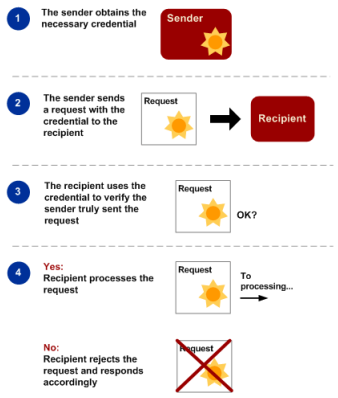
Request Authentication

Topics

This section explains how Amazon FWS authenticates your requests.

What Is Authentication?

Authentication is a process for identifying and verifying who is sending a request. The diagram below shows a simplified version of an authentication process.



General Process of Authentication

1	The sender obtains the necessary credential.
2	The sender sends a request with the credential to the recipient.
3	The recipient uses the credential to verify the sender truly sent the request.
4	If yes, the recipient processes the request. If no, the recipient rejects the request and responds accordingly.

During authentication, Amazon Web Services (AWS) verifies both the identity of the sender and whether the sender is registered to use services offered by AWS. If either test fails, the request is not processed further.

For further discussion of authentication, go to the techencylopedia.com entry for authentication. For definitions of common industry terms related to authentication, go to the RSA Laboratories Glossary.

The subsequent sections describe how FWS implements authentication to protect you and your customers' data.

Your AWS Account

To access any web services offered by AWS, 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; you can use an existing Amazon.com account login and password when creating the AWS account.

Alternately, you could create a new AWS-enabled Amazon.com account by using a new login and password. The e-mail address you provide as the account login must be valid. You'll be asked to provide a credit card or other payment method to cover the charges for any AWS products you use.

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

Your AWS Credentials

When you create an AWS account, AWS assigns you a pair of related credentials:

- 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

These are your AWS access key identifiers.

Caution

Your Secret Access Key is a secret and only you and AWS should know it. It is important to keep it confidential to protect your account. 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 must also include a digital signature. For all requests except those generated using the AWS libraries, you calculate the signature using your Secret Access Key. AWS uses the Access Key ID in the request to look up your Secret Access Key and then calculates a digital signature with the key. If the signature AWS calculates matches the signature you sent, the request is considered authentic. Otherwise, the request fails authentication and is not processed.

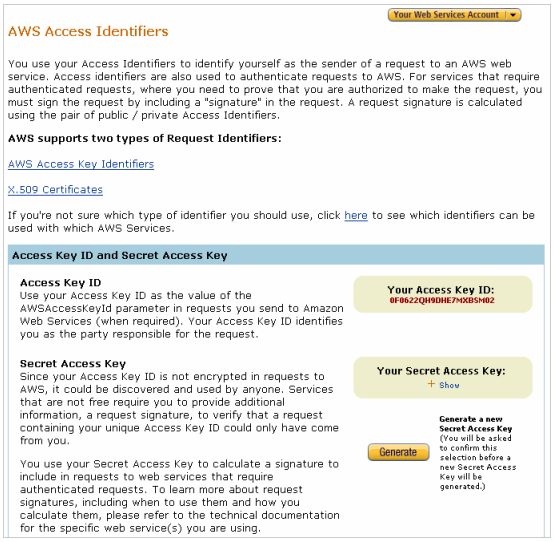
Viewing Your AWS Credentials

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 view your AWS access identifiers

1. Go to the Amazon Web Services web site at http://aws.amazon.com.
2. Point to **Your Web Services Account** to display a list of options.
3. Click **View Access Key Identifiers** and log in to your AWS account.

Your Access Key ID and Secret Access Key are displayed on the resulting **AWS Access Identifiers** page.



HMAC-SHA1 Signatures

Important

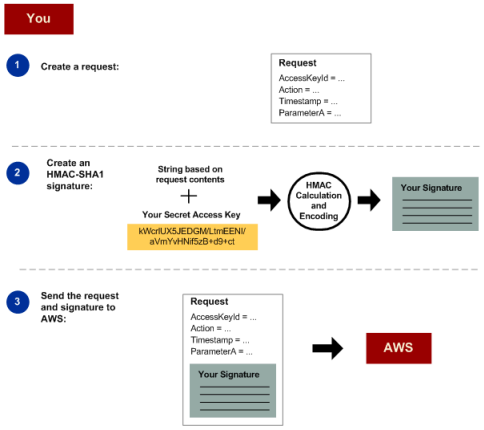
The libraries that AWS provide calculate SOAP headers for you. If you are using one of our libraries and sending in messages, you do not need to calculate your signature or time stamp.

When accessing Amazon FWS using Query, SOAP, or REST, you must provide the following items so the request can be authenticated:

- **AWSAccessKeyID—**Your AWS account is identified by your Access Key ID, which AWS uses to look up your Secret Access Key.
- **Signature—**Each request must contain a valid request signature, or the request is rejected. A request signature is calculated using your Secret Access Key, which is a shared secret known only to you and AWS.
- **Date—**Each request must contain the time stamp of the request. Depending on the API you're using (Query, SOAP, or REST), you can provide an expiration date and time for the request instead of or in addition to the time stamp. See the authentication topic for the particular API for details of what is required and allowed for that API.

Below is the series of tasks required to authenticate requests to AWS. It is assumed you have already created an AWS account (p. 13) and received an Access Key ID and Secret Access Key (p. 13).

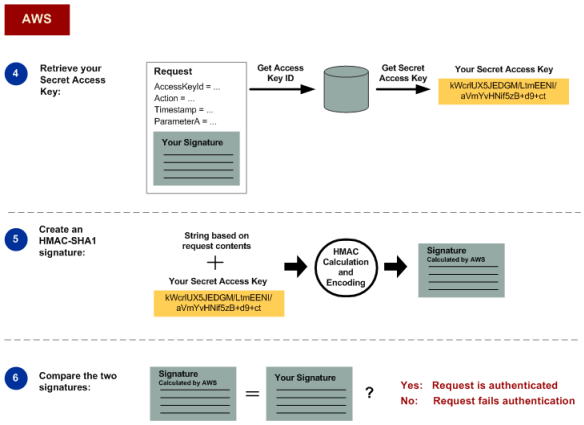
You perform the first three tasks.



Process for Authentication: Tasks You Perform

1	Construct a request to AWS.
2	Calculate a keyed-hash message authentication code (HMAC-SHA1) signature using your Secret Access Key (for information about HMAC, go to http://www.faqs.org/rfcs/rfc2104.html)
3	Include the signature and your Access Key ID in the request, and then send the request to AWS.

AWS performs the next three tasks.



Process for Authentication: Tasks AWS Performs

4	AWS uses the Access Key ID to look up your Secret Access Key.
5	AWS generates a signature from the request data and the Secret Access Key using the same algorithm you used to calculate the signature you sent in the request.
6	If the signature generated by AWS matches the one you sent in the request, the request is considered authentic. If the comparison fails, the request is discarded, and AWS returns an error response.

About the Time Stamp

The time stamp (or expiration time) you use in the request must be a `dateTime` object (for more information, go to http://www.w3.org/TR/xmlschema-2/#dateTime). Although it is not required, we recommend you provide the time stamp in the Coordinated Universal Time (Greenwich Mean Time) time zone. For example: 2007-01-31T23:59:59.183Z.

If you specify a time stamp (instead of an expiration time), the request automatically expires 15 minutes after the time stamp (in other words, AWS does not process a request if the request time stamp is more than 15 minutes earlier than the current time on AWS servers). Make sure your server's time is set correctly.

Important

If you are using .NET you must not send overly specific time stamps, due to different interpretations of how extra time precision should be dropped. To avoid overly specific time stamps, manually construct `dateTime` objects with no more than millisecond precision.

Java Sample Code for Base64 Encoding

HMAC-SHA1 request signatures must be base64 encoded. The following Java sample code shows how to perform base64 encoding.

```
package amazon.webservices.common;
/**
* This class defines common routines for encoding * data in AWS Platform re
quests.
*/
public class Encoding {
/**
* Performs base64-encoding of input bytes.
*
* @param rawData * Array of bytes to be encoded.
* @return * The base64 encoded string representation of rawData.
*/
public static String EncodeBase64(byte[] rawData) {
return Base64.encodeBytes(rawData);
}
}
```

Java Sample Code for Calculating HMAC-SHA1 Signatures

The following Java code sample shows how to calculate an HMAC request signature.

```
package amazon.webservices.common;

import java.security.SignatureException;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

/**
* This class defines common routines for generating
* authentication signatures for AWS Platform requests.
*/
public class Signature {
private static final String HMAC_SHA1_ALGORITHM = "HmacSHA1";
```

```
/**
* Computes RFC 2104-compliant HMAC signature.
* * @param data
* The data to be signed.
* @param key
* The signing key.
* @return
* The Base64-encoded RFC 2104-compliant HMAC signature.
* @throws
* java.security.SignatureException when signature generation fails
*/
public static String calculateRFC2104HMAC(String data, String key)
throws java.security.SignatureException
{
String result;
try {

// get an hmac_sha1 key from the raw key bytes
SecretKeySpec signingKey = new SecretKeySpec(key.getBytes(), HMAC_SHA1_AL
GORITHM);

// get an hmac_sha1 Mac instance and initialize with the signing key
Mac mac = Mac.getInstance(HMAC_SHA1_ALGORITHM);
mac.init(signingKey);

// compute the hmac on input data bytes
byte[] rawHmac = mac.doFinal(data.getBytes());

// base64-encode the hmac
result = Encoding.EncodeBase64(rawHmac);

} catch (Exception e) {
throw new SignatureException("Failed to generate HMAC : " + e.getMessage());
}
return result;
}
}
```

Amazon FWS SOAP Authentication

For security purposes, Amazon Fulfillment Web Service uses your AWS credentials and an HMAC-SHA1 signature in the header of each SOAP request. Specifically, you need to include the following required elements in the header:

- `AWSAccessKeyId`: Your AWS Access Key ID
- `Timestamp`: This must be a `dateTime` in the Coordinated Universal Time (Greenwich Mean Time) time zone, such as 2005-01-31T23:59:59:183Z (for more information, see About the Time Stamp (p. 16)). The request expires 15 minutes after the time stamp.

  Note

  Due to different interpretations regarding how extra time precision should be dropped, .NET users should take care not to send overly specific time stamps. This can be accomplished by manually constructing `dateTime` objects with no more than millisecond precision.

- `Signature`: The HMAC-SHA1 signature calculated from the concatenation of `Action+Timestamp`, using your AWS Secret Access Key as the key. For example, in the following sample request, the value

of the `Signature` element is the HMAC-SHA1 digest of this string:
`CreateQueue2005-01-31T23:59:59.183Z`

To calculate the signature

1. Concatenate the values of the `Action` and `Timestamp` request parameters, in that order.
2. Calculate an HMAC-SHA1 signature, using your Secret Access Key as the key.
3. Convert the resulting value to base64.
4. Pass the final value in the `Signature` parameter of the SOAP request.

Example SOAP Header

```
<soapenv:Header>

    <aws:AWSAccessKeyId

        soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"

        soapenv:mustUnderstand="0"

        xmlns:aws="http://security.amazonaws.com/doc/2007-01-01/"

        >0QY05JR56ZA8E56XPG82</aws:AWSAccessKeyId>

    <aws:Signature

        soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"

        soapenv:mustUnderstand="0"

        xmlns:aws="http://security.amazonaws.com/doc/2007-01-01/"

        >FvUH0tcso/u1P4vqJt7cXLPGrV8=</aws:Signature>

    <aws:Timestamp

        soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next"

        soapenv:mustUnderstand="0"

        xmlns:aws="http://security.amazonaws.com/doc/2007-01-01/"

        >2009-01-23T00:43:12Z</aws:Timestamp>

</soapenv:Header>
```

Amazon FWS Query Authentication

For security purposes, Amazon Fulfillment Web Service uses your AWS credentials and an HMAC-SHA1 signature in the top of each Query request. Specifically, you need to include the following required elements in each request:

- `AWSAccessKeyId`: Your AWS Access Key ID

- `Timestamp`: This must be a `dateTime` in the Coordinated Universal Time (Greenwich Mean Time) time zone, such as 2005-01-31T23:59:59:183Z (for more information, see About the Time Stamp (p. 16)). The request expires 15 minutes after the time stamp.

  Note

  Due to different interpretations regarding how extra time precision should be dropped, .NET users should take care not to send overly specific time stamps. This can be accomplished by manually constructing `dateTime` objects with no more than millisecond precision.

- `Signature`: The HMAC-SHA1 signature calculated from the concatenation of `Action+Timestamp`, using your AWS Secret Access Key as the key. For example, in the following sample request, the value of the `Signature` element is the HMAC-SHA1 digest of this string: `CreateQueue2005-01-31T23:59:59.183Z`

To calculate the signature

1. Concatenate the values of the `Action` and `Timestamp` request parameters, in that order.
2. Calculate an HMAC-SHA1 signature, using your Secret Access Key as the key.
3. Convert the resulting value to base64.
4. Pass the final value in the `Signature` parameter of the Query request.

Example Query Snippet

```
&AWSAccessKeyId=0QY05JR56ZA8E56XPG82

&Signature=FvUH0tcso/u1P4vqJt7cXLPGrV8=

&Timestamp=2009-01-23T00:43:12Z
```

Responses

This section describes the structure of a response to an Amazon FWS request.

Response Messages

In response to an action request, FWS returns an XML data structure that contains the results of the request. Other than the use of a message envelope in the case of SOAP, the structure for the results is the same for Query, SOAP, and REST responses.

The Structure of a Response

The response message is returned in an XML element named after the action. For example, the `GetServiceStatus` action returns a response element named `GetServiceStatusResponse`.

If a request is successful, a `ResponseStatus` element is returned (as a child of the main response element) containing a `StatusCode` element with a value of `Success`. It also contains a `RequestId` element.

If a request is unsuccessful, the main response element is named `Response`, irrespective of the action requested. This element contains an `Errors` element, with one or more `Error` child elements. Each `Error` includes:

- A `Code` that identifies the type of error that occurred
- A `Message` that describes the error condition in a human-readable form

The following is an example of a successful response:

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

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

    <soapenv:Body>

        <ns1:GetServiceStatusResponse

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

            <ns1:GetServiceStatusResult>

                <ns1:Status>2008-01-23 16:43:12.975 PST service responding.

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

            </ns1:GetServiceStatusResult>

            <ns1:ResponseMetadata>

              <ns1:RequestId>5fa04c3a-4bb4-485d-926c-e3c8002c192e</ns1:Request
Id>

            </ns1:ResponseMetadata>

        </ns1:GetServiceStatusResponse>

    </soapenv:Body>
```

```
</soapenv:Envelope>
```

The following is an example of an error response:

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

<GetServiceStatusErrorResponse

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

    <Error>

        <Code>InvalidClientTokenId</Code>

        <Message>The AWS Access Key Id you provided does not exist in our

            records.</Message>

    </Error>

    <RequestID>42fd0235-1c23-4e1e-872c-1fa3e4a4b050</RequestID>

</GetServiceStatusErrorResponse>
```

Related Topics

- SOAP Requests (p. 9)
- Query Requests (p. 11)

Inbound Use Cases

Topics

The use cases presented in this section represent the most common requests used with the inbound API. The use cases are part of the overall inbound process that uses both physical acts and the inbound API to collect and retrieve information for packaging your shipment to an Amazon fulfillment center. For more information, see The Inbound Process (p. 22).

The Java examples offered in this section consist of import statements and code snippets specific to the use case. The snippets require that you incorporate them in your own methods.

Tip

If you have further questions about parameters and values used with the requests, go to the Inbound Operations narrative in the Amazon Fulfillment Web Service API Reference.

The Inbound Process

Creating an inbound shipment involves both physical and programmatic tasks. This section details the process in full and provides instructions or links to Amazon.com help for the tasks that do not relate specifically to the API.

Tasks to Create an Inbound Shipment

1	Create a listing.
2	Get a fulfillment identifier for each item.
3	Prepare the items for shipping.
4	Create a new shipment.
5	Label shipment.
6	Ship your items.
7	Mark your shipment as "Shipped."

Create a Listing

Task Type:	Physical
Associated Amazon.com Help:	Products & Listings

or:

Task Type:	API
Amazon Marketplace Web Service:	SubmitFeed Operation

When you create a listing, you tell Amazon about the product you intend to sell. Some product information, such as an item's package dimension and weight, is also necessary to send your items to an Amazon fulfillment center. During the listing creation process, Amazon attempts to match your Merchant SKU with a catalog entry. If your product already exists in the catalog, the match is made. If your product doesn't exist in the catalog, you have to go through the process of creating a new product and describing the product in more detail.

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. For help with creating a listing, go to the Help for Products & Listings.

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

Get a Fulfillment Identifier

Task Type:	API
Associated Use Case:	Use Case: Getting a Fulfillment Identifier for MSKU (p. 27)

When you make the operational call to get a fulfillment identifier, you're requesting an Amazon Fulfillment Network SKU (FNSKU) for each supplied merchant item. This operation, which requires that you have already created a listing, is required whenever you need to register items for Amazon fulfillment that require labeling. For more information about creating a listing, see the Creating a Listing section. To see an example application of this API, see Use Case: Getting a Fulfillment Identifier for MSKU (p. 27).

Note

In the `GetFulfillmentIdentifierForMSKU` response, if the `FulfillmentNetworkSKU` equals the `ASIN`, the item is stickerless (see Prepare the Items for Shipping (p. 23)).

Prepare the Items for Shipping

Task Type:	Physical
Associated Amazon.com Help:	Packaging Products for Shipping

To prepare your items for shipping, you must first determine if your items will be commingled with other Amazon inventory (also called stickerless), or stickered.

Note

Only new items can be commingled. For more information, go to the Help for Stickerless, Commingled Inventory.

If you are sending stickerless items, ensure that your items comply with Amazon's product packaging requirements, then create a new shipment. If you are sending stickered items, ensure that your items comply with Amazon's product packaging requirements, and format and label each item with the `FulfillmentNetworkSKU` you received from the `getFulfillmentIdentiferForMSKU` operation. After that is complete, create a new shipment. For more information, go to the Help on Packaging Products for Shipping.

Create a New Shipment

Task Type:	API
Associated Use Case:	Use Case: Creating a New Shipment (p. 29)

When you create a new shipment, you're requesting Amazon to send you information about which items to group together in a shipment and which fulfillment center to address that shipment. To see an example of this API, see Use Case: Creating a New Shipment (p. 29).

Label the Shipment

Task Type:	Physical
Associated Amazon.com Help	Labeling Packages

Once you have a grouped your items as instructed in the Create a New Shipment task, pack them together and label each shipment with the information sent by Amazon as shown in the Creating a New Shipment (p. 29) use case. For more information about shipment labeling, go to the Help for Labeling Packages. For more information about packaging for shipment, go to the Help for Packaging Products for Shipping.

Ship Your Items

Task Type:	Physical
Associated Amazon.com Help	Shipping and Routing Requirements

For more information about shipping, go to the Help for Shipping and Routing Requirements.

Mark Your Shipment as "Shipped"

Task Type:	API
Associated Use Case:	Use Case: Marking a Shipment as Shipped (p. 32)

Once you've sent your shipment to an Amazon fulfillment center, use the inbound API to set the status of your shipment to `Shipped`. To see an example application of this API, see Use Case: Marking a Shipment as Shipped (p. 32).

Tip

For information about item label, packing slip, and shipping label specifications, see Inbound Label and Packing Slip Specifications (p. 55)

Use Case: Checking the Service Status (Inbound)

Description

This use case checks to see whether the inbound service is running. It shows an application of the API action `GetServiceStatus` using Java.

Outcome

If the service is running, Amazon FWS returns a message stating that the service is responding, along with the date, time, and the service version number. If the service is not running Amazon FWS returns an error.

Example Code

Import Statement

```
import com.amazonaws.fws.inbound.*;
import com.amazonaws.fws.inbound.model.*;
```

Code Snippet

```
try

{

    AmazonFBAInbound service =

        new AmazonFBAInboundClient(AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY);


    GetServiceStatus request = new GetServiceStatus();


    GetServiceStatusResponse response = service.getServiceStatus(request);


    GetServiceStatusResult result = response.getGetServiceStatusResult();


    System.out.println("Status: " + result.getStatus());

}

catch (AmazonFBAInboundException e)

{

   System.out.println("Error calling GetServiceStatus: " + e);

}
```

Programming Notes

- Use your Access Key ID for the `AWS_ACCESS_KEY_ID` value.
- Use your Secret Access Key for the `AWS_SECRET_ACCESS_KEY` value.

Use Case: Getting a Fulfillment Identifier for MSKU

Description

This use case requests the Amazon Fulfillment Network SKU (FNSKU) for each supplied merchant item. It shows an application of the API action `GetFulfillmentIdentifierForMSKU` using Java.

Outcome

Amazon FWS returns an aggregate that contains properties and identifiers for an item that Amazon can fulfill.

Example Code

Import Statement

```
import com.amazonaws.fws.inbound.*;
import com.amazonaws.fws.inbound.model.*;
```

Code Snippet

```
String accessKeyId = "";

String secretAccessKey = "";

AmazonFBAInbound service = new AmazonFBAInboundClient(accessKeyId, secretAccess
Key);



String merchantSKU = "my_merchant_sku_for_stickered";



MerchantItem item = new MerchantItem();

item.setCondition(ItemCondition.USED_VERY_GOOD);

item.setMerchantSKU(merchantSKU);



try {

  GetFulfillmentIdentifierForMSKU getFulfillmentIdentifierRequest = new GetFul
fillmentIdentifierForMSKU();

  getFulfillmentIdentifierRequest.getMerchantItem().add(item);



  //Get the fulfillment identifier required for labeling items for stickered
merchants

  GetFulfillmentIdentifierForMSKUResponse response = service.getFulfillmentIden
tifier(getFulfillmentIdentifierRequest);
```

```
  List fulfillmentItems = response.getGetFulfillmentIdentifierForMSKUResult().get
FulfillmentItem();

  for (FulfillmentItem fulfillmentItem : fulfillmentItems) {

      //Ensure we have the active FNSKU

      if(fulfillmentItem.isActive()) {

      //Get the FNSKU to print on lablel

     String fulfillmentNetworkSKU = fulfillmentItem.getFulfillmentNetworkSKU();


      System.out.println("fulfillmentNetworkSKU : " + fulfillmentNetworkSKU);

      }

  }

}catch (AmazonFBAInboundException ex) {

  System.out.println("GOT AN EXCEPTION: " + ex);

}
```

Programming Notes

- Set `accessKeyId` to your Access Key ID.
- Set `secretAccessKey` to your Secret Access Key.
- Replace the default parameters with your information. For help with these parameters, see the `GetFulfillmentIdentifierForMSKU` operation.

Use Case: Creating a New Shipment

Description

This use case shows a request for information required to create a shipment (or set of shipments) to a fulfillment center. It shows an application of the API actions `GetInboundShipmentPreview` and `PutInboundShipment` using Java.

Outcome

Amazon FWS returns the information needed to create a set of shipments for a given set of items and a source address.

Example Code

Import Statement

```
import com.amazonaws.fws.inbound.*;
import com.amazonaws.fws.inbound.model.*;
```

Code Snippet

```
String accessKeyId = "";

String secretAccessKey = "";

AmazonFBAInbound service = new AmazonFBAInboundClient(accessKeyId, secretAccess
Key);


String merchantSKU = "my_merchant_sku";


  MerchantSKUQuantityItem mSkuQuantityItem = new MerchantSKUQuantityItem();

  mSkuQuantityItem.setMerchantSKU(merchantSKU);

  mSkuQuantityItem.setQuantity(3);


  Address address = new Address();

  address.setAddressLine1("505 S 5th Street");

  address.setCity("Seattle");

  address.setCountryCode("US");

  address.setName("My Ship From Address");

  address.setPostalCode("98101");

  address.setStateOrProvinceCode("WA");
```

```
try {

   GetInboundShipmentPreview getInboundShipmentPreviewRequest = new GetInbound
ShipmentPreview();

   getInboundShipmentPreviewRequest.getMerchantSKUQuantityItem().add(mSkuQuant
ityIte);

    getInboundShipmentPreviewRequest.setShipFromAddress(address);



    // Determine which Amazon Fulfillment Center my item should go to

    GetInboundShipmentPreviewResponse resp = service.getInboundShipmentPre
view(getInboundShipmentPreviewRequest);



    // Amazon Fulfillment may separate items in a ShipmentPreviewRequest 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 shipmentPreviews = resp.getGetInboundShipmentPreviewResult().getShip
mentPreview();



    // 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 Fulfill
ment 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.getDestinationFulfillment
Center();

       fulfillmentCenterAddress = shipmentGroup.getShipToAddress();



      putInboundShipmentRequest.setDestinationFulfillmentCenter(destinationFul
fillmentCenter);

      putInboundShipmentRequest.getMerchantSKUQuantityItem().add(mSkuQuantity
Item);

       putInboundShipmentRequest.setShipFromAddress(fulfillmentCenterAddress);



       shipmentID = shipmentGroup.getShipmentId()



       putInboundShipmentRequest.setShipmentId(shipmentID);

       putInboundShipmentRequest.setShipmentName("My First Shipment");



        // Create the shipment

       service.putInboundShipment(putInboundShipmentRequest);

   }


}catch (AmazonFBAInboundException ex) {

  System.out.println("GOT AN EXCEPTION: " + ex);

}
```

Programming Notes

- Set `accessKeyId` to your Access Key ID.
- Set `secretAccessKey` to your Secret Access Key.
- Replace the default parameters with your own information. For help with these parameters, see the `GetInboundShipmentPreview` operation in the Amazon Fulfillment Web Service API Reference.

Use Case: Marking a Shipment as Shipped

Description

In this use case, we set the inbound shipment status to `Shipped`. It shows an application of the API action `SetInboundShipmentStatus` using Java.

Outcome

For a successful request, Amazon FWS returns no information. Otherwise Amazon FWS returns an error.

Example Code

Import Statement

```
import com.amazonaws.fws.inbound.*;
import com.amazonaws.fws.inbound.model.*;
```

Code Snippet

```
String accessKeyId = "";

String secretAccessKey = "";

AmazonFBAInbound service = new AmazonFBAInboundClient(accessKeyId, secretAccess
Key);


// the shipment ID is retrieved from ShipmentPreview results

public void markShipmentAsShipped(String shipmentID) {

  try {


    SetInboundShipmentStatus setInboundShipmentStatusRequest = new SetInbound
ShipmentStatus();

    setInboundShipmentStatusRequest.setShipmentId(shipmentID);

    setInboundShipmentStatusRequest.setShipmentStatus(ShipmentStatus.SHIPPED);


    service.setInboundShipmentStatus(setInboundShipmentStatusRequest);

  }catch (AmazonFBAInboundException ex) {

    System.out.println("GOT AN EXCEPTION: " + ex);

  }

}
```

Programming Notes

- Set `accessKeyId` to your Access Key ID.
- Set `secretAccessKey` to your Secret Access Key.

Use Case: Finding Partially Received Shipment Items

Description

This use case requests returns the first set of inbound shipments a merchant has created. It shows an application of the API actions `ListInboundShipments`, `ListInboundShipmentsByNextToken`, `ListInboundShipmentItems`, and `ListInboundShipmentItemsByNextToken` using Java.

Outcome

Amazon FWS returns the first set of inbound shipments a merchant has created

Example Code

Import Statement

```
import java.util.List;
import java.util.ArrayList;
import com.amazonaws.fws.inbound.*;
import com.amazonaws.fws.inbound.model.*;
```

Code Snippet

```
import com.amazonaws.fba.inbound.*;

import com.amazonaws.fba.inbound.model.*;


public static void main(String[] args) {

        String accessKeyId = "";

        String secretAccessKey = "";


        AmazonFBAInbound service = new AmazonFBAInboundClient(accessKeyId,
secretAccessKey);


        try {

            getAllPartiallyReceivedShipmentItems(service);

        } catch (AmazonFBAInboundException ex) {

            System.out.println("GOT EXCEPTION: " + ex);

        }

    }
```

```
    private static List getAllPartiallyReceivedShipmentItems(AmazonFBAInbound
service)

    throws AmazonFBAInboundException

    {

       ListInboundShipments listInboundShipmentsRequest = new ListInboundShip
ments();



        // Add all the shipment statuses we are interested in

        List shipmentStatuses = new ArrayList();

        shipmentStatuses.add(ShipmentStatus.SHIPPED);

        shipmentStatuses.add(ShipmentStatus.IN_TRANSIT);

        shipmentStatuses.add(ShipmentStatus.RECEIVING);



        // Note that this call is paginated, so set the maximum number of
shipments

        // we want back for each operation invocation

        int maxCount = 100;



        listInboundShipmentsRequest.setShipmentStatus(shipmentStatuses);

        listInboundShipmentsRequest.setMaxCount(maxCount);



        ListInboundShipmentsResponse listInboundShipmentsResponse = null;



       listInboundShipmentsResponse = service.listInboundShipments(listInbound
ShipmentsRequest);



       ListInboundShipmentsResult listInboundShipmentsResult = listInboundShip
mentsResponse.getListInboundShipmentsResult();



       List inboundShipmentData = listInboundShipmentsResult.getShipmentData();
```

```
        List myShipmentIds = new ArrayList();

        for (InboundShipmentData data : inboundShipmentData)

        {

            myShipmentIds.add(data.getShipmentId());

        }


        // While there are more results, keep paginating through them

        String nextToken = listInboundShipmentsResult.getNextToken();

        boolean hasMoreResults = listInboundShipmentsResult.isHasNext();

        while (hasMoreResults)

        {

           ListInboundShipmentsByNextToken request = new ListInboundShipments
ByNextToken();

            request.setNextToken(nextToken);


            ListInboundShipmentsByNextTokenResponse response = service.listIn
boundShipmentsByNextToken(request);

            ListInboundShipmentsByNextTokenResult result = response.getListIn
boundShipmentsByNextTokenResult();


            nextToken = result.getNextToken();

            hasMoreResults = result.isHasNext();


            for (InboundShipmentData data : result.getShipmentData())

            {

                myShipmentIds.add(data.getShipmentId());

            }


        }


        // Now, for each of the shipments, determine which items have not yet
```

```
been fully

        // received by Amazon.

        List partiallyReceivedShipmentItems = new ArrayList();

        for (String shipmentId : myShipmentIds)

        {

            ListInboundShipmentItems listInboundShipmentItemsRequest = new
ListInboundShipmentItems();



            listInboundShipmentItemsRequest.setMaxCount(maxCount);

            listInboundShipmentItemsRequest.setShipmentId(shipmentId);



           ListInboundShipmentItemsResponse listInboundShipmentItemsResponse
= service.listInboundShipmentItems(listInboundShipmentItemsRequest);

            ListInboundShipmentItemsResult listInboundShipmentItemsResult =
listInboundShipmentItemsResponse.getListInboundShipmentItemsResult();



          for (InboundShipmentItem item : listInboundShipmentItemsResult.get
ShipmentItem())

            {

                if (item.getQuantityShipped() > item.getQuantityReceived())

                    partiallyReceivedShipmentItems.add(item);

            }



            // For large shipments with lots of shipment items we will have to
 paginate

            nextToken = listInboundShipmentItemsResult.getNextToken();

            hasMoreResults = listInboundShipmentItemsResult.isHasNext();

            while (hasMoreResults)

            {

                ListInboundShipmentItemsByNextToken request = new ListInbound
ShipmentItemsByNextToken();

                request.setNextToken(nextToken);
```

```
                ListInboundShipmentItemsByNextTokenResponse response = ser
vice.listInboundShipmentItemsByNextToken(request);

              ListInboundShipmentItemsByNextTokenResult result = response.get
ListInboundShipmentItemsByNextTokenResult();


                nextToken = result.getNextToken();

                hasMoreResults = result.isHasNext();


                for (InboundShipmentItem item : result.getShipmentItem())

                {

                   if (item.getQuantityShipped() > item.getQuantityReceived())


                        partiallyReceivedShipmentItems.add(item);

                }

            }

        }


        return partiallyReceivedShipmentItems;

    }
```

Programming Notes

- Set `accessKeyId` to your Access Key ID.
- Set `secretAccessKey` to your Secret Access Key.
- Set `maxCount` to the number of returns you want in each call.

Outbound Use Cases

Topics

The use cases presented in this section represent the most common requests used with the outbound API. The outbound use cases are straightforward, with a one-to-one correlation between a single API operation and each use case action.

The Java examples offered in this section consist of import statements and code snippets specific to the use case. You must incorporate the snippets in your own methods to test them.

Tip

If you have further questions about parameters and values used with the requests, see the outbound operations narrative in the Amazon Fulfillment Web Service API Reference.

Use Case: Checking the Service Status (Outbound)

Description

This use case checks to see whether the outbound service is running. It shows an application of the API action `GetServiceStatus` using Java.

Outcome

If the service is running Amazon FWS returns a message stating that the service is responding, along with the date, time, and the service version number. If the service is not running Amazon FWS returns an error.

Example Code

Import Statement

```
import com.amazonaws.fba.outbound.AmazonFBAOutbound;
import com.amazonaws.fba.outbound.AmazonFBAOutboundException;
import com.amazonaws.fba.outbound.model.GetServiceStatus;
import com.amazonaws.fba.outbound.model.GetServiceStatusResponse;
import com.amazonaws.fba.outbound.model.GetServiceStatusResult;
```

Code Snippet

```
try

{

    AmazonFBAOutbound service =

        new AmazonFBAOutboundClient(AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY);



    GetServiceStatus request = new GetServiceStatus();


    GetServiceStatusResponse response = service.getServiceStatus(request);


    GetServiceStatusResult result = response.getGetServiceStatusResult();


    System.out.println("Status: " + result.getStatus());

}

catch (AmazonFBAOutboundException e)

{
```

```
    System.out.println("Error calling GetServiceStatus: " + e);

}
```

Programming Notes

- Use your Access Key ID for the `AWS_ACCESS_KEY_ID` parameter in the AmazonFBAOutboundClient constructor.
- Use your Secret Access Key for the `AWS_SECRET_ACCESS_KEY` parameter in the AmazonFBAOutboundClient constructor.

Use Case: Creating a Fulfillment Order

Description

This use case generates a request for Amazon to ship items from the merchant's inventory to a destination address. It shows an application of the API action `CreateFulfillmentOrder` using Java.

Tip

You can see the status of your orders in the Seller Central Manage Orders page.

Outcome

For a successful request, Amazon FWS returns no information. Otherwise Amazon FWS returns an error.

Example Code

Import Statement

```
import com.amazonaws.fba.outbound.AmazonFBAOutbound;
import com.amazonaws.fba.outbound.AmazonFBAOutboundException;
import com.amazonaws.fba.outbound.model.Address;
import com.amazonaws.fba.outbound.model.CreateFulfillmentOrder;
import com.amazonaws.fba.outbound.model.CreateFulfillmentOrderItem;
import com.amazonaws.fba.outbound.model.CreateFulfillmentOrderResponse;
import java.util.ArrayList;
import java.util.List;
```

Code Snippet

```
try

{

    AmazonFBAOutbound service =

        new AmazonFBAOutboundClient(AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY);



    CreateFulfillmentOrder request = new CreateFulfillmentOrder();


    request.setMerchantFulfillmentOrderId("TEST-00000001");

    request.setDisplayableOrderId("TEST-00000001");

    request.setDisplayableOrderDateTime("2007-08-02T00:00:00Z");

    request.setDisplayableOrderComment("Thank you for your order!");

    request.setShippingSpeedCategory("Standard");
```

```
    Address address = new Address();

    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<CreateFulfillmentOr
derItem>(1);


    CreateFulfillmentOrderItem item1 = new CreateFulfillmentOrderItem();

    item1.setMerchantSKU("MSKU-00000001");

    item1.setMerchantFulfillmentOrderItemId("TEST-00000001-1");

    item1.setQuantity(2);

    items.add(item1);


    request.setItem(items);


   CreateFulfillmentOrderResponse response = service.createFulfillmentOrder(re
quest);

}

catch (AmazonFBAOutboundException e)

{

   System.out.println("Error calling CreateFulfilmentOrder: " + e);

}
```

Programming Notes

- Use your Access Key ID for the `AWS_ACCESS_KEY_ID` parameter in the AmazonFBAOutboundClient constructor.
- Use your Secret Access Key for the `AWS_SECRET_ACCESS_KEY` parameter in the AmazonFBAOutboundClient constructor.
- Replace the default parameters with information from your own inventory. For help with these parameters, see the `CreateFulfillmentOrder` operation in the Amazon Fulfillment Web Service API Reference.

Use Case: Getting a Fulfillment Order

Description

This use case requests detailed information about a fulfillment order. It shows an application of the API action `GetFulfillmentOrder` using Java.

Outcome

Amazon FWS returns all relevant shipping information, including the original fulfillment order request, the status of both the order and its items in the Amazon Fulfillment Network, and the shipments that have been generated to fulfill the order.

Example Code

Import Statement

```
import com.amazonaws.fba.outbound.AmazonFBAOutbound;
import com.amazonaws.fba.outbound.AmazonFBAOutboundException;
import com.amazonaws.fba.outbound.model.FulfillmentOrder;
import com.amazonaws.fba.outbound.model.FulfillmentOrderItem;
import com.amazonaws.fba.outbound.model.GetFulfillmentOrder;
import com.amazonaws.fba.outbound.model.GetFulfillmentOrderResponse;
import com.amazonaws.fba.outbound.model.GetFulfillmentOrderResult;
import java.util.List;
```

Code Snippet

```
try

{

    AmazonFBAOutbound service =

        new AmazonFBAOutboundClient(AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY);



    GetFulfillmentOrder request = new GetFulfillmentOrder();


    request.setMerchantFulfillmentOrderId("TEST-00000001");


   GetFulfillmentOrderResponse response = service.getFulfillmentOrder(request);



   GetFulfillmentOrderResult result = response.getGetFulfillmentOrderResult();
```

```
    FulfillmentOrder order = result.getFulfillmentOrder();

    List<FulfillmentOrderItem> items = result.getFulfillmentOrderItem();


    System.out.println("Order: " + order.getMerchantFulfillmentOrderId());


    for (FulfillmentOrderItem item : items)

    {

        System.out.println("\tItem: " + item.getMerchantFulfillmentOrder
ItemId());

    }

}

catch (AmazonFBAOutboundException e)

{

   System.out.println("Error calling GetFulfillmentOrder: " + e);

}
```

Programming Notes

- Use your Access Key ID for the `AWS_ACCESS_KEY_ID` parameter in the AmazonFBAOutboundClient constructor.
- Use your Secret Access Key for the `AWS_SECRET_ACCESS_KEY` parameter in the AmazonFBAOutboundClient constructor.
- Replace the `TEST-00000001` parameter in the setMerchantFulfillmentOrderId method with your own `MerchantFulfillmentOrderId` (used in Use Case: Creating a Fulfillment Order (p. 42)).

Use Case: Listing All Fulfillment Orders

Description

This use case requests a list of all fulfillment orders to be fulfilled on a specified date. It shows an application of the API actions `ListAllFulfillmentOrders` and `ListAllFulfillmentOrdersByNextToken` using Java.

Outcome

Amazon FWS returns all relevant shipping information, including the original fulfillment order request and the status of both the order and its items in the Amazon Fulfillment Network.

Example Code

Import Statement

```
import com.amazonaws.fba.outbound.AmazonFBAOutbound;
import com.amazonaws.fba.outbound.AmazonFBAOutboundException;
import com.amazonaws.fba.outbound.model.FulfillmentOrder;
import com.amazonaws.fba.outbound.model.ListAllFulfillmentOrders;
import com.amazonaws.fba.outbound.model.ListAllFulfillmentOrdersResponse;
import com.amazonaws.fba.outbound.model.ListAllFulfillmentOrdersResult;
import com.amazonaws.fba.outbound.model.ListAllFulfillmentOrdersByNextToken;
import com.amazonaws.fba.outbound.model.ListAllFulfillmentOrdersByNextTokenRe
sponse;
import com.amazonaws.fba.outbound.model.ListAllFulfillmentOrdersByNextTokenRes
ult;
import java.util.List;
```

Code Snippet

```
try

{

    AmazonFBAOutbound service =

        new AmazonFBAOutboundClient(AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY);



    ListAllFulfillmentOrders request = new ListAllFulfillmentOrders();


    request.setNumberOfResultsRequested(25);


    ListAllFulfillmentOrdersResponse response = service.listAllFulfillment
Orders(request);
```

```
   ListAllFulfillmentOrdersResult result = response.getListAllFulfillmentOrder
sResult();


    List<FulfillmentOrder> orders = result.getFulfillmentOrder();

    boolean hasNext = result.isHasNext();

    String nextToken = result.getNextToken();


    while (hasNext)

    {

        for (FulfillmentOrder order : orders)

        {

             System.out.println("Order: " + order.getMerchantFulfillmentOr
derId());

        }


        ListAllFulfillmentOrdersByNextToken nextRequest = new ListAllFulfill
mentOrdersByNextToken();

        nextRequest.setNextToken(nextToken);


        ListAllFulfillmentOrdersByNextTokenResponse nextResponse =

            service.listAllFulfillmentOrdersByNextToken(nextRequest);


        ListAllFulfillmentOrdersByNextTokenResult nextResult =

            nextResponse.getListAllFulfillmentOrdersByNextTokenResult();


        orders = nextResult.getFulfillmentOrder();

        hasNext = nextResult.isHasNext();

        nextToken = nextResult.getNextToken();

    }

}

catch (AmazonFBAOutboundException e)
```

```
{

   System.out.println("Error calling ListAllFulfillmentOrders: " + e);

}
```

Programming Notes

- Use your Access Key ID for the `AWS_ACCESS_KEY_ID` parameter in the AmazonFBAOutboundClient constructor.
- Use your Secret Access Key for the `AWS_SECRET_ACCESS_KEY` parameter in the AmazonFBAOutboundClient constructor.
- Replace the `25` parameter in the setNumberOfResultsRequested method with your own `MerchantFulfillmentOrderId` (used in Use Case: Creating a Fulfillment Order (p. 42)).

Use Case: Canceling a Fulfillment Order

Description

This use case requests Amazon not to fulfill an existing fulfillment order. It shows an application of the API action `CancelFulfillmentOrder` using Java.

Outcome

For a successful request, Amazon FWS returns no information. Otherwise Amazon FWS returns an error.

Example Code

Import Statement

```
import com.amazonaws.fba.outbound.AmazonFBAOutbound;
import com.amazonaws.fba.outbound.AmazonFBAOutboundException;
import com.amazonaws.fba.outbound.model.CancelFulfillmentOrder;
import com.amazonaws.fba.outbound.model.CancelFulfillmentOrderResponse;
```

Code Snippet

```
try

{

    AmazonFBAOutbound service =

        new AmazonFBAOutboundClient(AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY);



    CancelFulfillmentOrder request = new CancelFulfillmentOrder();


    request.setMerchantFulfillmentOrderId("TEST-00000001");


   CancelFulfillmentOrderResponse response = service.cancelFulfillmentOrder(re
quest);

}

catch (AmazonFBAOutboundException e)

{

   System.out.println("Error calling CancelFulfillmentOrder: " + e);

}
```

Programming Notes

- Use your Access Key ID for the `AWS_ACCESS_KEY_ID` parameter in the AmazonFBAOutboundClient constructor.
- Use your Secret Access Key for the `AWS_SECRET_ACCESS_KEY` parameter in the AmazonFBAOutboundClient constructor.
- Replace the `TEST-00000001` parameter in the setMerchantFulfillmentOrderId method with your own `MerchantFulfillmentOrderId` (used in Use Case: Creating a Fulfillment Order (p. 42)).

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. 52).

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`

Appendix

Topics

This appendix covers the shipping label, packing slip, pallet, and font specifications you'll need to follow to ensure that your inbound shipment is promptly received at the fulfillment center.

Inbound Label and Packing Slip Specifications

Topics

Amazon FWS uses two types of labels and a packing slip:

- **Item label—** the label that you place on each individual item you send to the Amazon fulfillment center. You do not need to place labels on items that will be commingled.
- **Shipping label—**the label you affix on the outside of the box or carton you ship to the Amazon fulfillment center.
- **Packing slip—** is the single sheet of paper contained in each box or carton you ship to the Amazon fulfillment center.

If you plan to design your own labels and packing slip, be sure to submit printed samples of the designs to FBA. FBA will test your designs for both scannability and rigor (smudging, fading, or smearing).

Tip

We recommend that you implement pixel shaving to remove one pixel from each barcode combination. This will increase the readability of the barcode itself and will not affect its symbology.

Labels and Slip Design Process

1	Design an item label.
2	Design a shipping label.
3	Design a packing slip.
4	Get FBA approval for the functionality of the any new design.

Item Label Specifications

Before you begin labeling your items, use the following guidelines to ensure that your items can be processed in a timely manner after they reach the fulfillment center:

- **Cover all existing barcodes—**Unless you're sending items that are to be commingled, make sure that the existing barcodes on each product are covered.
- **Use the proper print media for your printer—**Each label must be readable and scannable for at least 24 months. This means that the label must not smudge or fade for at least two years. To ensure that your labels last a long time, use inkjet paper and inkjet ink with an inkjet printer. Use laser paper and laser toner with a laser printer.

The label that you place on each item must include the following information:

1. Barcode, with appropriate quiet zone areas
2. String identifier for the barcode (for example, **XXXXX12345**)
3. Title and description
4. Condition (for example, **USED**)
5. Optional merchant-specified information

Note

The merchant-specified information field is reserved for text only. Do not use any additional barcodes. Any additional information you provide may impact the accuracy and speed of inventory tracking.

Item Label Example

Print Specifications

Specification	Description
Label color	White
Font	See Label Font Specifications (p. 60)
Label size	Use a label with a height by width dimension from 1" x 2" up to 2" x 3". For example, you could use a label that is 1" x 3" or 2" x 2", as long as all the required information is included.
Label type	Removable adhesive

Barcode Specifications

Specification	Description
Format	Code 128A
Height	Greater than 0.25" (or 15% of the barcode length)
Narrow barcode element	The specification depends on your printer's resolution. • **For 300 dpi—**13.33 mils • **For 200 dpi—**20 mils
Wide-to-narrow element ratio	3:1
Quiet zone (sides)	Greater than 0.25"
Quiet zone (top and bottom)	Greater than 0.125"

Shipping Label Specifications

The shipping label that you place on the outside of each carton must include the following information:

1. Merchant name
2. Ship to address (the fulfillment center address FBA provided you)
3. Ship from address
4. FBA shipment ID
5. In a section clearly marked **Internal Use Only**, enter the following carton information:
   - Indicate carton quantity for single ASIN cases
   - Indicate "Mixed ASINs" for cartons containing multiple ASINs
   - Label all partial cases as "Partial Case"
6. Note any merchant-specific information below the Internal Use Only section, in a section clearly marked **Merchant Use Only** (not shown in the example below)

Shipping Label Example

Physical Specifications

Specification	Description
Label size	We recommend that you use a label with a height by width dimension of 4" x 6". However you can use a label size up to 6" x 8".
Label style	Permanent
Font	See Label Font Specifications (p. 60)

Barcode Specifications

Specification	Description
Format	Code 128A
Height	Greater than 0.5" (or 15% of the barcode length)
Narrow barcode element	Greater than 0.0075" (7.5 mils)
Wide-to-narrow element ratio	3:1
Quiet zone (sides)	Greater than 0.25"
Quiet zone (top and bottom)	Greater than 0.25"

Packing Slip Specifications

The packing slip that you place on the inside of each carton is divided into five sections:

1. Header
2. Merchant information
3. Merchant checklist
4. Amazon use only
5. Footer

The following lists detail the information required for each section.

Header and Footer Sections

These sections must include this text in bold:

- **Place this packing slip on top of the items in the carton**

Merchant Information Section

This section must include the following information:

- Indication that this is an FBA shipment (display the text **FBA** in a large, bold font)
- Merchant friendly name
- Merchant ship from address

- FBA shipment ID, in both text and scannable barcode

Merchant Checklist Section

This section must include the following information:

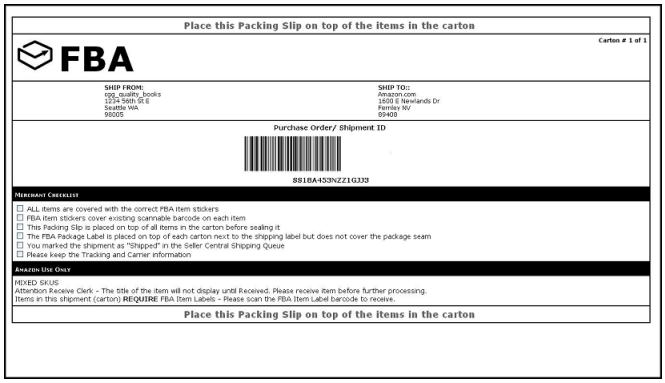
- All items are covered with the correct FBA item stickers
- FBA item stickers cover existing scannable barcode on each item
- This Packing Slip is placed on top of all items in the carton before sealing it
- The FBA Package Label is placed on top of each carton next to the shipping label but does not cover the package seam
- Please keep the Tracking and Carrier information

Amazon Use Only Section

In this section do the following:

- Note whether the carton contains Mixed SKUs or Single SKU
- Note whether the carton contains Used, Used and New, or New items
- Note whether items should be pre-stickered or are commingled

Packing Slip Example



Physical Specifications

Specification	Description
Size	Letter (8.5" x 11")
Font	See Label Font Specifications (p. 60)

Barcode Specifications

Specification	Description
Format	Code 128A
Height	Greater than 0.5" (or 15% of the barcode length)
Narrow barcode element	Greater than 0.0075" (7.5 mils)

Specification	Description
Wide-to-narrow element ratio	3:1
Quiet zone (sides)	Greater than 0.25"
Quiet zone (top and bottom)	Greater than 0.25"

Label and Slip Approval

Once you have designed your labels and packing slip, test them with a scanning device. Verify that they work with your own scanning device before you send them. If they don't work for you, they probably won't work for FBA.

Once you have tested your labels and packing slip, print samples of each and attach them to the Label and Packing Slip Approval Form.

Inbound Pallet Specifications

If you're shipping your items on a pallet, use the following guidelines to ensure that your items can be processed in a timely manner after they reach the fulfillment center:

Physical Pallet Specifications

Specification	Description
Height	Pallet height cannot exceed 50", including the pallet
Overhang	Merchandise must not overhang the pallet edge by more than 1"
Wrapping	Each pallet must be shrink-wrapped in clear plastic
Pallet Type	Pallets must be standard GMA (Grocery Manufacturer's Association), four-way access (48" x 40"), #2 quality or better

Pallet Label Specifications

Specification	Description
Packing Slip	All shipments must contain a packing slip on the outside of the lead pallet
Label	Each pallet must be labelled with **pallet # __of __ pallets** indicator
SKU Indication	Pallets with mixed SKUs must be labeled **MIXED SKU**

Label Font Specifications

The font you use for your labels can help speed up the processing of your inventory at the fulfillment center. Choose a font that clearly distinguishes letters from numbers.

Font Considerations

- Use a font that clearly delineates a zero from the letter "O"—a slashed zero, for example
- Use a font that distinguishes the number one from a lowercase letter "l" or an uppercase letter "i"
- Unless otherwise noted, we recommend that you use the normal font weight (not bold) and that the font size be 8 point

We recommend the fonts listed in the following table.

Recommended Font Types

Font	Operating System	Download
Consolas	Windows	This system font is installed with Windows Vista and Microsoft Office 2007. To download Consolas, go to http://www.microsoft.com/downloads/details.aspx?familyid=22e69ae4-7e40-4807-8a86-b3d36fab68d3&displaylang=en.
Monaco	Mac	This system font is installed with all Mac OS versions.
Andale Mono	Mac, Linux	This system font is installed with the Mac OS at 10.4 and above. For information about downloading Andale Mono to your Linux machine, go to http://corefonts.sourceforge.net/.

Glossary

Access Key ID	Your Access Key ID identifies you as the party responsible for the request. Used for request authentication. See Your AWS Account (p. 13).
Amazon fulfillment center	The warehouse where your inventory is stored. Amazon has many fulfillment centers throughout the country. When you create a shipment, we'll tell you which items need to be in a separate shipment, and we'll tell you for each shipment which fulfillment center to send the items.
Amazon Fulfillment Network	The Amazon Fulfillment Network refers to the infrastructure that Amazon FWS taps into to offer an optimized and cost-effective fulfillment solution for merchants. See Also Fulfillment by Amazon.
Amazon Standard Item Number	The Amazon Standard Item Number (ASIN) is a ten-digit Amazon.com product identifier. Each ASIN is unique. When you submit a SKU, it is linked to an ASIN.
commingled	Your inventory is commingled (also called stickerless) if you send it to the fulfillment center to be combined with inventory that Amazon or other merchants have of the same kind and condition. When someone purchases an item from your inventory, Amazon doesn't physically distinguish your inventory from that of other merchants. However, Amazon will decrement inventory availability for the sale from you, so you always have up-to-date knowledge of your stock. For further information about commingled inventory, go to the Help for Stickerless, Commingled Inventory.

Amazon Fulfillment Network SKU	The Amazon Fulfillment Network SKU (FNSKU) is the unique identifier for each inventory item stored in one of Amazon's fulfillment centers. See Also SKU .
epoch	The date from which time is measured. For most Unix environments the epoch is January 1, 1970, and that is the date used in Amazon FWS.
Fulfillment by Amazon	Fulfillment by Amazon is the general service Amazon provides that allows merchants to share in Amazon's fulfillment network. This general service is offered as a web interface, a text feed, and a web service.
fulfillment identifier	A fulfillment identifier is another term for an Amazon Fulfillment Network SKU (FNSKU).
HMAC-SHA1	A hash message authentication code (HMAC) is a short piece of information used to authenticate a message, calculated using a cryptographic hash function (in this case, the SHA1 function) and a secret key. For more information, go to HMAC: Keyed-Hashing for Message Authentication.
ISBN	The international standard book number is a unique machine-readable, 10- or 13-digit number assigned to each published book.
labeling	In most cases, inventory sent to Amazon for fulfillment must be labeled in advance using a standard bar code identifier that we provide. These labels are how Amazon processes and tracks your inventory in our fulfillment centers. For further help with labeling, see the FBA Merchant Manual.
merchant stock keeping unit	See MSKU.
MSKU	The merchant stock keeping unit (MerchantSKU or MSKU) is your product identifier. See Also SKU .
multi-channel	Multi-channel refers to shipments that are sent at the merchant's request to fulfill orders placed anywhere other than the Amazon.com Marketplace (for example, an order placed on eBay or on your own website).
non-sortable	An item is *non-sortable* if it is oversized. For more help, go to the FBA Guidelines and Frequently Asked Questions page. See Also sortable , oversized .
oversized	The dimensions and weight of the product being fulfilled determine whether the product is oversized or not. Products that are larger than 18"x14"x8" or that weigh 20 pounds or more are considered oversized.
pickable	An inventory item is pickable if it is is in a location that a human being can immediately walk up to it, pick it off the shelf and drop it on a conveyer to be packed and shipped.
quiet zone	The clear area surrounding a bar code.
Secret Access Key	A key assigned to you by Amazon Web Services (AWS) when you sign up for an AWS account. Used for request authentication.

	To learn more about request signatures, including when to use them see Your AWS Account (p. 13).
SKU	A stock keeping unit is a specific merchant's product identifier. The SKU is a critical piece of data in every inventory file that you submit to us. Amazon.com uses the SKUs in your inventory file to associate your products with the appropriate product detail page in our catalog (if one already exists). This means the following: • Every product in your inventory file must have a unique SKU. • An existing SKU cannot be changed; it remains in the catalog until you delete it. That is, you cannot change SKU "123" so that it reads "ABC" in our system simply by changing the SKU in your inventory file.
sortable	Amazon has three classifications for fulfillment items: media, non-media, and oversized. Media items include books, music, software, video games and videos (DVD, VHS). Non-media items include everything else. An item is *sortable* if it is media or non-media. Anything classified as oversized is non-sortable. See Also non-sortable , oversized .
stickered	Your inventory is stickered if you don't want the fulfillment center to combine it with any inventory that Amazon or other merchants have of the same kind and condition. See Also stickerless .
stickerless	Your inventory is stickerless (also called commingled) if you send it to the fulfillment center to be combined with inventory that Amazon or other merchants have of the same kind and condition. When someone purchases an item from your inventory, Amazon doesn't physically distinguish your inventory from that of other merchants. However, Amazon will decrement inventory availability for the sale from you, so you always have up-to-date knowledge of your stock. For further information about stickerless and commingled inventory, go to the Help for Stickerless, Commingled Inventory.
stock keeping unit	See SKU.
supply	Supply is inventory that is available for fulfilling orders. This includes all sellable inventory received by Amazon that is not reserved for existing orders or for internal fulfillment center processes. It also includes inventory expected to be received from inbound shipments.
time point	An element in the inventory API that represents a specific instant in time. A time point is either a specific date and time, or a logical value (e.g., "Immediately").
web service	A web service is an interface to an application.

web services description language	The service's Web Services Description Language file describes the operations along with the format and data types of the actions' requests and responses. Your SOAP or other toolkit interprets the WSDL file to provide your application access to the actions. For most toolkits, your application calls a service action using routines and classes provided or generated by the toolkit.

Index

T

U

W