# Amazon Marketplace Web Service Fulfillment API Sections

## Developer Guide

## Version 2010-06-01-BETA-02

amazon services

# Amazon Marketplace Web Service Fulfillment API Sections: Developer Guide

Copyright © 2010 Amazon.com, Inc. or its affiliates. All rights reserved.

AMAZON and AMAZON.COM are registered trademarks of Amazon.com, Inc. or its affiliates. All other trademarks are the property of their respective owners.

# Table of Contents

Welcome ..... 1
What's New ..... 4
Introduction ..... 5
Using the Amazon MWS Fulfillment API Section ..... 8
    Inbound Use Cases ..... 8
        The Inbound Process ..... 9
        Use Case: Checking the Service Status (Inbound) ..... 12
        Use Case: Planning the Shipment ..... 14
        Use Case: Creating a New Shipment ..... 18
        Use Case: Marking a Shipment as Shipped ..... 20
    Outbound Use Cases ..... 21
        Use Case: Checking the Service Status (Outbound) ..... 22
        Use Case: Creating a Fulfillment Order ..... 24
        Use Case: Getting a Fulfillment Order ..... 26
        Use Case: Listing All Fulfillment Orders ..... 35
        Use Case: Canceling a Fulfillment Order ..... 40
    Inventory Use Cases ..... 41
        Use Case: Checking the Service Status (Inventory) ..... 42
        Use Case: Listing Inventory Supply ..... 44
Making Requests ..... 48
    Understanding Requests and Responses ..... 48
        Requests ..... 48
        Endpoints ..... 49
        Responses ..... 50
    Authentication and Authorization ..... 51
        What Is Authentication? ..... 51
        MWS Credentials ..... 52
        Calculating Signatures ..... 53
        Authorization ..... 56
    Required Headers ..... 57
        X-Amazon-User-Agent Header ..... 58
Appendix ..... 60
    Inbound Label and Packing Slip Specifications ..... 60
        Item Label Specifications ..... 61
        Shipping Label Specifications ..... 62
        Packing Slip Specifications ..... 63
        Label and Slip Approval ..... 65
    Inbound Pallet Specifications ..... 65
    Label Font Specifications ..... 65
Glossary ..... 67
Document Conventions ..... 70
Index ..... 73

# Welcome

**Topics**


- Audience (p. 1)
- Required Knowledge and Skills (p. 1)
- How This Guide Is Organized (p. 2)
- Resources (p. 2)


This is the *Amazon Marketplace Web Service Fulfillment API Sections* Developer Guide. This section describes who should read this guide, how the guide is organized, and other resources related to the Amazon Marketplace Web Service Fulfillment API Sections.

The Amazon Marketplace Web Service Fulfillment API Sections will occasionally be referred to within this guide as simply "Amazon MWS Fulfillment API Section" or "fulfillment"; all copyrights and legal protections still apply.

## Audience

This guide is intended for developers who are building web-enabled applications that 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.

## 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
- Basic knowledge about selling on Amazon. Comprehensive information about selling on Amazon is located at:
  - DE: https://sellercentral.amazon.de/gp/help/home.html
  - FR: https://sellercentral.amazon.fr/gp/help/home.htm

- UK: https://sellercentral.amazon.co.uk/gp/help/
- US: *https://sellercentral.amazon.com/gp/help*

# How This Guide Is Organized

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

| Information | Relevant Sections |
|---|---|
| General information | Introduction (p. 5) |
| Code samples for common use cases | Using the Amazon MWS Fulfillment API Section (p. 8) |
| Information about requests and responses | Understanding Requests and Responses (p. 48) |
| Appendix: Label, Pallet, and Font Specifications | Appendix (p. 60) |
| Glossary | Glossary (p. 67) |
| Typographic and symbol conventions | Document Conventions (p. 70) |
| Index | Index (p. 73) |

# Resources

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

| Resource | Description |
|---|---|
| Amazon Marketplace Web Service Fulfillment API Sections API Reference | A reference for the operations and associated parameters used in the inbound, outbound, and inventory sections. |
| • DE: https://sellercentral.amazon.de/gp/help/<br>• FR: https://sellercentral.amazon.fr/gp/help/<br>• JP: https://sellercentral-japan.amazon.com/gp/help/home.html<br>• UK: https://sellercentral.amazon.co.uk/gp/help/<br>• US: *https://sellercentral.amazon.com/gp/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. |

| Resource | Description |
|---|---|
| • DE: *http://developer.amazonservices.de*<br>• FR: *http://developer.amazonservices.fr*<br>• JP: *http://developer.amazonservices.jp*<br>• UK: *http://developer.amazonservices.co.uk*<br>• US: *http://developer.amazonservices.com* | MWS primary Web page for MWS registration and that contains the MWS documentation, the client libraries, and FAQ. |
| SOA Guide to XML | The Selling on Amazon Guide to XML contains important information about using XML for Amazon product feeds and reports. |
| • DE: *http://www.amazon.de/gp/seller-account/mm-product-page.html?topic=200331350*<br>• FR: *http://www.amazon.fr/gp/seller-account/mm-product-page.html?topic=200328420*<br>• JP: *http://www.amazonservices.jp/jp/services/sell-on-amazon.htm*<br>• UK: *http://services.amazon.co.uk/services/sell-on-amazon/how-it-works/index.html*<br>• US: *http://www.amazonservices.com/content/sell-on-amazon.htm* | Comprehensive information about selling on Amazon. |
| Discussion Forums | Community-based forum for developers to discuss technical questions related to Amazon MWS. |
| Conditions of Use | Detailed information about the copyright and trademark usage at Amazon and other topics. |

# What's New

This What's New is associated with the 2010-06-01-BETA-02 release of the *Amazon Marketplace Web Service Fulfillment API Sections*. This guide was last updated on July 14, 2010.

The following table describes the important changes since the last release of the Amazon Marketplace Web Service Fulfillment API Sections Developer Guide.

| Change | Description | Release Date |
|---|---|---|
| First release | This is the first release of the Amazon Marketplace Web Service Fulfillment API Sections. | 14 July 2010 |

# Introduction

**Topics**


- Overview (p. 5)
- Three Sections (p. 6)
- Features (p. 7)


This introduction is intended to give you a detailed summary of the web service. After reading this section, you should have a good idea of what it 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 the Amazon Fulfillment Network programmatically.

Using these operations, 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 and have Amazon fulfill from both your web site and Amazon.

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 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 Sections

There are three fulfillment sections: inbound, outbound, and inventory. Each section handles different aspects of the fulfillment process.

The Fulfillment Inbound Shipment API Section deals with the first two steps in the fulfillment process: sending products to Amazon and having Amazon store those products.




The Fulfillment Inbound Shipment API Section 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 or you can sell them on another site. If you plan to sell only on Amazon, you only need to use the Fulfillment Inbound Shipment API Section. Amazon handles all the outbound fulfillment for you. However, if you plan to sell on channels besides Amazon, you need both the fulfillment inbound and outbound sections.

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

**Note**

If you only sell on Amazon, you don't need to use the Fulfillment Outbound Shipment API Section. When your items are sold on Amazon, the Amazon Fulfillment Network automatically packs and ships the items to your customers.

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

# Features

The following are 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, you can enable your customers to submit orders any time. There's no lag time while waiting for the seller 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.

# Using the Amazon MWS Fulfillment API Section

**Topics**


- Inbound Use Cases (p. 8)
- Outbound Use Cases (p. 21)
- Inventory Use Cases (p. 41)


This section provides Java code examples for common use cases. The provided sample program snippets are based on libraries. To use these samples, be sure to download the libraries.

For more information about using web services APIs in general, see Making Requests (p. 48).

## Inbound Use Cases

**Topics**


- The Inbound Process (p. 9)
- Use Case: Checking the Service Status (Inbound) (p. 12)
- Use Case: Planning the Shipment (p. 14)
- Use Case: Creating a New Shipment (p. 18)
- Use Case: Marking a Shipment as Shipped (p. 20)


The use cases presented in this section represent the most common requests used with the Fulfillment Inbound Shipment API Section. The use cases are part of the overall inbound process that uses both physical acts and the Fulfillment Inbound Shipment API Section to collect and retrieve information for packaging your shipment to an Amazon fulfillment center. For more information, see The Inbound Process (p. 9).

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

# 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 help for the tasks that do not relate specifically to the operations.

### Tasks to Create an Inbound Shipment

| | |
|---|---|
| 1 | Create a listing. |
| 2 | Plan the shipment and 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*. |

## Creating a Listing

| | |
|---|---|
| Task Type: | Physical |
| Associated Amazon 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 Seller 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 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 Fulfillment Inbound Shipment API Section does not mark the listing on Amazon as Amazon fulfilled. The Fulfillment Inbound Shipment API Section only involves inbound shipments. For more information about marking your listing as Amazon fulfilled, go to the Help for Convert Listings to 'Fulfilled by Amazon'.

## Creating an Inbound Shipment Plan

| | |
|---|---|
| Task Type: | API |
| Associated Use Case: | Use Case: Planning the Shipment (p. 14) |

When you make the operational call to plan a shipment, you're requesting a shipment identifier and destination fulfillment center for the shipment and an Amazon Fulfillment Network SKU (FNSKU) for each supplied seller 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, see Use Case: Planning the Shipment (p. 14).



**Note**

In the `CreateInboundShipmentPlan` response, if the `FulfillmentNetworkSKU` equals the `ASIN`, the item is stickerless (see Preparing the Items for Shipping (p. 10)).

## Preparing the Items for Shipping

| | |
|---|---|
| Task Type: | Physical |
| Associated Amazon 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 `CreateInboundShipmentPlan` operation. After that is complete, create a new shipment. For more information, go to the Help on Packaging Products for Shipping.

## Creating a New Shipment

| | |
|---|---|
| Task Type: | API |
| Associated Use Case: | Use Case: Creating a New Shipment (p. 18) |

When you create a new shipment, you are creating a new shipment using the shipment ID and fulfillment center information received in the `CreateInboundShipmentPlan` response. To see an example, see Use Case: Creating a New Shipment (p. 18).

## Labelling the Shipment

| | |
|---|---|
| Task Type: | Physical |
| Associated Amazon 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. 18) 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.

## Shipping Your Items

| | |
|---|---|
| Task Type: | Physical |
| Associated Amazon Help | Shipping and Routing Requirements |

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

## Marking Your Shipment as Shipped

| | |
|---|---|
| Task Type: | API |
| Associated Use Case: | Use Case: Marking a Shipment as Shipped (p. 20) |

Once you've sent your shipment to an Amazon fulfillment center, use `UpdateInboundShipment` to set the status of your shipment to `Shipped`. To see an example application, see Use Case: Marking a Shipment as Shipped (p. 20).



**Tip**

For information about item label, packing slip, and shipping label specifications, see Inbound Label and Packing Slip Specifications (p. 60)

# 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 operation `GetServiceStatus` using Java.

## Outcome

If the service is running, it returns a success response, along with the date, time, and the service version number. If the service is not running, it returns an error.

## Example Code

### Code Snippet

```
    public static void invokeGetServiceStatus(FBAInboundServiceMWS service,
GetServiceStatusRequest request) {
        try {

            GetServiceStatusResponse response = service.getServiceStatus(re
quest);


            System.out.println ("GetServiceStatus Action Response");
            System.out.println
("=============================================================================");

            System.out.println ();

            System.out.println("    GetServiceStatusResponse");
            System.out.println();
            if (response.isSetGetServiceStatusResult()) {
                System.out.println("        GetServiceStatusResult");
                System.out.println();
                GetServiceStatusResult  getServiceStatusResult = response.get
GetServiceStatusResult();
                if (getServiceStatusResult.isSetServiceStatusInfo()) {
                    System.out.println("            ServiceStatusInfo");
                    System.out.println();
                   System.out.println("              " + getServiceStatusRes
ult.getServiceStatusInfo());
                    System.out.println();
                }
            }
            if (response.isSetResponseMetadata()) {
                System.out.println("        ResponseMetadata");
                System.out.println();
                ResponseMetadata  responseMetadata = response.getResponse
Metadata();
                if (responseMetadata.isSetRequestId()) {
                    System.out.println("            RequestId");
                    System.out.println();
                    System.out.println("                " + response
Metadata.getRequestId());
                    System.out.println();
```

```
                }
            }
            System.out.println();


        } catch (FBAInboundServiceMWSException ex) {

            System.out.println("Caught Exception: " + ex.getMessage());
            System.out.println("Response Status Code: " + ex.getStatusCode());

            System.out.println("Error Code: " + ex.getErrorCode());
            System.out.println("Error Type: " + ex.getErrorType());
            System.out.println("Request ID: " + ex.getRequestId());
            System.out.print("XML: " + ex.getXML());
        }
    }
```



**Note**

Use your Access Key ID for the `accessKeyId` value.
Use your Secret Access Key for the `secretKeyId` value.

# Use Case: Planning the Shipment

## Description

This use case requests a shipment identifier and destination fulfillment center for the shipment and an Amazon Fulfillment Network SKU (FNSKU) for each supplied item. It shows an application of the operation CreateInboundShipmentPlan using Java.

## Outcome

The service returns an aggregate that contains properties and identifiers for a fulfillable item.

## Example Code

### Code Snippet

```
    public static void invokeCreateInboundShipmentPlan(FBAInboundServiceMWS
service, CreateInboundShipmentPlanRequest request) {
        try {

           CreateInboundShipmentPlanResponse response = service.createInbound
ShipmentPlan(request);


            System.out.println ("CreateInboundShipmentPlan Action Response");
            System.out.println
("==============================================================================");

            System.out.println ();

            System.out.println("    CreateInboundShipmentPlanResponse");
            System.out.println();
            if (response.isSetCreateInboundShipmentPlanResult()) {
                System.out.println("        CreateInboundShipmentPlanResult");

                System.out.println();
              CreateInboundShipmentPlanResult  createInboundShipmentPlanResult
 = response.getCreateInboundShipmentPlanResult();
              if (createInboundShipmentPlanResult.isSetInboundShipmentPlans())
 {
                    System.out.println("            InboundShipmentPlans");
                    System.out.println();
                    InboundShipmentPlanList  inboundShipmentPlans = createIn
boundShipmentPlanResult.getInboundShipmentPlans();
                    java.util.List<InboundShipmentPlan> memberList = inbound
ShipmentPlans.getMember();
                    for (InboundShipmentPlan member : memberList) {
                        System.out.println("                member");
                        System.out.println();
                        if (member.isSetShipmentId()) {
                          System.out.println("                    ShipmentId");

                            System.out.println();
                            System.out.println("                        " +
member.getShipmentId());
                            System.out.println();
```

```
                        }
                        if (member.isSetDestinationFulfillmentCenterId()) {
                           System.out.println("                    Destination
FulfillmentCenterId");
                            System.out.println();
                            System.out.println("                        " +
member.getDestinationFulfillmentCenterId());
                            System.out.println();
                        }
                        if (member.isSetShipToAddress()) {
                            System.out.println("                     ShipToAd
dress");
                            System.out.println();
                           Address  shipToAddress = member.getShipToAddress();

                            if (shipToAddress.isSetName()) {
                                System.out.println("
Name");
                                System.out.println();
                                System.out.println("
  " + shipToAddress.getName());
                                System.out.println();
                            }
                            if (shipToAddress.isSetAddressLine1()) {
                                System.out.println("
AddressLine1");
                                System.out.println();
                                System.out.println("
  " + shipToAddress.getAddressLine1());
                                System.out.println();
                            }
                            if (shipToAddress.isSetAddressLine2()) {
                                System.out.println("
AddressLine2");
                                System.out.println();
                                System.out.println("
  " + shipToAddress.getAddressLine2());
                                System.out.println();
                            }
                            if (shipToAddress.isSetDistrictOrCounty()) {
                                System.out.println("
DistrictOrCounty");
                                System.out.println();
                                System.out.println("
  " + shipToAddress.getDistrictOrCounty());
                                System.out.println();
                            }
                            if (shipToAddress.isSetCity()) {
                                System.out.println("
City");
                                System.out.println();
                                System.out.println("
  " + shipToAddress.getCity());
                                System.out.println();
                            }
                            if (shipToAddress.isSetStateOrProvinceCode()) {
                                System.out.println("
StateOrProvinceCode");
                                System.out.println();
```

```
                                System.out.println("
  " + shipToAddress.getStateOrProvinceCode());
                                System.out.println();
                            }
                            if (shipToAddress.isSetCountryCode()) {
                                System.out.println("
CountryCode");
                                System.out.println();
                                System.out.println("
  " + shipToAddress.getCountryCode());
                                System.out.println();
                            }
                            if (shipToAddress.isSetPostalCode()) {
                                System.out.println("
PostalCode");
                                System.out.println();
                                System.out.println("
  " + shipToAddress.getPostalCode());
                                System.out.println();
                            }
                        }
                        if (member.isSetLabelPrepType()) {
                            System.out.println("                    LabelPrep
Type");
                            System.out.println();
                            System.out.println("                        " +
member.getLabelPrepType());
                            System.out.println();
                        }
                        if (member.isSetItems()) {
                            System.out.println("                    Items");
                            System.out.println();
                            InboundShipmentPlanItemList  items = mem
ber.getItems();
                           java.util.List<InboundShipmentPlanItem> member1List
 = items.getMember();
                           for (InboundShipmentPlanItem member1 : member1List)
 {
                                System.out.println("
member");
                                System.out.println();
                                if (member1.isSetSellerSKU()) {
                                    System.out.println("
      SellerSKU");
                                    System.out.println();
                                    System.out.println("
          " + member1.getSellerSKU());
                                    System.out.println();
                                }
                                if (member1.isSetFulfillmentNetworkSKU()) {
                                    System.out.println("
      FulfillmentNetworkSKU");
                                    System.out.println();
                                    System.out.println("
          " + member1.getFulfillmentNetworkSKU());
                                    System.out.println();
                                }
                                if (member1.isSetQuantity()) {
                                    System.out.println("
```

```
      Quantity");
                                    System.out.println();
                                    System.out.println("
          " + member1.getQuantity());
                                    System.out.println();
                                }
                            }
                        }
                    }
                }
              }
              if (response.isSetResponseMetadata()) {
                  System.out.println("        ResponseMetadata");
                  System.out.println();
                  ResponseMetadata  responseMetadata = response.getResponse
Metadata();
                  if (responseMetadata.isSetRequestId()) {
                      System.out.println("            RequestId");
                      System.out.println();
                      System.out.println("                " + response
Metadata.getRequestId());
                      System.out.println();
                  }
              }
              System.out.println();


        } catch (FBAInboundServiceMWSException ex) {

            System.out.println("Caught Exception: " + ex.getMessage());
            System.out.println("Response Status Code: " + ex.getStatusCode());

            System.out.println("Error Code: " + ex.getErrorCode());
            System.out.println("Error Type: " + ex.getErrorType());
            System.out.println("Request ID: " + ex.getRequestId());
            System.out.print("XML: " + ex.getXML());
        }
    }
```



**Note**

Replace the default parameters with your information. For help with these parameters, refer to the Amazon Marketplace Web Service Fulfillment API Sections API Reference.

Set `accessKeyId` to your Access Key ID.

Set `secretAccessKey` to your Secret Access Key.

# Use Case: Creating a New Shipment

## Description

This use case shows a request to create a shipment (or set of shipments) to a fulfillment center using information received from the `CreateInboundShipmentPlan` response. It shows an application of the operation `CreateInboundShipment` using Java.

## Outcome

The service returns a shipment ID.

## Example Code

### Code Snippet

```
   public static void invokeCreateInboundShipment(FBAInboundServiceMWS service,
 CreateInboundShipmentRequest request) {
        try {

           CreateInboundShipmentResponse response = service.createInboundShip
ment(request);


            System.out.println ("CreateInboundShipment Action Response");
            System.out.println
("================================================================================");

            System.out.println ();

            System.out.println("    CreateInboundShipmentResponse");
            System.out.println();
            if (response.isSetCreateInboundShipmentResult()) {
                System.out.println("        CreateInboundShipmentResult");
                System.out.println();
                CreateInboundShipmentResult  createInboundShipmentResult = re
sponse.getCreateInboundShipmentResult();
                if (createInboundShipmentResult.isSetShipmentId()) {
                    System.out.println("            ShipmentId");
                    System.out.println();
                    System.out.println("                " + createInboundShip
mentResult.getShipmentId());
                    System.out.println();
                }
            }
            if (response.isSetResponseMetadata()) {
                System.out.println("        ResponseMetadata");
                System.out.println();
                ResponseMetadata  responseMetadata = response.getResponse
Metadata();
                if (responseMetadata.isSetRequestId()) {
                    System.out.println("            RequestId");
                    System.out.println();
                    System.out.println("                " + response
Metadata.getRequestId());
                    System.out.println();
```

```
                }
            }
            System.out.println();


        } catch (FBAInboundServiceMWSException ex) {

            System.out.println("Caught Exception: " + ex.getMessage());
            System.out.println("Response Status Code: " + ex.getStatusCode());

            System.out.println("Error Code: " + ex.getErrorCode());
            System.out.println("Error Type: " + ex.getErrorType());
            System.out.println("Request ID: " + ex.getRequestId());
            System.out.print("XML: " + ex.getXML());
        }
    }
```



**Note**

Replace the default parameters with your information. For help with these parameters, refer to the Amazon Marketplace Web Service Fulfillment API Sections API Reference.

Set `accessKeyId` to your Access Key ID.

Set `secretAccessKey` to your Secret Access Key.

# Use Case: Marking a Shipment as Shipped

## Description

In this use case, you set the inbound shipment status to `Shipped`. The example shows an application of the operation `UpdateInboundShipment` using Java.

## Outcome

For a successful request, the service returns the shipment ID. Otherwise, the service returns an error.

## Example Code

### Code Snippet

```
   public static void invokeUpdateInboundShipment(FBAInboundServiceMWS service,
 UpdateInboundShipmentRequest request) {
        try {

           UpdateInboundShipmentResponse response = service.updateInboundShip
ment(request);


            System.out.println ("UpdateInboundShipment Action Response");
            System.out.println
("=============================================================================");

            System.out.println ();

            System.out.println("    UpdateInboundShipmentResponse");
            System.out.println();
            if (response.isSetUpdateInboundShipmentResult()) {
                System.out.println("        UpdateInboundShipmentResult");
                System.out.println();
                UpdateInboundShipmentResult  updateInboundShipmentResult = re
sponse.getUpdateInboundShipmentResult();
                if (updateInboundShipmentResult.isSetShipmentId()) {
                    System.out.println("            ShipmentId");
                    System.out.println();
                    System.out.println("                " + updateInboundShip
mentResult.getShipmentId());
                    System.out.println();
                }
            }
            if (response.isSetResponseMetadata()) {
                System.out.println("        ResponseMetadata");
                System.out.println();
                ResponseMetadata  responseMetadata = response.getResponse
Metadata();
                if (responseMetadata.isSetRequestId()) {
                    System.out.println("            RequestId");
                    System.out.println();
                    System.out.println("                " + response
Metadata.getRequestId());
                    System.out.println();
                }
```

```
            }
            System.out.println();


        } catch (FBAInboundServiceMWSException ex) {

            System.out.println("Caught Exception: " + ex.getMessage());
            System.out.println("Response Status Code: " + ex.getStatusCode());

            System.out.println("Error Code: " + ex.getErrorCode());
            System.out.println("Error Type: " + ex.getErrorType());
            System.out.println("Request ID: " + ex.getRequestId());
            System.out.print("XML: " + ex.getXML());
        }
    }
```



**Note**

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

# Outbound Use Cases

**Topics**


- Use Case: Checking the Service Status (Outbound) (p. 22)
- Use Case: Creating a Fulfillment Order (p. 24)
- Use Case: Getting a Fulfillment Order (p. 26)
- Use Case: Listing All Fulfillment Orders (p. 35)
- Use Case: Canceling a Fulfillment Order (p. 40)


The use cases presented in this section represent the most common requests used with the Fulfillment Outbound Shipment API Section. The outbound use cases are straightforward, with a one-to-one correlation between a single 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. To use the snippets, you must incorporate them in your own methods.

# Use Case: Checking the Service Status (Outbound)

## Description

In this use case, you check to see whether the outbound service is running. It shows an application of `GetServiceStatus` using Java.

## Outcome

If the service is running; it returns a success response, along with the date, time, and the service version number. If the service is not running, it returns an error.

## Example Code

### Code Snippet

```
    public static void invokeGetServiceStatus(FBAOutboundServiceMWS service,
GetServiceStatusRequest request) {
        try {

            GetServiceStatusResponse response = service.getServiceStatus(re
quest);


            System.out.println ("GetServiceStatus Action Response");
            System.out.println
("================================================================================");

            System.out.println ();

            System.out.println("    GetServiceStatusResponse");
            System.out.println();
            if (response.isSetGetServiceStatusResult()) {
                System.out.println("        GetServiceStatusResult");
                System.out.println();
                GetServiceStatusResult  getServiceStatusResult = response.get
GetServiceStatusResult();
                if (getServiceStatusResult.isSetServiceStatusInfo()) {
                    System.out.println("            ServiceStatusInfo");
                    System.out.println();
                   System.out.println("              " + getServiceStatusRes
ult.getServiceStatusInfo());
                    System.out.println();
                }
            }
            if (response.isSetResponseMetadata()) {
                System.out.println("        ResponseMetadata");
                System.out.println();
                ResponseMetadata  responseMetadata = response.getResponse
Metadata();
                if (responseMetadata.isSetRequestId()) {
                    System.out.println("            RequestId");
                    System.out.println();
                    System.out.println("                " + response
Metadata.getRequestId());
                    System.out.println();
```

```
                }
            }
            System.out.println();


        } catch (FBAOutboundServiceMWSException ex) {

            System.out.println("Caught Exception: " + ex.getMessage());
            System.out.println("Response Status Code: " + ex.getStatusCode());

            System.out.println("Error Code: " + ex.getErrorCode());
            System.out.println("Error Type: " + ex.getErrorType());
            System.out.println("Request ID: " + ex.getRequestId());
            System.out.print("XML: " + ex.getXML());
        }
    }
```



**Note**

Use your Access Key ID for the `accessKeyId` parameter in the AmazonFBAOutboundClient constructor.

Use your Secret Access Key for the `secretKeyId` parameter in the AmazonFBAOutboundClient constructor.

# Use Case: Creating a Fulfillment Order

## Description

In this use case, you generate a request for Amazon to ship items from the seller's inventory to a destination address. It shows an application of the operation `CreateFulfillmentOrder` using Java.



**Tip**

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

## Outcome

For a successful request, the service returns a RequestID only. Otherwise, the service returns an error.

## Example Code

### Code Snippet

```
    public static void invokeCreateFulfillmentOrder(FBAOutboundServiceMWS ser
vice, CreateFulfillmentOrderRequest request) {
        try {

          CreateFulfillmentOrderResponse response = service.createFulfillment
Order(request);


            System.out.println ("CreateFulfillmentOrder Action Response");
            System.out.println
("=============================================================================");

            System.out.println ();

            System.out.println("    CreateFulfillmentOrderResponse");
            System.out.println();
            if (response.isSetResponseMetadata()) {
                System.out.println("        ResponseMetadata");
                System.out.println();
                ResponseMetadata  responseMetadata = response.getResponse
Metadata();
                if (responseMetadata.isSetRequestId()) {
                    System.out.println("            RequestId");
                    System.out.println();
                    System.out.println("                " + response
Metadata.getRequestId());
                    System.out.println();
                }
            }
            System.out.println();


        } catch (FBAOutboundServiceMWSException ex) {

            System.out.println("Caught Exception: " + ex.getMessage());
            System.out.println("Response Status Code: " + ex.getStatusCode());
```

```
            System.out.println("Error Code: " + ex.getErrorCode());
            System.out.println("Error Type: " + ex.getErrorType());
            System.out.println("Request ID: " + ex.getRequestId());
            System.out.print("XML: " + ex.getXML());
        }
    }
```



### Note

Replace the default parameters with information from your own order. For help with these parameters, see the `CreateFulfillmentOrder` operation in the Amazon Marketplace Web Service Fulfillment API Sections API Reference.

Use your Access Key ID for the `accessKeyId` parameter in the AmazonFBAOutboundClient constructor.

Use your Secret Access Key for the `secretKeyId` parameter in the AmazonFBAOutboundClient constructor.

# Use Case: Getting a Fulfillment Order

## Description

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

## Outcome

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

## Example Code

### Code Snippet

```
   public static void invokeGetFulfillmentOrder(FBAOutboundServiceMWS service,
 GetFulfillmentOrderRequest request) {
        try {

          GetFulfillmentOrderResponse response = service.getFulfillmentOrder(re
quest);


            System.out.println ("GetFulfillmentOrder Action Response");
            System.out.println
("==============================================================================");

            System.out.println ();

            System.out.println("    GetFulfillmentOrderResponse");
            System.out.println();
            if (response.isSetGetFulfillmentOrderResult()) {
                System.out.println("        GetFulfillmentOrderResult");
                System.out.println();
                GetFulfillmentOrderResult  getFulfillmentOrderResult = re
sponse.getGetFulfillmentOrderResult();
                if (getFulfillmentOrderResult.isSetFulfillmentOrder()) {
                    System.out.println("            FulfillmentOrder");
                    System.out.println();
                  FulfillmentOrder  fulfillmentOrder = getFulfillmentOrderRes
ult.getFulfillmentOrder();
                    if (fulfillmentOrder.isSetSellerFulfillmentOrderId()) {
                        System.out.println("                SellerFulfillment
OrderId");
                        System.out.println();
                       System.out.println("                    " + fulfillment
Order.getSellerFulfillmentOrderId());
                        System.out.println();
                    }
                    if (fulfillmentOrder.isSetDisplayableOrderId()) {
                      System.out.println("              DisplayableOrderId");

                        System.out.println();
                       System.out.println("                    " + fulfillment
```

```
Order.getDisplayableOrderId());
                        System.out.println();
                    }
                    if (fulfillmentOrder.isSetDisplayableOrderDateTime()) {
                        System.out.println("               DisplayableOrderD
ateTime");
                        System.out.println();
                       System.out.println("                  " + fulfillment
Order.getDisplayableOrderDateTime());
                        System.out.println();
                    }
                    if (fulfillmentOrder.isSetDisplayableOrderComment()) {
                       System.out.println("              DisplayableOrderCom
ment");
                        System.out.println();
                       System.out.println("                  " + fulfillment
Order.getDisplayableOrderComment());
                        System.out.println();
                    }
                    if (fulfillmentOrder.isSetShippingSpeedCategory()) {
                        System.out.println("               ShippingSpeedCat
egory");
                        System.out.println();
                       System.out.println("                  " + fulfillment
Order.getShippingSpeedCategory());
                        System.out.println();
                    }
                    if (fulfillmentOrder.isSetDestinationAddress()) {
                      System.out.println("              DestinationAddress");

                        System.out.println();
                        Address  destinationAddress = fulfillmentOrder.getDes
tinationAddress();
                        if (destinationAddress.isSetName()) {
                            System.out.println("                    Name");
                            System.out.println();
                            System.out.println("                        " +
destinationAddress.getName());
                            System.out.println();
                        }
                        if (destinationAddress.isSetLine1()) {
                            System.out.println("                    Line1");
                            System.out.println();
                            System.out.println("                        " +
destinationAddress.getLine1());
                            System.out.println();
                        }
                        if (destinationAddress.isSetLine2()) {
                            System.out.println("                    Line2");
                            System.out.println();
                            System.out.println("                        " +
destinationAddress.getLine2());
                            System.out.println();
                        }
                        if (destinationAddress.isSetLine3()) {
                            System.out.println("                    Line3");
                            System.out.println();
                            System.out.println("                        " +
destinationAddress.getLine3());
```

```
                            System.out.println();
                        }
                        if (destinationAddress.isSetDistrictOrCounty()) {
                            System.out.println("                    DistrictOr
County");
                            System.out.println();
                            System.out.println("                        " +
destinationAddress.getDistrictOrCounty());
                            System.out.println();
                        }
                        if (destinationAddress.isSetCity()) {
                            System.out.println("                     City");
                            System.out.println();
                            System.out.println("                         " +
destinationAddress.getCity());
                            System.out.println();
                        }
                        if (destinationAddress.isSetStateOrProvinceCode()) {
                            System.out.println("                     StateOr
ProvinceCode");
                            System.out.println();
                            System.out.println("                         " +
destinationAddress.getStateOrProvinceCode());
                            System.out.println();
                        }
                        if (destinationAddress.isSetCountryCode()) {
                            System.out.println("                     Country
Code");
                            System.out.println();
                            System.out.println("                         " +
destinationAddress.getCountryCode());
                            System.out.println();
                        }
                        if (destinationAddress.isSetPostalCode()) {
                          System.out.println("                   PostalCode");

                            System.out.println();
                            System.out.println("                         " +
destinationAddress.getPostalCode());
                            System.out.println();
                        }
                        if (destinationAddress.isSetPhoneNumber()) {
                            System.out.println("                     PhoneNum
ber");
                            System.out.println();
                            System.out.println("                         " +
destinationAddress.getPhoneNumber());
                            System.out.println();
                        }
                    }
                    if (fulfillmentOrder.isSetFulfillmentPolicy()) {
                       System.out.println("               FulfillmentPolicy");

                        System.out.println();
                       System.out.println("                    " + fulfillment
Order.getFulfillmentPolicy());
                        System.out.println();
                    }
                    if (fulfillmentOrder.isSetFulfillmentMethod()) {
```

```
                        System.out.println("               FulfillmentMethod");

                         System.out.println();
                        System.out.println("                   " + fulfillment
Order.getFulfillmentMethod());
                         System.out.println();
                    }
                    if (fulfillmentOrder.isSetReceivedDateTime()) {
                        System.out.println("              ReceivedDateTime");

                         System.out.println();
                        System.out.println("                   " + fulfillment
Order.getReceivedDateTime());
                         System.out.println();
                    }
                    if (fulfillmentOrder.isSetFulfillmentOrderStatus()) {
                        System.out.println("               FulfillmentOrder
Status");
                         System.out.println();
                        System.out.println("                   " + fulfillment
Order.getFulfillmentOrderStatus());
                         System.out.println();
                    }
                    if (fulfillmentOrder.isSetStatusUpdatedDateTime()) {
                        System.out.println("               StatusUpdatedDate
Time");
                         System.out.println();
                        System.out.println("                   " + fulfillment
Order.getStatusUpdatedDateTime());
                         System.out.println();
                    }
                    if (fulfillmentOrder.isSetNotificationEmailList()) {
                        System.out.println("               NotificationEmail
List");
                         System.out.println();
                        NotificationEmailList  notificationEmailList = fulfill
mentOrder.getNotificationEmailList();
                      java.util.List<String> memberList  =  notificationEmail
List.getMember();
                         for (String member : memberList) {
                             System.out.println("                   member");
                                 System.out.println();
                             System.out.println("                       " +
member);
                         }
                    }
                }
                if (getFulfillmentOrderResult.isSetFulfillmentOrderItem()) {
                    System.out.println("           FulfillmentOrderItem");
                    System.out.println();
                  FulfillmentOrderItemList  fulfillmentOrderItem = getFulfill
mentOrderResult.getFulfillmentOrderItem();
                   java.util.List<FulfillmentOrderItem> member1List = fulfill
mentOrderItem.getMember();
                    for (FulfillmentOrderItem member1 : member1List) {
                        System.out.println("               member");
                        System.out.println();
                        if (member1.isSetSellerSKU()) {
                           System.out.println("                  SellerSKU");
```

```
                            System.out.println();
                            System.out.println("                        " +
member1.getSellerSKU());
                            System.out.println();
                        }
                        if (member1.isSetSellerFulfillmentOrderItemId()) {
                            System.out.println("                    SellerFul
fillmentOrderItemId");
                            System.out.println();
                            System.out.println("                        " +
member1.getSellerFulfillmentOrderItemId());
                            System.out.println();
                        }
                        if (member1.isSetQuantity()) {
                           System.out.println("                    Quantity");

                            System.out.println();
                            System.out.println("                        " +
member1.getQuantity());
                            System.out.println();
                        }
                        if (member1.isSetGiftMessage()) {
                            System.out.println("                     GiftMes
sage");
                            System.out.println();
                            System.out.println("                        " +
member1.getGiftMessage());
                            System.out.println();
                        }
                        if (member1.isSetDisplayableComment()) {
                           System.out.println("                    Displayable
Comment");
                            System.out.println();
                            System.out.println("                        " +
member1.getDisplayableComment());
                            System.out.println();
                        }
                        if (member1.isSetFulfillmentNetworkSKU()) {
                           System.out.println("                    Fulfillment
NetworkSKU");
                            System.out.println();
                            System.out.println("                        " +
member1.getFulfillmentNetworkSKU());
                            System.out.println();
                        }
                        if (member1.isSetOrderItemDisposition()) {
                            System.out.println("                     OrderItem
Disposition");
                            System.out.println();
                            System.out.println("                        " +
member1.getOrderItemDisposition());
                            System.out.println();
                        }
                        if (member1.isSetCancelledQuantity()) {
                            System.out.println("                     Can
celledQuantity");
                            System.out.println();
                            System.out.println("                        " +
```

```
member1.getCancelledQuantity());
                            System.out.println();
                        }
                        if (member1.isSetUnfulfillableQuantity()) {
                            System.out.println("                    Unfulfil
lableQuantity");
                            System.out.println();
                            System.out.println("                        " +
member1.getUnfulfillableQuantity());
                            System.out.println();
                        }
                        if (member1.isSetEstimatedShipDateTime()) {
                            System.out.println("                    Estimated
ShipDateTime");
                            System.out.println();
                            System.out.println("                        " +
member1.getEstimatedShipDateTime());
                            System.out.println();
                        }
                        if (member1.isSetEstimatedArrivalDateTime()) {
                           System.out.println("                   EstimatedAr
rivalDateTime");
                            System.out.println();
                            System.out.println("                        " +
member1.getEstimatedArrivalDateTime());
                            System.out.println();
                        }
                        if (member1.isSetPerUnitDeclaredValue()) {
                            System.out.println("                    PerUnitDe
claredValue");
                            System.out.println();
                          Currency  perUnitDeclaredValue = member1.getPerUnit
DeclaredValue();
                            if (perUnitDeclaredValue.isSetCurrencyCode()) {
                                System.out.println("
CurrencyCode");
                                System.out.println();
                                System.out.println("
  " + perUnitDeclaredValue.getCurrencyCode());
                                System.out.println();
                            }
                            if (perUnitDeclaredValue.isSetValue()) {
                                System.out.println("
Value");
                                System.out.println();
                                System.out.println("
  " + perUnitDeclaredValue.getValue());
                                System.out.println();
                            }
                        }
                    }
                }
                if (getFulfillmentOrderResult.isSetFulfillmentShipment()) {
                    System.out.println("            FulfillmentShipment");
                    System.out.println();
                    FulfillmentShipmentList  fulfillmentShipment = getFulfill
mentOrderResult.getFulfillmentShipment();
                    java.util.List<FulfillmentShipment> member2List = fulfill
mentShipment.getMember();
```

```
                    for (FulfillmentShipment member2 : member2List) {
                        System.out.println("               member");
                        System.out.println();
                        if (member2.isSetAmazonShipmentId()) {
                           System.out.println("                   AmazonShip
mentId");
                            System.out.println();
                            System.out.println("                        " +
member2.getAmazonShipmentId());
                            System.out.println();
                        }
                        if (member2.isSetFulfillmentCenterId()) {
                           System.out.println("                   Fulfillment
CenterId");
                            System.out.println();
                            System.out.println("                        " +
member2.getFulfillmentCenterId());
                            System.out.println();
                        }
                        if (member2.isSetFulfillmentShipmentStatus()) {
                           System.out.println("                   Fulfillment
ShipmentStatus");
                            System.out.println();
                            System.out.println("                        " +
member2.getFulfillmentShipmentStatus());
                            System.out.println();
                        }
                        if (member2.isSetShippingDateTime()) {
                           System.out.println("                   ShippingDat
eTime");
                            System.out.println();
                            System.out.println("                        " +
member2.getShippingDateTime());
                            System.out.println();
                        }
                        if (member2.isSetEstimatedArrivalDateTime()) {
                           System.out.println("                   EstimatedAr
rivalDateTime");
                            System.out.println();
                            System.out.println("                        " +
member2.getEstimatedArrivalDateTime());
                            System.out.println();
                        }
                        if (member2.isSetFulfillmentShipmentItem()) {
                           System.out.println("                   Fulfillment
ShipmentItem");
                            System.out.println();
                          FulfillmentShipmentItemList  fulfillmentShipmentItem
 = member2.getFulfillmentShipmentItem();
                           java.util.List<FulfillmentShipmentItem> member3List
 = fulfillmentShipmentItem.getMember();
                           for (FulfillmentShipmentItem member3 : member3List)
 {
                                System.out.println("
member");
                                System.out.println();
                                if (member3.isSetSellerSKU()) {
                                    System.out.println("
      SellerSKU");
```

```
                                    System.out.println();
                                    System.out.println("
            " + member3.getSellerSKU());
                                    System.out.println();
                                }
                               if (member3.isSetSellerFulfillmentOrderItemId())
 {
                                    System.out.println("
      SellerFulfillmentOrderItemId");
                                    System.out.println();
                                    System.out.println("
            " + member3.getSellerFulfillmentOrderItemId());
                                    System.out.println();
                                }
                                if (member3.isSetQuantity()) {
                                    System.out.println("
      Quantity");
                                    System.out.println();
                                    System.out.println("
            " + member3.getQuantity());
                                    System.out.println();
                                }
                                if (member3.isSetPackageNumber()) {
                                    System.out.println("
      PackageNumber");
                                    System.out.println();
                                    System.out.println("
            " + member3.getPackageNumber());
                                    System.out.println();
                                }
                            }
                        }
                        if (member2.isSetFulfillmentShipmentPackage()) {
                           System.out.println("                    Fulfillment
ShipmentPackage");
                            System.out.println();
                          FulfillmentShipmentPackageList  fulfillmentShipment
Package = member2.getFulfillmentShipmentPackage();
                            java.util.List<FulfillmentShipmentPackage> mem
ber4List = fulfillmentShipmentPackage.getMember();
                            for (FulfillmentShipmentPackage member4 : mem
ber4List) {
                                System.out.println("
member");
                                System.out.println();
                                if (member4.isSetPackageNumber()) {
                                    System.out.println("
      PackageNumber");
                                    System.out.println();
                                    System.out.println("
            " + member4.getPackageNumber());
                                    System.out.println();
                                }
                                if (member4.isSetCarrierCode()) {
                                    System.out.println("
      CarrierCode");
                                    System.out.println();
                                    System.out.println("
            " + member4.getCarrierCode());
```

```
                                    System.out.println();
                                }
                                if (member4.isSetTrackingNumber()) {
                                    System.out.println("
        TrackingNumber");
                                    System.out.println();
                                    System.out.println("
            " + member4.getTrackingNumber());
                                    System.out.println();
                                }
                                if (member4.isSetEstimatedArrivalDateTime()) {

                                    System.out.println("
        EstimatedArrivalDateTime");
                                    System.out.println();
                                    System.out.println("
            " + member4.getEstimatedArrivalDateTime());
                                    System.out.println();
                                }
                            }
                        }
                    }
                }
                if (response.isSetResponseMetadata()) {
                    System.out.println("        ResponseMetadata");
                    System.out.println();
                    ResponseMetadata  responseMetadata = response.getResponse
Metadata();
                    if (responseMetadata.isSetRequestId()) {
                        System.out.println("            RequestId");
                        System.out.println();
                        System.out.println("                " + response
Metadata.getRequestId());
                        System.out.println();
                    }
                }
                System.out.println();


            } catch (FBAOutboundServiceMWSException ex) {

                System.out.println("Caught Exception: " + ex.getMessage());
                System.out.println("Response Status Code: " + ex.getStatusCode());

                System.out.println("Error Code: " + ex.getErrorCode());
                System.out.println("Error Type: " + ex.getErrorType());
                System.out.println("Request ID: " + ex.getRequestId());
                System.out.print("XML: " + ex.getXML());
            }
        }
```

# 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 operations `ListAllFulfillmentOrders` and `ListAllFulfillmentOrdersByNextToken` using Java.

## Outcome

The service returns all relevant shipping information, including the original fulfillment order request and the status of the order in the Amazon Fulfillment Network.

## Example Code

### Code Snippet

```
    public static void invokeListAllFulfillmentOrders(FBAOutboundServiceMWS
service, ListAllFulfillmentOrdersRequest request) {
        try {

           ListAllFulfillmentOrdersResponse response = service.listAllFulfill
mentOrders(request);


            System.out.println ("ListAllFulfillmentOrders Action Response");
            System.out.println
("==============================================================================");

            System.out.println ();

            System.out.println("    ListAllFulfillmentOrdersResponse");
            System.out.println();
            if (response.isSetListAllFulfillmentOrdersResult()) {
                System.out.println("        ListAllFulfillmentOrdersResult");
                System.out.println();
                ListAllFulfillmentOrdersResult  listAllFulfillmentOrdersResult
 = response.getListAllFulfillmentOrdersResult();
                if (listAllFulfillmentOrdersResult.isSetNextToken()) {
                    System.out.println("            NextToken");
                    System.out.println();
                   System.out.println("               " + listAllFulfillment
OrdersResult.getNextToken());
                    System.out.println();
                }
                if (listAllFulfillmentOrdersResult.isSetFulfillmentOrders()) {

                    System.out.println("            FulfillmentOrders");
                    System.out.println();
                    FulfillmentOrderList  fulfillmentOrders = listAllFulfill
mentOrdersResult.getFulfillmentOrders();
                    java.util.List<FulfillmentOrder> memberList = fulfillment
Orders.getMember();
                    for (FulfillmentOrder member : memberList) {
                        System.out.println("                member");
                        System.out.println();
```

```
                        if (member.isSetSellerFulfillmentOrderId()) {
                            System.out.println("                    SellerFul
fillmentOrderId");
                            System.out.println();
                            System.out.println("                        " +
member.getSellerFulfillmentOrderId());
                            System.out.println();
                        }
                        if (member.isSetDisplayableOrderId()) {
                           System.out.println("                    Displayable
OrderId");
                            System.out.println();
                            System.out.println("                        " +
member.getDisplayableOrderId());
                            System.out.println();
                        }
                        if (member.isSetDisplayableOrderDateTime()) {
                           System.out.println("                    Displayable
OrderDateTime");
                            System.out.println();
                            System.out.println("                        " +
member.getDisplayableOrderDateTime());
                            System.out.println();
                        }
                        if (member.isSetDisplayableOrderComment()) {
                           System.out.println("                    Displayable
OrderComment");
                            System.out.println();
                            System.out.println("                        " +
member.getDisplayableOrderComment());
                            System.out.println();
                        }
                        if (member.isSetShippingSpeedCategory()) {
                            System.out.println("                      Shipping
SpeedCategory");
                            System.out.println();
                            System.out.println("                        " +
member.getShippingSpeedCategory());
                            System.out.println();
                        }
                        if (member.isSetDestinationAddress()) {
                           System.out.println("                    Destination
Address");
                            System.out.println();
                          Address  destinationAddress = member.getDestination
Address();
                            if (destinationAddress.isSetName()) {
                                System.out.println("
Name");
                                System.out.println();
                                System.out.println("
  " + destinationAddress.getName());
                                System.out.println();
                            }
                            if (destinationAddress.isSetLine1()) {
                                System.out.println("
Line1");
                                System.out.println();
                                System.out.println("
```

```
  " + destinationAddress.getLine1());
                                System.out.println();
                            }
                            if (destinationAddress.isSetLine2()) {
                                System.out.println("
Line2");
                                System.out.println();
                                System.out.println("
  " + destinationAddress.getLine2());
                                System.out.println();
                            }
                            if (destinationAddress.isSetLine3()) {
                                System.out.println("
Line3");
                                System.out.println();
                                System.out.println("
  " + destinationAddress.getLine3());
                                System.out.println();
                            }
                            if (destinationAddress.isSetDistrictOrCounty()) {
                                System.out.println("
DistrictOrCounty");
                                System.out.println();
                                System.out.println("
  " + destinationAddress.getDistrictOrCounty());
                                System.out.println();
                            }
                            if (destinationAddress.isSetCity()) {
                                System.out.println("
City");
                                System.out.println();
                                System.out.println("
  " + destinationAddress.getCity());
                                System.out.println();
                            }
                            if (destinationAddress.isSetStateOrProvinceCode())
 {
                                System.out.println("
StateOrProvinceCode");
                                System.out.println();
                                System.out.println("
  " + destinationAddress.getStateOrProvinceCode());
                                System.out.println();
                            }
                            if (destinationAddress.isSetCountryCode()) {
                                System.out.println("
CountryCode");
                                System.out.println();
                                System.out.println("
  " + destinationAddress.getCountryCode());
                                System.out.println();
                            }
                            if (destinationAddress.isSetPostalCode()) {
                                System.out.println("
PostalCode");
                                System.out.println();
                                System.out.println("
  " + destinationAddress.getPostalCode());
                                System.out.println();
```

```
                            }
                            if (destinationAddress.isSetPhoneNumber()) {
                                System.out.println("
PhoneNumber");
                                System.out.println();
                                System.out.println("
  " + destinationAddress.getPhoneNumber());
                                System.out.println();
                            }
                        }
                        if (member.isSetFulfillmentPolicy()) {
                           System.out.println("                    Fulfillment
Policy");
                            System.out.println();
                            System.out.println("                         " +
member.getFulfillmentPolicy());
                            System.out.println();
                        }
                        if (member.isSetFulfillmentMethod()) {
                           System.out.println("                    Fulfillment
Method");
                            System.out.println();
                            System.out.println("                         " +
member.getFulfillmentMethod());
                            System.out.println();
                        }
                        if (member.isSetReceivedDateTime()) {
                           System.out.println("                    ReceivedDat
eTime");
                            System.out.println();
                            System.out.println("                         " +
member.getReceivedDateTime());
                            System.out.println();
                        }
                        if (member.isSetFulfillmentOrderStatus()) {
                           System.out.println("                    Fulfillment
OrderStatus");
                            System.out.println();
                            System.out.println("                         " +
member.getFulfillmentOrderStatus());
                            System.out.println();
                        }
                        if (member.isSetStatusUpdatedDateTime()) {
                            System.out.println("                      StatusUp
datedDateTime");
                            System.out.println();
                            System.out.println("                         " +
member.getStatusUpdatedDateTime());
                            System.out.println();
                        }
                        if (member.isSetNotificationEmailList()) {
                            System.out.println("                      Notific
ationEmailList");
                            System.out.println();
                            NotificationEmailList  notificationEmailList =
member.getNotificationEmailList();
                          java.util.List<String> member1List  =  notificationE
mailList.getMember();
                            for (String member1 : member1List) {
```

```
                                System.out.println("
member");
                                    System.out.println();
                                System.out.println("
  " + member1);
                            }
                        }
                    }
                }
            }
            if (response.isSetResponseMetadata()) {
                System.out.println("        ResponseMetadata");
                System.out.println();
                ResponseMetadata  responseMetadata = response.getResponse
Metadata();
                if (responseMetadata.isSetRequestId()) {
                    System.out.println("            RequestId");
                    System.out.println();
                    System.out.println("                " + response
Metadata.getRequestId());
                    System.out.println();
                }
            }
            System.out.println();


        } catch (FBAOutboundServiceMWSException ex) {

            System.out.println("Caught Exception: " + ex.getMessage());
            System.out.println("Response Status Code: " + ex.getStatusCode());

            System.out.println("Error Code: " + ex.getErrorCode());
            System.out.println("Error Type: " + ex.getErrorType());
            System.out.println("Request ID: " + ex.getRequestId());
            System.out.print("XML: " + ex.getXML());
        }
    }
```



**Note**

Use your Access Key ID for the `accessKeyId` parameter in the AmazonFBAOutboundClient constructor.

Use your Secret Access Key for the `secretKeyId` parameter in the AmazonFBAOutboundClient constructor.

# 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 operation `CancelFulfillmentOrder` using Java.

## Outcome

For a successful request, the service returns a RequestID only. Otherwise, the service returns an error.

## Example Code

### Code Snippet

```
    public static void invokeCancelFulfillmentOrder(FBAOutboundServiceMWS ser
vice, CancelFulfillmentOrderRequest request) {
        try {

          CancelFulfillmentOrderResponse response = service.cancelFulfillment
Order(request);


            System.out.println ("CancelFulfillmentOrder Action Response");
            System.out.println
("==============================================================================");

            System.out.println ();

            System.out.println("    CancelFulfillmentOrderResponse");
            System.out.println();
            if (response.isSetResponseMetadata()) {
                System.out.println("        ResponseMetadata");
                System.out.println();
                ResponseMetadata  responseMetadata = response.getResponse
Metadata();
                if (responseMetadata.isSetRequestId()) {
                    System.out.println("            RequestId");
                    System.out.println();
                    System.out.println("                " + response
Metadata.getRequestId());
                    System.out.println();
                }
            }
            System.out.println();


        } catch (FBAOutboundServiceMWSException ex) {

            System.out.println("Caught Exception: " + ex.getMessage());
            System.out.println("Response Status Code: " + ex.getStatusCode());

            System.out.println("Error Code: " + ex.getErrorCode());
            System.out.println("Error Type: " + ex.getErrorType());
            System.out.println("Request ID: " + ex.getRequestId());
            System.out.print("XML: " + ex.getXML());
```

```
        }
    }
```

# Inventory Use Cases

**Topics**


- Use Case: Checking the Service Status (Inventory) (p. 42)
- Use Case: Listing Inventory Supply (p. 44)


The use cases presented in this section represent the most common requests used with the Fulfillment Inventory API Section.

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

# Use Case: Checking the Service Status (Inventory)

## Description

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

## Outcome

If the service is running, it returns a success response, along with the date, time, and the service version number. If the service is not running, it returns an error.

## Example Code

### Code Snippet

```
    public static void invokeGetServiceStatus(FBAInventoryServiceMWS service,
GetServiceStatusRequest request) {
        try {

            GetServiceStatusResponse response = service.getServiceStatus(re
quest);


            System.out.println ("GetServiceStatus Action Response");
            System.out.println
("==========================================================================");

            System.out.println ();

            System.out.println("    GetServiceStatusResponse");
            System.out.println();
            if (response.isSetGetServiceStatusResult()) {
                System.out.println("        GetServiceStatusResult");
                System.out.println();
                GetServiceStatusResult  getServiceStatusResult = response.get
GetServiceStatusResult();
                if (getServiceStatusResult.isSetServiceStatusInfo()) {
                    System.out.println("            ServiceStatusInfo");
                    System.out.println();
                   System.out.println("              " + getServiceStatusRes
ult.getServiceStatusInfo());
                    System.out.println();
                }
            }
            if (response.isSetResponseMetadata()) {
                System.out.println("        ResponseMetadata");
                System.out.println();
                ResponseMetadata  responseMetadata = response.getResponse
Metadata();
                if (responseMetadata.isSetRequestId()) {
                    System.out.println("            RequestId");
                    System.out.println();
                    System.out.println("                " + response
Metadata.getRequestId());
                    System.out.println();
```

```
                }
            }
            System.out.println();


        } catch (FBAInventoryServiceMWSException ex) {

            System.out.println("Caught Exception: " + ex.getMessage());
            System.out.println("Response Status Code: " + ex.getStatusCode());

            System.out.println("Error Code: " + ex.getErrorCode());
            System.out.println("Error Type: " + ex.getErrorType());
            System.out.println("Request ID: " + ex.getRequestId());
            System.out.print("XML: " + ex.getXML());
        }
    }
```



**Note**

Use your Access Key ID for the `accessKeyId` value.
Use your Secret Access Key for the `secretKeyId` value.

# Use Case: Listing Inventory Supply

## Description

This use case requests a list of inventory supply. It shows an application of the operations `ListInventorySupply` and `ListInventorySupplyByNextToken` using Java.

## Outcome

The service returns all relevant inventory information, including identifiers and quantity information.

## Example Code

### Code Snippet

```
   public static void invokeListInventorySupply(FBAInventoryServiceMWS service,
 ListInventorySupplyRequest request) {
        try {

            ListInventorySupplyResponse response = service.listInventorySup
ply(request);


            System.out.println ("ListInventorySupply Action Response");
            System.out.println
("==============================================================================");

            System.out.println ();

            System.out.println("    ListInventorySupplyResponse");
            System.out.println();
            if (response.isSetListInventorySupplyResult()) {
                System.out.println("        ListInventorySupplyResult");
                System.out.println();
                ListInventorySupplyResult  listInventorySupplyResult = re
sponse.getListInventorySupplyResult();
                if (listInventorySupplyResult.isSetInventorySupplyList()) {
                    System.out.println("            InventorySupplyList");
                    System.out.println();
                  InventorySupplyList  inventorySupplyList = listInventorySup
plyResult.getInventorySupplyList();
                    java.util.List<InventorySupply> memberList = inventorySup
plyList.getMember();
                    for (InventorySupply member : memberList) {
                        System.out.println("                member");
                        System.out.println();
                        if (member.isSetSellerSKU()) {
                           System.out.println("                    SellerSKU");

                            System.out.println();
                            System.out.println("                        " +
member.getSellerSKU());
                            System.out.println();
                        }
                        if (member.isSetFNSKU()) {
                            System.out.println("                    FNSKU");
```

```
                            System.out.println();
                            System.out.println("                          " +
member.getFNSKU());
                            System.out.println();
                        }
                        if (member.isSetASIN()) {
                            System.out.println("                    ASIN");
                            System.out.println();
                            System.out.println("                        " +
member.getASIN());
                            System.out.println();
                        }
                        if (member.isSetCondition()) {
                           System.out.println("                  Condition");

                            System.out.println();
                            System.out.println("                        " +
member.getCondition());
                            System.out.println();
                        }
                        if (member.isSetTotalSupplyQuantity()) {
                            System.out.println("                     TotalSup
plyQuantity");
                            System.out.println();
                            System.out.println("                        " +
member.getTotalSupplyQuantity());
                            System.out.println();
                        }
                        if (member.isSetInStockSupplyQuantity()) {
                           System.out.println("                   InStockSup
plyQuantity");
                            System.out.println();
                            System.out.println("                        " +
member.getInStockSupplyQuantity());
                            System.out.println();
                        }
                        if (member.isSetEarliestAvailability()) {
                          System.out.println("                 EarliestAvail
ability");
                            System.out.println();
                            Timepoint  earliestAvailability = member.getEarli
estAvailability();
                            if (earliestAvailability.isSetTimepointType()) {
                                System.out.println("
TimepointType");
                                System.out.println();
                                System.out.println("
  " + earliestAvailability.getTimepointType());
                                System.out.println();
                            }
                            if (earliestAvailability.isSetDateTime()) {
                                System.out.println("
DateTime");
                                System.out.println();
                                System.out.println("
  " + earliestAvailability.getDateTime());
                                System.out.println();
                            }
                        }
```

```
                        if (member.isSetSupplyDetail()) {
                            System.out.println("                    SupplyDe
tail");
                            System.out.println();
                         InventorySupplyDetailList  supplyDetail = member.get
SupplyDetail();
                            java.util.List<InventorySupplyDetail> member1List
= supplyDetail.getMember();
                            for (InventorySupplyDetail member1 : member1List)
{
                                System.out.println("
member");
                                System.out.println();
                                if (member1.isSetQuantity()) {
                                    System.out.println("
      Quantity");
                                    System.out.println();
                                    System.out.println("
          " + member1.getQuantity());
                                    System.out.println();
                                }
                                if (member1.isSetSupplyType()) {
                                    System.out.println("
      SupplyType");
                                    System.out.println();
                                    System.out.println("
          " + member1.getSupplyType());
                                    System.out.println();
                                }
                                if (member1.isSetEarliestAvailableToPick()) {
                                    System.out.println("
      EarliestAvailableToPick");
                                    System.out.println();
                                    Timepoint  earliestAvailableToPick = mem
ber1.getEarliestAvailableToPick();
                                  if (earliestAvailableToPick.isSetTimepoint
Type()) {
                                      System.out.println("
              TimepointType");
                                      System.out.println();
                                      System.out.println("
                  " + earliestAvailableToPick.getTimepointType());
                                      System.out.println();
                                  }
                                if (earliestAvailableToPick.isSetDateTime())
 {
                                      System.out.println("
              DateTime");
                                      System.out.println();
                                      System.out.println("
                  " + earliestAvailableToPick.getDateTime());
                                      System.out.println();
                                  }
                                }
                                if (member1.isSetLatestAvailableToPick()) {
                                    System.out.println("
      LatestAvailableToPick");
                                    System.out.println();
                                    Timepoint  latestAvailableToPick = mem
```

```
ber1.getLatestAvailableToPick();
                                    if (latestAvailableToPick.isSetTimepoint
Type()) {
                                        System.out.println("
              TimepointType");
                                        System.out.println();
                                        System.out.println("
                  " + latestAvailableToPick.getTimepointType());
                                        System.out.println();
                                    }
                                    if (latestAvailableToPick.isSetDateTime())
 {
                                        System.out.println("
              DateTime");
                                        System.out.println();
                                        System.out.println("
                  " + latestAvailableToPick.getDateTime());
                                        System.out.println();
                                    }
                                }
                            }
                        }
                    }
                }
                if (listInventorySupplyResult.isSetNextToken()) {
                    System.out.println("            NextToken");
                    System.out.println();
                    System.out.println("                " + listInventorySup
plyResult.getNextToken());
                    System.out.println();
                }
            }
            if (response.isSetResponseMetadata()) {
                System.out.println("        ResponseMetadata");
                System.out.println();
                ResponseMetadata  responseMetadata = response.getResponse
Metadata();
                if (responseMetadata.isSetRequestId()) {
                    System.out.println("            RequestId");
                    System.out.println();
                    System.out.println("                " + response
Metadata.getRequestId());
                    System.out.println();
                }
            }
            System.out.println();


        } catch (FBAInventoryServiceMWSException ex) {

            System.out.println("Caught Exception: " + ex.getMessage());
            System.out.println("Response Status Code: " + ex.getStatusCode());

            System.out.println("Error Code: " + ex.getErrorCode());
            System.out.println("Error Type: " + ex.getErrorType());
            System.out.println("Request ID: " + ex.getRequestId());
            System.out.print("XML: " + ex.getXML());
        }
    }
```

# Making Requests

**Topics**


- Understanding Requests and Responses (p. 48)
- Authentication and Authorization (p. 51)
- Required Headers (p. 57)


The Amazon MWS Fulfillment API Section is part of the Amazon MWS platform.

This section offers information about how to formulate requests and receive responses with Amazon MWS, including detailed instructions to help you with authorization and authentication.

## Understanding Requests and Responses

**Topics**


- Requests (p. 48)
- Endpoints (p. 49)
- Responses (p. 50)


This section addresses how you formulate requests. After reading this section, you should understand the components and requirements of a request, know how to authenticate a request, and understand the content of responses.

### Requests

Amazon MWS 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 or HTTP body, respectively. We require HTTPS to prevent third-party eavesdropping on your communication with Amazon.

You must terminate each of the HTTP header lines with a carriage return and a line feed. For more information, go to Section 2.2 of the *HTTP/1.1 Specification*. Query requests must contain an `Action` parameter to indicate the action to be performed. The response is an XML document.

Amazon MWS limits requests to 1,000 per hour for the combination of the developer account and Amazon seller account.

## Structure of a Query Request

Amazon MWS Query requests are URLs. The URL consists of:

- **Endpoint—**The domain name of the request, such as `https://mws.amazonservices.com/`
- **Action—**The action you want to perform on the endpoint, such as `CreateInboundShipment`
- **Parameters—**Any request parameters



**Note**

For information about parameters that you *must* include in every Amazon MWS Query request, see Authorization (p. 56).

For information about the X-Amazon-User-Agent header line, required in every request, see X-Amazon-User-Agent Header (p. 58).

For a list of endpoints, see Endpoints (p. 49).

After the endpoint is a question mark (?), which separates the endpoint from the parameters.

## Example Query Request

Because the format is hard to read if left complete as a long string, (and might be cut off when viewing in certain formats), and although the Amazon MWS client's actual query string must not contain any line breaks or white spaces between the query string parameters, all Query examples in this guide are presented in the reader-friendly parsed format.

The following is an example Query request that lists items for an inbound shipment.

```
https://mws.amazonservices.com/FulfillmentInboundShipment/2010-06-01-BETA-02
?Action=ListInboundShipmentItems
&Version=2010-06-01-BETA-02
&AWSAccessKeyId=1QZHP81N5R44N89PFAKE
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Signature=VY6sqvdk01VEG0V+h4oj3FBFAKE=
&Timestamp=2010-06-01T02:40:36Z
&Marketplace=ATExampleER
&SellerId=1234567890
&ShipmentId=SSFI9ASOZZJH1J
```

# Endpoints

| Amazon Marketplace | MWS Endpoint | MWS Website |
|---|---|---|
| DE | https://mws.amazonservices.de | http://developer.amazonservices.de |
| FR | https://mws.amazonservices.fr | http://developer.amazonservices.fr |
| UK | https://mws.amazonservices.co.uk | http://developer.amazonservices.co.uk |
| US | https://mws.amazonservices.com | http://developer.amazonservices.com |

# Responses

## Response Messages

In response to an action request, Amazon MWS returns an XML data structure or a flat file that contains the results of the request.

## The Structure of a Response

If a request is successful, the response returns with the data requested.

The following example shows a successful response.

```
<?xml version="1.0"?>
    <CreateInboundShipmentResponse  xmlns:"http://mws.amazonaws.com/Fulfill
mentInboundShipment/2010-06-01-BETA-02/">
      <CreateInboundShipmentResult>
        <ShipmentId>FBAB1NX2</ShipmentId>
      </CreateInboundShipmentResult>
      <ResponseMetadata>
        <RequestId>98cf5fca-3948-4f97-9470-84be9531e7a7</RequestId>
      </ResponseMetadata>
    </CreateInboundShipmentResponse>
```

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

- An error code that identifies the type of error that occurred
- A message code that describes the error condition in a human-readable form
- An error type, identifying either the receiver or the sender as the error originator

The following example shows an error response.

```
<ErrorResponse xmlns="http://mws.amazonaws.com/doc/2010-06-01-BETA-02/">
    <Error>
        <Type>Sender</Type>
        <Code>InvalidClientTokenId</Code>
        <Message>The Access Key Id you provided does not exist in our re
cords.</Message>
        <Detail>com.amazonaws.mws.model.Error$Detail@17b6643</Detail>
    </Error>
    <RequestID>b7afc6c3-6f75-4707-bcf4-0475ad23162c</RequestID>
</ErrorResponse>
```

# Authentication and Authorization

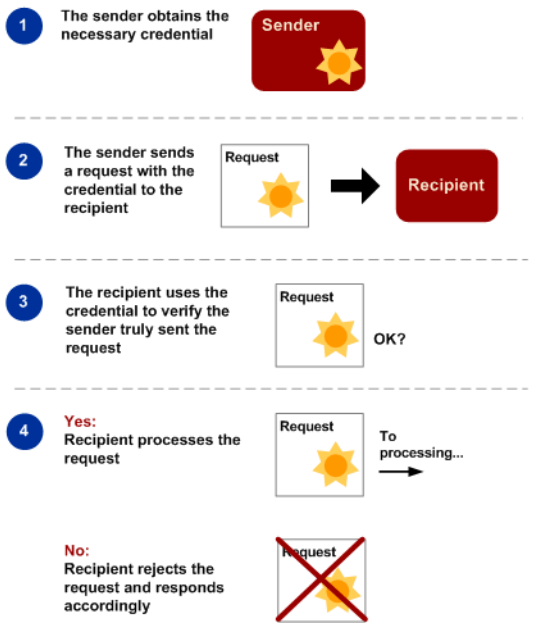
**Topics**


- What Is Authentication? (p. 51)
- MWS Credentials (p. 52)
- Calculating Signatures (p. 53)
- Authorization (p. 56)


This section describes authentication, which is the process for identifying and verifying who is sending a request; and authorization, which determines whether you have the credentials to perform a specific action.

## What Is Authentication?

Authentication is a process for identifying and verifying who is sending a request. The following diagram 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 Marketplace Web Service verifies both the identity of the sender and whether the sender is registered to use services offered by Amazon MWS. 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.

# MWS Credentials

When you register for MWS you are issued a pair of developer credentials, which you need to make successful requests to Amazon MWS, as well as a developer account identifier, which is necessary for you to retain for future use. If you are developing MWS applications or otherwise providing MWS-related development services to other sellers, you will need to provide this developer account identifier to those sellers so that they can authorize you to access their Amazon seller accounts with MWS.

Your MWS developer credentials are comprised of the following.

- Access Key ID (a 20-character, alphanumeric sequence)
  For example: 022QF0EXAMPLEH9DHM02
- Secret Access Key (a 40-character sequence)
  For example: kWcrlEXAMPLEM/LtmEENI/aVmYvHNif5zB+d9+ct



**Caution**

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

The Access Key ID is associated with your MWS registration. You include it in Amazon MWS 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 Amazon MWS. 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 Amazon MWS client libraries, you calculate the signature using your Secret Access Key. MWS 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 MWS calculates matches the signature you sent, the request is considered authentic. Otherwise, the request fails authentication and is not processed.

## Viewing Your MWS Credentials and Identifier

Your Access Key ID, Secret Access Key, and Account Number are displayed to you when you register to use MWS. They are not e-mailed to you. You should print the web page or save it to your hard drive.



**Important**

If you need to see the credentials and identifier again, you can repeat the MWS registration process. Your developer account credentials and identifier are displayed on the final page of the registration process.

# Calculating Signatures

The Amazon Marketplace Web Service client libraries calculate signatures for you. If you are using one of our libraries and sending messages, you do not need to calculate your signature or time stamp.

When accessing Amazon MWS, you must provide the following items so the request can be authenticated:

- **AccessKeyId—**Your MWS account is identified by your Access Key ID, which MWS 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 MWS.
- **SignatureVersion—**The signature version. This is MWS-specific information that tells MWS the algorithm you used to form the string that is the basis of the signature. For Amazon MWS, this is always `SignatureVersion=2`.
- **SignatureMethod—**The HMAC hash algorithm used to calculate your signature, either SHA1 or SHA256.
- **Timestamp—**Each request must contain the time stamp of the request. Depending on the operation you're using, you can provide an expiration date and time for the request instead of or in addition to the time stamp. See the particular operation for details of what is required and allowed for that function.

Following is the series of tasks required to authenticate requests to MWS. We assume you have already registered for MWS and received an Access Key ID and Secret Access Key.

You perform the first three tasks.

**Process for Authentication: Tasks You Perform**

| | |
|---|---|
| 1 | Construct a request to MWS. |
| 2 | Calculate a keyed-hash message authentication code (HMAC-SHA) 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 MWS. |

MWS performs the next three tasks.

**Process for Authentication: Tasks MWS Performs**

| | |
|---|---|
| 4 | MWS uses the Access Key ID to look up your Secret Access Key. |
| 5 | MWS 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 MWS matches the one you sent in the request, the request is considered authentic. If the comparison fails, the request is discarded, and MWS returns an error response. |

# Calculating a Signature

The request must include an HMAC-SHA signature, which is included in the `Signature` parameter of the request.

The following procedure describes how to compute the HMAC signature.

### To create the signature

1. Create the canonicalized query string that you need later in this procedure:

   a. Sort the UTF-8 query string components by parameter name with natural byte ordering. The parameters can come from the GET URI or from the POST body (when Content-Type is application/x-www-form-urlencoded).

   b. URL encode the parameter name and values according to the following rules:

      - Do not URL encode any of the unreserved characters that RFC 3986 defines. These unreserved characters are A-Z, a-z, 0-9, hyphen ( - ), underscore ( _ ), period ( . ), and tilde ( ~ ).
      - Percent encode all other characters with %XY, where X and Y are hex characters 0-9 and uppercase A-F.
      - Percent encode extended UTF-8 characters in the form %XY%ZA...
      - Percent encode the space character as %20 (and not +, as other common encoding schemes do).

   c. Separate the encoded parameter names from their encoded values with the equals sign ( = ) (ASCII character 61), even if the parameter value is empty.

   d. Separate the name-value pairs with an ampersand ( & ) (ASCII code 38).

2. Create the string to sign according to the following pseudo-grammar (the "\n" represents a line break).

```
StringToSign = HTTPVerb + "\n" +
               ValueOfHostHeaderInLowercase + "\n" +
               HTTPRequestURI + "\n" +
               CanonicalizedQueryString <from the preceding step>
```

   The HTTPRequestURI component is the HTTP absolute path component of the URI up to, but not including, the query string. If the HTTPRequestURI is empty, use a forward slash ( / ).

   The following example shows a string to sign.

```
POST
mws.amazonservices.com
/
&Version=2010-06-01-BETA-02
&AWSAccessKeyId=1QZHP81N5R44N89PFAKE
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Signature=VY6sqvdk01VEG0V+h4oj3FBFAKE=
&Timestamp=2010-06-01T02:40:36Z
&Marketplace=ATExampleER
&SellerId=1234567890
&Action=CreateInboundShipmentPlan
&ShipFromAddress.Name=Test Address
&ShipFromAddress.AddressLine1=Address Line 1
&ShipFromAddress.AddressLine2=Address Line 2
&ShipFromAddress.City=Seattle
&ShipFromAddress.StateOrProvinceCode=WA
&ShipFromAddress.CountryCode=US
```

```
&ShipFromAddress.PostalCode=98101
&LabelPrepPreference=
&InboundShipmentPlanRequestItem.1.SellerSKU=ssof_dev_drt_afn_item
&InboundShipmentPlanRequestItem.1.Quantity=10
&InboundShipmentPlanRequestItem.1.ASIN=
&InboundShipmentPlanRequestItem.1.Condition=New
&InboundShipmentPlanRequestItem.2.SellerSKU=ssof_dev_drt_mfn_item
&InboundShipmentPlanRequestItem.2.Quantity=4
&InboundShipmentPlanRequestItem.2.ASIN=
&InboundShipmentPlanRequestItem.2.Condition=New
```

3. Calculate an RFC 2104-compliant HMAC with the string you just created, your Secret Access Key as the key, and SHA1 or SHA256 as the hash algorithm.
4. Convert the resulting value to base64.
5. Use the resulting value as the value of the `Signature` request parameter.

## 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 that you provide the time stamp in the Coordinated Universal Time (UTC or zulu) time zone. For example: "2009-03-03T18:12:22Z" or "2009-02-23T18:12:22.093-07:00". The `Timestamp` field must contain the client's machine time in ISO8601 format; requests with a time stamp significantly different (15 minutes or greater) than the receiving machine's clock will be rejected to help prevent replay attacks.

Network Time Protocol (NTP) is a protocol designed to synchronize the clocks of computers over a network. For more information, go to http://www.ntp.org/.

Every request from Amazon MWS includes a `Date` header in its HTTP response that you can use to check whether your local machine's time matches our server's time: `2009-03-24T20:34Z`

You can also load https://mws.amazonservices.com/ in any Web browser (no `Signature` is required) to check on the MWS server time:

```
<?xml version="1.0"?>
<PingResponse>
<Timestamp timestamp="2009-03-24T20:29:19:22Z"/>
</PingResponse>
```

In order to allow us to extend the content of the `PingResponse`, any software you write to parse out the `Timestamp` should not break if sibling XML tags start to appear. Generally, you should ignore unknown tags in any XML we send you, according to the web architectural principle in Section 5.2 of http://www.w3.org/TR/webarch/.

If you specify a time stamp (instead of an expiration time), the request automatically expires 15 minutes after the time stamp (in other words, MWS does not process a request if the request time stamp is more than 15 minutes earlier than the current time on MWS 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.

# Authorization

Authentication determines who you are, whereas authorization determines whether you are allowed to perform a specific action. Amazon MWS authenticates your request using your Access Key ID and Secret Key. Authorization determines whether you've registered to use Amazon MWS. Only a request that is successfully identified as originating from a developer who has registered to use Amazon MWS is successfully processed.

Every request to MWS must contain:

- Access Key ID
- Seller ID

The Seller ID specifies the seller for which a request is intended. Again, the MWS developer account must be authorized for this seller in order for the request to succeed.

Amazon MWS requests are authenticated and authorized by verifying information contained within the request. This verification uses the information in the following table, which *must* be included in every Amazon MWS request.

| Name | Description | Required |
|---|---|---|
| *AccessKeyId* | Identifies the sender's developer account, which is used to look up the Secret Access Key.<br><br>Type: xs:string | Yes |
| Signature | Each request must contain a valid request signature. A request signature is calculated using the Secret Access Key assigned to the developer's account, which is a shared secret known only to Amazon MWS and the developer.<br><br>Type: xs:string | Yes |
| SignatureVersion | Specifies the version you want to use to calculate your signature. Version 2 is the only supported version.<br><br>Type: xs:string | Yes |
| SignatureMethod | Specifies the method used to calculate your signature.<br><br>Valid Values: HmacSHA256 \| HmacSHA1<br><br>Type: xs:string | Yes |
| Timestamp | The time stamp of the request, which ensures that malicious third parties cannot resubmit old requests.<br><br>Type: xs:datetime | Yes |

| Name | Description | Required |
|---|---|---|
| `Seller` | The seller account to which the operation is applied.<br><br>Type: xs:string | Yes |
| `Version` | The specific version of Amazon MWS to call (e.g., 2010-06-01-BETA-02).<br><br>Type: xs:string | Yes |

### Example Query Snippet

```
https://mws.amazonservices.com/FulfillmentInboundShipment/2010-06-01-BETA-02
?Action=CreateInboundShipmentPlan
&Version=2010-06-01-BETA-02
&AWSAccessKeyId=1QZHP81N5R44N89PFAKE
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Signature=VY6sqvdk01VEG0V+h4oj3FBFAKE=
&Timestamp=2010-06-01T02:40:36Z
&Marketplace=ATExampleER
&SellerId=1234567890
```

# Required Headers

**Topics**

- X-Amazon-User-Agent Header (p. 58)

This section explains how to formulate the X-Amazon-User-Agent field in the HTTP header, which is a required field for all MWS calls.

# X-Amazon-User-Agent Header

The X-Amazon-User-Agent field in the HTTP header is a required field for all MWS calls, and is used to identify your application, its version number, and programming language. The X-Amazon-User-Agent header information enables us to identify problems with particular applications, application versions, and programming languages.

The Amazon MWS client libraries provide an easy-to-use method for passing the X-Amazon-User-Agent parameter with every MWS request. When you initialize the client library, you add the Application or Company Name and the Version Number. Other HTTP libraries also provide easy methods for constructing X-Amazon-User-Agent headers, but if you have any difficulties with this requirement, please request assistance. You can contact us by using the contact form at:

| |
|---|
| DE: *http://developer.amazonservices.de* |
| FR: *http://developer.amazonservices.fr* |
| UK: *http://developer.amazonservices.co.uk* |
| US: *http://developer.amazonservices.com* |

To meet the requirements, you must specify the following:

1. Application name
2. Application version
3. Language name-value pair
4. Additional attributes (separated by semi-colons)

The following example illustrates a minimally acceptable X-Amazon-User-Agent header.

```
AppId/AppVersionId (Language=LanguageNameAndOptionallyVersion)
```

If you are a third-party application integrator, you might want an X-Amazon-User-Agent header similar to the following.

```
My Desktop Seller Tool/2.0 (Language=Java/1.6.0.11; Platform=Windows/XP)
```

If you are a large seller who is integrating through your own IT department, you might want an X-Amazon-User-Agent header similar to the following.

```
MyCompanyName/build1611 (Language=Perl; Host=jane.laptop.example.com)
```



**Note**

To specify additional attributes, use the format `AttributeName=Value;`, separating each name value pair with a semi-colon. If you want to use a backslash (\), escape it with another backslash (\\). Similarly, escape a forward slash in the application name (\/), an opening parenthesis in the application version (\(), an equal sign in the attribute name (\=), and both a closing parenthesis (\)), and a semicolon (\;) in attribute values.



**Tip**

Because the X-Amazon-User-Agent header is transmitted in every request, try not to exceed 200 characters to avoid consuming unnecessary bandwidth. We will reject an X-Amazon-User-Agent header if it is larger than 500 characters, but suggest you stay far below this hard limit.

# Appendix

**Topics**


- Inbound Label and Packing Slip Specifications (p. 60)
- Inbound Pallet Specifications (p. 65)
- Label Font Specifications (p. 65)


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**


- Item Label Specifications (p. 61)
- Shipping Label Specifications (p. 62)
- Packing Slip Specifications (p. 63)
- Label and Slip Approval (p. 65)


The fulfillment operations use 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 Seller-specified information



**Note**

The seller-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. 65) |
| 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.<br>• **For 300 dpi—**13.33 mils<br>• **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. Seller 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 seller-specific information below the Internal Use Only section, in a section clearly marked **Seller 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. 65) |

### 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. Seller information
3. Seller 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**

### Seller 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)
- Seller friendly name
- Seller ship from address

- FBA shipment ID, in both text and scannable barcode

### Seller Checklist Section

This section must include the following information:

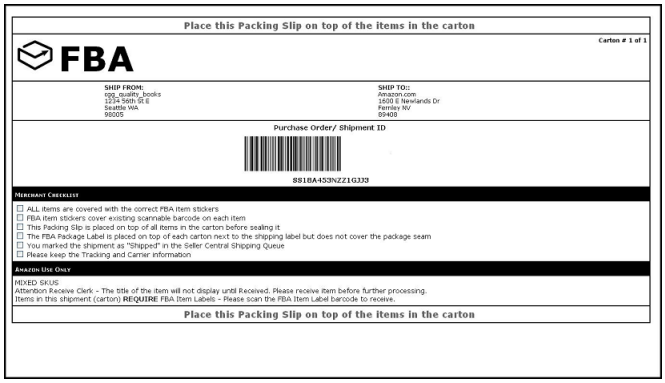
- All items are covered with the correct FBA item stickers
- FBA item stickers cover existing scannable barcode on each item
- This Packing Slip is placed on top of all items in the carton before sealing it
- The FBA Package Label is placed on top of each carton next to the shipping label but does not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


Place this Packing Slip on top of the items in the carton

FBA

Carton # 1 of 1

SHIP FROM:
qqq_quality_books
1234 56th St E
Seattle WA
98005

SHIP TO::
Amazon.com
1600 E Newlands Dr
Fernley NV
89408

Purchase Order/ Shipment ID

8818A453NZZ1GJJJ

MERCHANT CHECKLIST

- ☐ 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
- ☐ You marked the shipment as "Shipped" in the Seller Central Shipping Queue
- ☐ Please keep the Tracking and Carrier information

AMAZON USE ONLY

MIXED SKUS
Attention Receive Clerk - The title of the item will not display until Received. Please receive item before further processing.
Items in this shipment (carton) **REQUIRE** FBA Item Labels - Please scan the FBA Item Label barcode to receive.

Place this Packing Slip on top of the items in the carton

### Physical Specifications

| Specification | Description |
|---|---|
| Size | Letter (8.5" x 11") |
| Font | See Label Font Specifications (p. 65) |

### 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.<br><br>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. |
| 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 offers an optimized and cost-effective fulfillment solution for sellers. See Also Fulfillment by Amazon. |
| Amazon Standard Item Number | The Amazon Standard Item Number (ASIN) is a ten-digit Amazon 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 sellers 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 sellers. 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.<br>See Also SKU . |
| epoch | The date from which time is measured. For most Unix environments the epoch is January 1, 1970. |
| Fulfillment by Amazon | Fulfillment by Amazon is the general service Amazon provides that allows sellers 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 Seller Manual. |
| seller stock keeping unit | See MSKU. |
| MSKU | The seller stock keeping unit (SellerSKU or MSKU) is your product identifier.<br>See Also SKU . |
| multi-channel | Multi-channel refers to shipments that are sent at the seller's request to fulfill orders placed anywhere other than the Amazon 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.<br>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 when you sign up for an account. Used for request authentication. |

| | |
|---|---|
| SKU | A stock keeping unit is a specific seller's product identifier. The SKU is a critical piece of data in every inventory file that you submit to us. Amazon 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:<br>• Every product in your inventory file must have a unique SKU.<br>• 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.<br><br>An item is *sortable* if it is media or non-media. Anything classified as oversized is non-sortable.<br>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 sellers have of the same kind and condition.<br>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 sellers 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 sellers. 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 Fulfillment Inventory API Section 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. |

# Document Conventions

This section lists the common typographical and symbol use conventions for 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.<br><br>You can use this resource regularly. |
| Code in text | Inline code samples (including XML) and commands are identified with a special font.<br><br>You can use the command `java -version`. |
| Code blocks | Blocks of sample code are set apart from the body and marked accordingly.<br><br>`# ls -l /var/www/html/index.html`<br>`-rw-rw-r--  1 root root 1872 Jun 21 09:33`<br>`/var/www/html/index.html`<br>`# date`<br>`Wed Jun 21 09:33:42 EDT 2006` |
| Emphasis | Unusual or important words and phrases are marked with a special font.<br><br>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.<br><br>See Document Conventions (p. 70). |

| 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.<br><br>If the value is `null`, the returned response will be `false`. |
| Product and feature names | Named products and features are identified on first use.<br><br>Create an *Amazon Machine Image* (AMI). |
| Operations | In-text references to operations.<br><br>Use the `GetHITResponse` operation. |
| Parameters | In-text references to parameters.<br><br>The operation accepts the parameter `AccountID`. |
| Response elements | In-text references to responses.<br><br>A container for one `CollectionParent` and one or more `CollectionItems`. |
| Technical publication references | References to other publications. If the reference is hyperlinked, it is also underscored.<br><br>For detailed conceptual information, see the *Amazon Mechanical Turk Developer Guide*. |
| User entered values | A special font marks text that the user types.<br><br>At the password prompt, type **`MyPassword`**. |
| User interface controls and labels | Denotes named items on the UI for easy identification.<br><br>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.<br><br>% ec2-register *`<your-s3-bucket>`*/image.manifest<br><br>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.<br><br>`% data = hdfread (start \| stride \| edge)` |
| Optional parameters<br><br>XML variable text | [square brackets] | Within a code description, square brackets denote completely optional commands or parameters.<br><br>`% sed [-n, -quiet]`<br><br>Use square brackets in XML examples to differentiate them from tags.<br><br>`<CustomerId>[ID]</CustomerId>` |
| Variables | <arrow brackets> | Within a code sample, arrow brackets denote a variable that must be replaced with a valid value.<br><br>`% ec2-register <your-s3-bucket>/image.manifest` |

# Index


## A

about the service, 5
Access Key ID, 52
  defined, 67
accounts, 52, 56
  associating, 56
  developer, 56
  seller, 56
Amazon fulfillment center, 67
Amazon Fulfillment Network
  defined, 67
audience, 1
authentication
  process of, 51
  use in MWS, 51
authorization, 56

## C

canceling a fulfillment order, 40
checking the service status
  inbound, 12
  inventory, 42
  outbound, 22
creating a fulfillment order, 24

## D

developer account identifier, 52
diagnostics, 58

## E

endpoints, 49
  Query requests, 48
epoch
  defined, 68
error codes
  responses, 50

## F

FBA
  defined, 68
features, 5
FNSKU
  defined, 68
fulfillment orders
  canceling, 40
  creating, 24
  listing all, 35
  requesting information about, 26

## G

guide organization, 1

## H

HMAC
  defined, 68
  signatures, 53
HTTP header, 58

## I

inbound
  overview of inbound shipment process, 9
  use cases, 8
introduction, 1
inventory
  use cases, 41
inventory supply
  listing, 44
item label, 61

## L

label specifications, 60
labeling a shipment, 9
labels
  item, 61
  packing slip, 63
  shipping, 62
listing all fulfillment orders, 35
listing inventory supply, 44

## M

metrics, 58
MWS
  credentials, 52

## N

non-sortable
  defined, 68

## O

outbound
  use cases, 21

## P

packing slip, 63
pallet specifications, 65
parameters summary, 56

## Q

Query requests, 48
quiet zone
  defined, 68
  example, 60

## R

requesting fulfillment order information, 26

requests, 48, 49, 56
required knowledge, 1
resources, 1
response messages, 50
responses, 50

## S

Secret Access Key, 52
section
  inbound, explained, 5
  inventory, explained, 5
  outbound, explained, 5
Seller ID, 56
seller stock keeping unit (see MSKU)
shipments
  labeling, 9
  setting status of, 20
shipping label, 62
signature, 53
  base64 encoded, 55
  example of, 55
Signature, 56
SignatureMethod, 56
SignatureVersion, 56
signing up, 56
SKU, defined, 69
sortable
  defined, 69
specifications
  label, 60
  pallet, 65
status
  checking for service, 12, 22, 42
  requesting for fulfillment order, 26
  setting for shipment, 20
stickered, defined, 69
stickerless, defined, 69
stock keeping unit (see SKU)

## T

tasks, code for
  creating a new shipment, 18
  getting destination fulfillment center for, 14
  getting fulfillment identifier for, 14
  getting shipment identifier for, 14
throttling, 48
time stamp
  using in requests, 55

## U

use cases
  inbound, 8
  inventory, 41
  outbound, 21

## V

Version, 56

## W

workflow, 5

## X

X-Amazon-User-Agent header, 58