amazonpayments

# The Instant Order Processing Notification API Guide

A reference guide for developers who want to automate order notifications with their Pay with Amazon orders.

## Table of Contents

# 1 Overview of the Instant Order Processing Notification API

Order notifications inform you about new orders or status changes for existing orders placed using Pay with Amazon. When you receive these notifications, you can pass them to your internal order management system (OMS) and then process the orders.

An instant order notification is an HTTPS POST request, issued when the order is placed or changed, containing the XML-based notification data in its body. To use the Instant Order Processing Notification API, you must be running a web service that can receive the Pay with Amazon notifications and then pass the required information to your internal order fulfillment system.

This document shows the types and structure of different notifications. This document also explains how you can enable notification, how to authenticate notifications, and how to process the received notifications.

## 1.1 Prerequisites for Using the Instant Order Processing Notification API

Before you can use the Instant Order Processing Notification API, your systems must meet the following prerequisites.

1. You must be using the `order.xsd` version 2009-05-15 or later. (Download the latest version here.)
2. You must be using the `iopn.xsd` version 2008-11-30 or later. (Download the latest version here.)
3. You must be running a web service that can receive HTTPS POST requests made to your endpoint using port 443.
4. Your HTTPS must use valid SSL certificates from a trusted certificate provider.

## 1.2 Your Tasks When Using the Instant Order Processing Notification API

You have two tasks to accomplish when using the Instant Order Processing Notification API:

1. Enable Order Notifications
2. Process the Order Notifications

The rest of this document discusses how to accomplish these two tasks.

## 1.3 Other Documentation

You can read our downloadable files (PDF) to understand more about Pay with Amazon.

| Resource | Audience, Purpose, and Goals |
|---|---|
| *Getting Started Guide* (PDF) | For merchants who want to use a simple, step-by-step approach in setting up Pay with Amazon on their websites, from start to finish. The Getting Started Guide walks merchants through the process, from gathering the required information for signing up for an account setting up a button on their websites. |
| *Implementation Guide Using HTML-based Button Code and Shopping Carts* (PDF) | For merchants and developers who want detailed information about integrating their website with Pay with Amazon using HTML-based buttons and shopping carts. Designed for HTML-based integration. |
| *Implementation Guide Using XML-based Button Code and Shopping Carts* (PDF) | For merchants and developers who want detailed information about integrating their website with Pay with Amazon using XML-based buttons and shopping carts. Designed for XML-based integration. |

## 1.4 What's New in This Document

| Ver | Date | Changes |
|---|---|---|
| 3.0 | 2015-01-20 | Updates for IN Pay with Amazon release. |
| 2.1 | 2014-06-03 | Updated references to a menu option "Access Key" to "MWS Access Key". And removed outdated links. |
| 2.0 | 2014-03-31 | Updated to clarify access and secret access keys. |
| 1.9 | 2011-10-31 | Deprecated Shopping Cart |
| 1.8 | 2011-10-31 | Updated to reflect current functionality |
| 1.7 | 2010-05-30 | Update for Implementation Guide |
| 1.0 | 2009-02-01 | Initial Release |

# 2 Specifications

The following section specifies the content and type of order notifications you can receive.

## 2.1 Order Notifications Content

This is general format for the content of an order notification. For details of these parameters, please see Appendix A - The Instant Order Processing Notification API Schema.

```
<xs:complexType name=[NotificationType]>
  <xs:sequence>
        <xs:element name="NotificationReferenceId" type="xs:string" />
        <xs:element name="ProcessedOrder" type="tns:ProcessedOrder" />
  </xs:sequence>
</xs:complexType>
```

## 2.2 Order Notification Types

Order notifications are HTTPS POST requests containing the notification data formatted as XML. You can receive one of three types of notifications:

1. New Order Notification
2. Order Ready-to-Ship Notification
3. Order Canceled Notification

### 2.2.1 1. New Order Notification

We send you this notification when a new order is placed. Generally, this notification is sent immediately when the order is first created; however, there is a 90-minute delay for 1-Click® or Express Checkout orders.

> **Note**
>
> Do not ship an order when you receive the New Order notification. Please wait until you receive the Order Ready-to-Ship notification.

### 2.2.2 2. Order Ready-to-Ship Notification

We send you this notification when we authorize the buyer's payment method. This notification indicates that you can fulfill the order (send the items to the buyer).

> **Note**
>
> Do not ship an order until you receive this Order Ready-to-Ship notification.

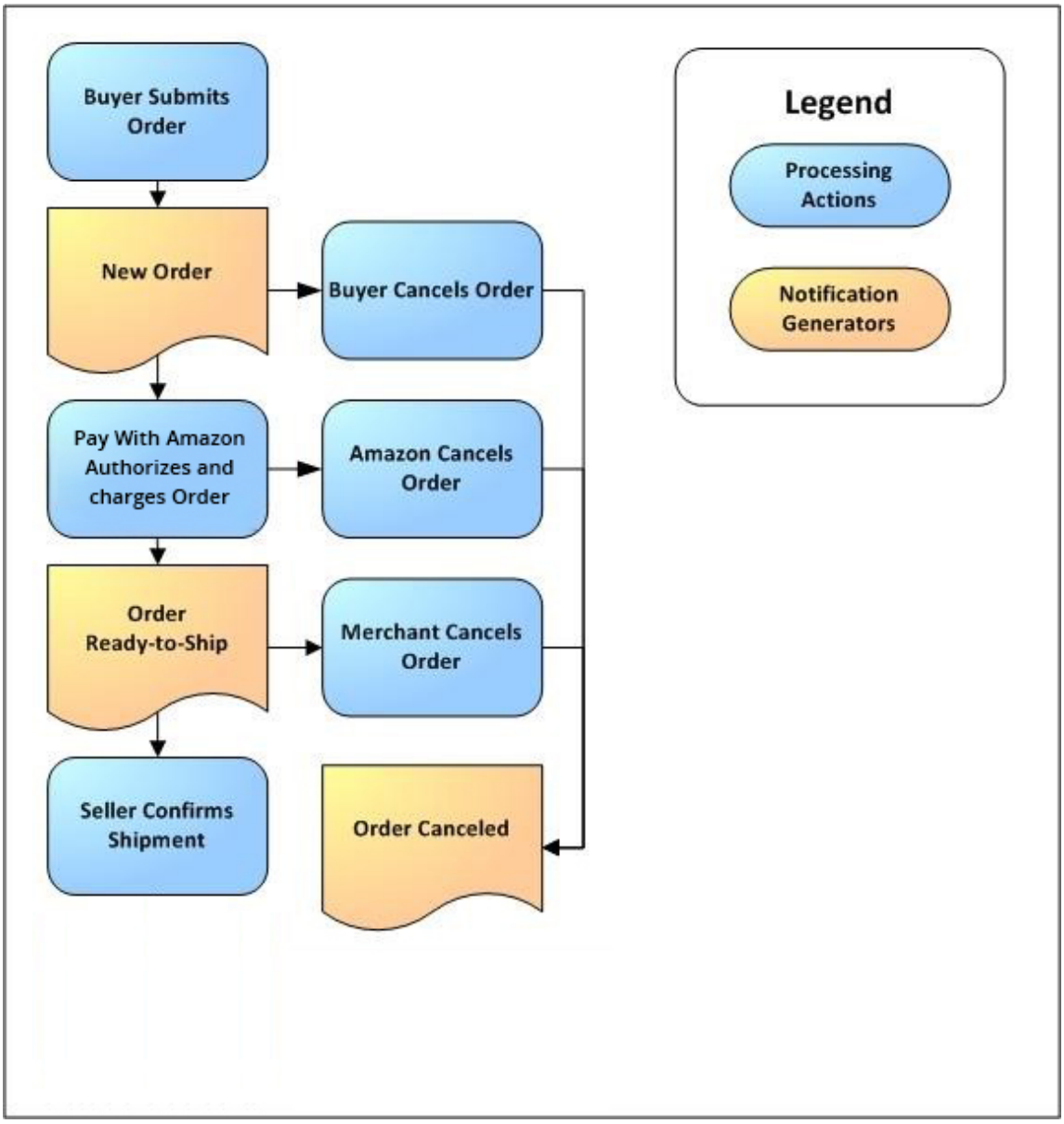
### 2.2.3 3. Order Canceled Notification

We send you this notification when the order is canceled. An order can be canceled by the seller, the buyer, or Amazon Payments.

> **Note**
>
> No other details of the cancellation beyond this notification are provided.

## 2.3 The Order Notification Flow

This image shows the notification flow; that is, the order you can expect notifications to arrive in. Note that the state changes shown in yellow result in notifications.




In some cases, the New Order notification might arrive after the Order Ready-to-Ship or Order Canceled notifications. See Why Did I Receive an Order Canceled Notification or Order Ready-to-Ship Notification Before the New Order Notification for the Same Order? for more information about these issues.

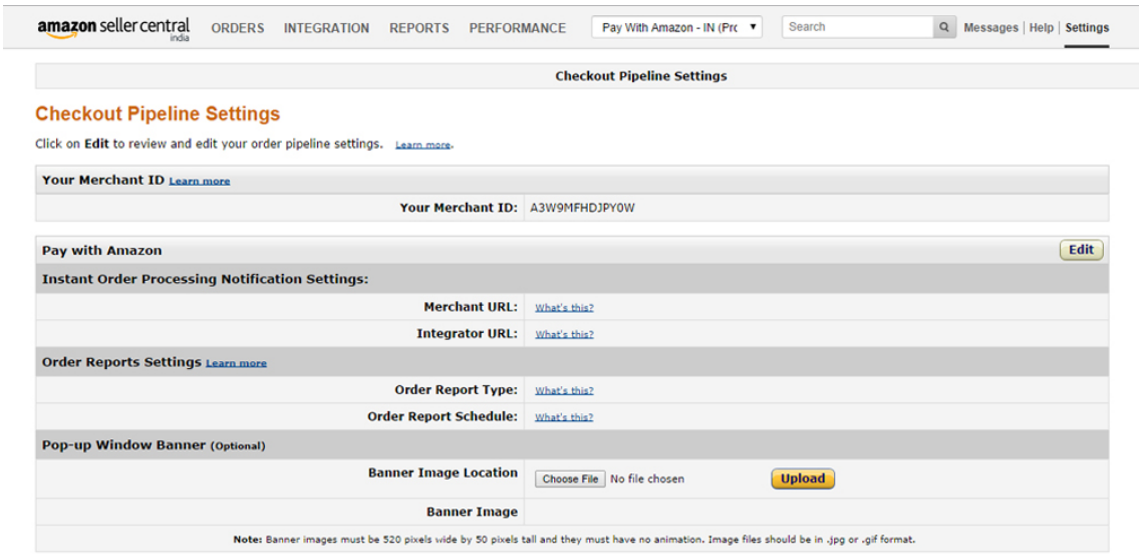
## 2.4 Notifications and Multiple Checkouts

For multiple checkouts that are consolidated into a single Amazon Order ID, merchants will receive only one `NewOrderNotification` and `OrderReadyToShipNotification` Instant Order Processing Notification message. Each checkout has distinct `CartId`, and for the consolidated orders, there will be multiple distinct `CartId`s in new and ready-to-ship Instant Order Processing Notification messages.

# 3 Enabling the Instant Order Processing Notification API

To enable the Instant Order Processing Notification API, log in to Seller Central and specify your Merchant URL endpoint (on your web service), your Integrator's endpoint (on your Integrator's web service), or both. You must enter at least one endpoint URL to enable Instant Order Processing Notification API. If you do not have an Integrator, you do not need to specify the Integrator endpoint.

The following screen shot shows the new Instant Order Processing Notifications Settings section in the Checkout Pipeline Settings page.

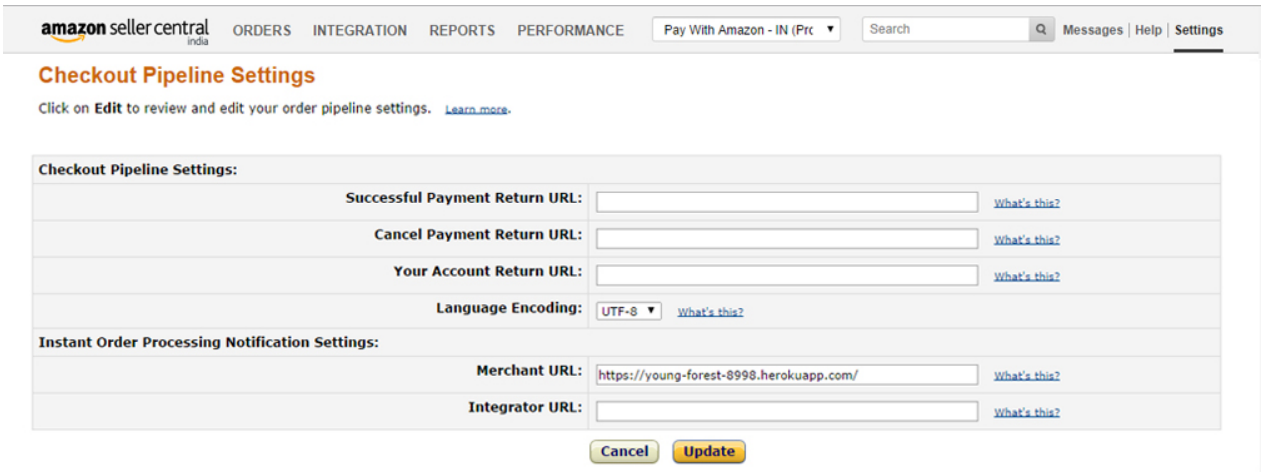



To set or edit the endpoint settings, follow these instructions:

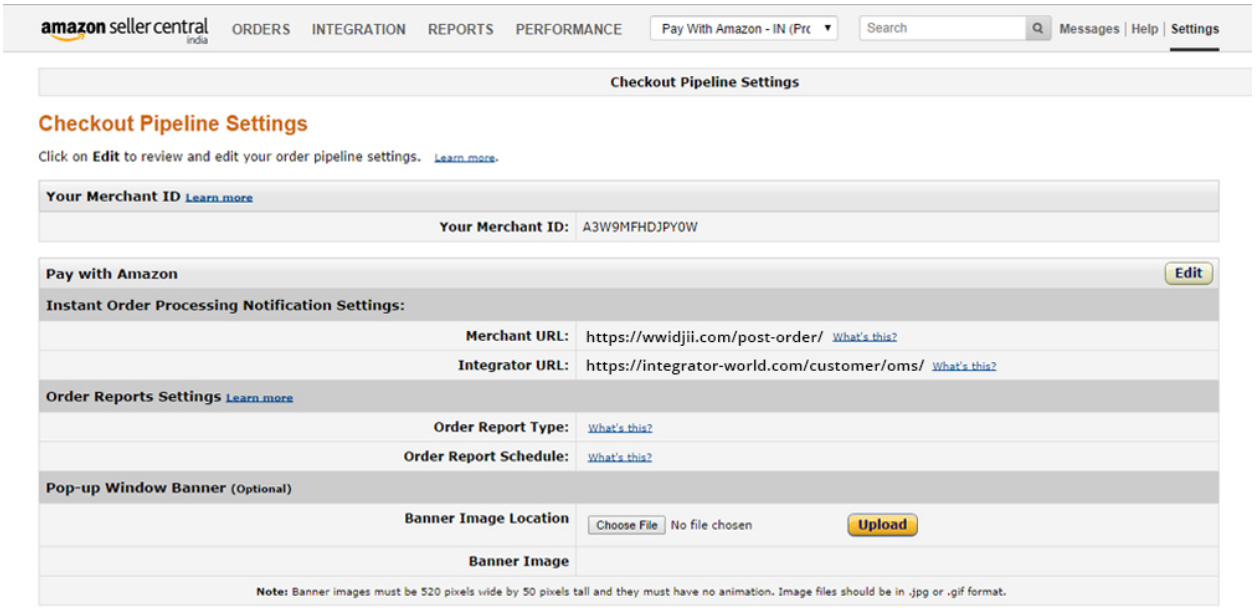
## 3.1 Set or Edit Your Endpoint URL

1. Log in to Seller Central.

2. Click **Settings** > **Checkout Pipeline Settings** and then click **Edit.**
   *The following screen appears*:




3. In the **Merchant URL** box, enter the endpoint on your web service that accepts notifications.
   *Remember to use* `https://` *for your Production account.*
4. In the **Integrator URL** box, enter the endpoint on your integrator's web service that accepts notifications.
   *Remember to use* `https://` *for your Production account.*
5. Click **Update.**
   *The endpoint URLs appear on the Settings page*:




## 3.2 Retries to the Endpoint

The following explains how retries are handled to the endpoint

- If both the Merchant URL and Integrator URL are set, then notifications are sent to both endpoints.
- If only one endpoint is set, then notifications are sent only to that endpoint.
- If a specified endpoint is down (not available), then retries are done only for that endpoint.

If an endpoint URL is not available or not responding to the notification request, then the notification is re-sent to the URL according to the Retry Policy.

**Note**

You must specify at least one endpoint when you use the Instant Order Processing Notification API.

# 4 Receiving and Processing Notifications

To enable the Instant Order Processing Notification API, you must establish a web service that receives and processes Pay with Amazon notifications. Your web service must be secured by SSL v3 or TLS and must use a valid SSL certificate. The URL you use for your Production account must use HTTPS and the default secure port 443. The URL you use for your Sandbox account can use either port 443 (for secure HTTPS requests) or port 80 (for unsecure HTTP requests).

> **Notes**
>
> If your receiver end point is down, Pay with Amazon will retry sending the notification for 14 days. For information on the frequency of these notification retries, please see the Retry Policy.

Pay with Amazon authenticates the notification by calculating a `Signature` using the merchant's Seller Central Secret Access Key. The following sections explain the contents of the notification.

> **Notes**
>
> Be sure you use the Seller Central Secret Access Key to validate and generate signatures for requests and responses – this is the key you find in Seller Central on the **Integration** > **MWS Access Key** page.

## 4.1 The Notification Content

The notification we send you is a simple string, URL-encoded and formatted using UTF-8. The request contains following six sections.

| Entry | Definition | Inclusion |
|---|---|---|
| 1. The UUID | A Universally Unique Identifier, unique for each notification, whether it's the original notification or a retried notification. | Only for signed carts |
| 2. The Timestamp | The request time using GMT, unique for each notification, whether it's the original notification or a retried notification. | Only for signed carts |
| 3. The Signature | Calculated using the `UUID`, the `Timestamp`, and your Seller Central Secret Access Key. | Only for signed carts |
| 4. The NotificationType | The notification type, which can be `NewOrderNotification`, `OrderReadyToShipNotification`, or `OrderCancelledNotification`. | Always included |
| 5. The NotificationData | The content (body) of the request | Always included |
| 6. The AWSAccessKeyId | The Seller Central Access Key ID used to sign the cart. | Only for signed carts |

Note that `Signature` and `Timestamp` have an initial capital letter. `UUID` is all capital letters. `NotificationData`, and `NotificationType` are "camel-capped."

This request is a set of key-value pairs, concatenated as one string, with each key-value pair delimited using the standard "&" character. The key is in plain text, and the value is URL-encoded.

The request contains the following key-value pairs. The order is not important (the spaces are for clarity):

```
UUID=[UUID_value] & Timestamp=[Timestamp_value] & Signature=[Signature_value] &
NotificationType=[notification_type] & NotificationData=[notification_data_value] &
AWSAccessKeyId=[ AWSAccessKeyId_value]
```

Here is an example of a New Order Notification:

```
UUID=5d1769d2-b217-4981-a499-fbad1b9acf15&Timestamp=2009-01-20T06%3A41%3A20.471Z&AWSAccessKe
yId=AKIAJC2EVPPNIFRXIVKQ&Signature=Ltu1alXSqkyZRqzVSjI%2BDPuWQms%3D&NotificationType=NewOrde
rNotification&NotificationData=%3C%3Fxml%20version%3D%221.0%22%20encoding%3D%22UTF-8%22%3F%3
E%0A%3CNewOrderNotification%20xmlns%3D%22http%3A%2F%2Fpayments.amazon.com%2Fcheckout%2F2009-
05-15%2F%22%3E%0A%20%20%3CNotificationReferenceId%3Eae51d3a6-7843-4cbb-ad1d-ee8cc591e10d%3C%
2FNotificationReferenceId%3E%0A%20%20%3CProcessedOrder%3E%0A%20%20%20%20%3COrderChannel%3EAm
azon%20Checkout%20%28Live%29%3C%2FOrderChannel%3E%0A%20%20%20%20%3CAmazonOrderID%3E101-12345
67-9876543%3C%2FAmazonOrderID%3E%0A%20%20%20%20%3COrderDate%3E2009-08-31%3C%2FOrderDate%3E%0
A%20%20%20%20%3CBuyerInfo%3E%0A%20%20%20%20%20%20%3CBuyerName%3EKelly%20Green%3C%2FBuyerName
%3E%0A%20%20%20%20%20%20%3CBuyerEmailAddress%3Esomeone%40amazon.com%3C%2FBuyerEmailAddress%3
E%0A%20%20%20%20%3C%2FBuyerInfo%3E%0A%20%20%20%20%3CShippingAddress%3E%0A%20%20%20%20%20%20%
3CName%3EKelly%20Green%3C%2FName%3E%0A%20%20%20%20%20%20%3CAddressFieldOne%3E123%20Oak%20Ave
nue%20SE%20%3C%2FAddressFieldOne%3E%0A%20%20%20%20%20%20%3CAddressFieldTwo%3EApt.%20221-B%20
%3C%2FAddressFieldTwo%3E%0A%20%20%20%20%20%20%3CCity%3ESeattle%3C%2FCity%3E%0A%20%20%20%20%2
0%20%3CState%3EWA%3C%2FState%3E%0A%20%20%20%20%20%20%3CPostalCode%3E98104-1234%3C%2FPostalCo
de%3E%0A%20%20%20%20%20%20%3CCountryCode%3EUSA%3C%2FCountryCode%3E%0A%20%20%20%20%3C%2FShipp
ingAddress%3E%0A%20%20%20%20%20%20%3CShippingServiceLevel%3EStandard%3C%2FShippingServiceLevel%3E%
0A%20%20%20%20%3CProcessedOrderItems%3E%0A%20%20%20%20%20%20%3CProcessedOrderItem%3E%0A%20%2
0%20%20%20%20%20%20%3CAmazonOrderItemCode%3E12345%3C%2FAmazonOrderItemCode%3E%0A%20%20%20%20
%20%20%20%20%3CMerchantId%3EAEIOU1234AEIOU%3C%2FMerchantId%3E%0A%20%20%20%20%20%20%20%20%3CS
KU%3EABC123%3C%2FSKU%3E%0A%20%20%20%20%20%20%20%20%3CTitle%3ERed%20Fish%3C%2FTitle%3E%0A%20%
20%20%20%20%20%20%20%3CDescription%3EA%20red%20fish%20packed%20in%20spring%20water.%3C%2FDes
cription%3E%0A%20%20%20%20%20%20%20%20%3CClientRequestId%3E12345%3C%2FClientRequestId%3E%0A%
20%20%20%20%20%20%20%20%3CCartId%3Emiq%3A%2F%2Frose%3A1.0%2Fcart%2Fcba%3A1.0%2Fcart%3A2.0%2F
AZ4B0ZS3LGLX%2FA3DBMG2ZGA2JEQ%2F1-1232431180394-57EEF607E6D40DB5%3C%2FCartId%3E%0A%20%20%20%
20%20%20%20%20%3CIntegratorId%3EVWXYZ98765VWXYZ%3C%2FIntegratorId%3E%0A%20%20%20%20%20%20%20
%20%3CIntegratorName%3Ewww.IntegratorWorld.com%3C%2FIntegratorName%3E%0A%20%20%20%20%20%20%2
0%20%3CPrice%3E%0A%20%20%20%20%20%20%20%20%20%20%3CAmount%3E5.0%3C%2FAmount%3E%0A%20%20%20%2
0%20%20%20%20%20%20%20%3CCurrencyCode%3EUSD%3C%2FCurrencyCode%3E%0A%20%20%20%20%20%20%20%20%3C%
2FPrice%3E%0A%20%20%20%20%20%20%20%20%3CQuantity%3E1%3C%2FQuantity%3E%0A%20%20%20%20%20%20%2
0%20%3CWeight%3E%0A%20%20%20%20%20%20%20%20%20%20%3CAmount%3E1.0%3C%2FAmount%3E%0A%20%20%20%
20%20%20%20%20%20%20%3CUnit%3Elb%3C%2FUnit%3E%0A%20%20%20%20%20%20%20%20%3C%2FWeight%3E%0A%2
0%20%20%20%20%20%20%20%3CFulfillmentNetwork%3EMERCHANT%3C%2FFulfillmentNetwork%3E%0A%20%20%2
0%20%20%20%20%20%20%3CItemCharges%3E%0A%20%20%20%20%20%20%20%20%20%20%3CComponent%3E%0A%20%20%2
0%20%20%20%20%20%20%20%20%20%20%3CType%3EPrincipal%3C%2FType%3E%0A%20%20%20%20%20%20%20%20%20%2
0%20%20%3CCharge%3E%0A%20%20%20%20%20%20%20%20%20%20%3CAmount%3E5.0%3C%2FAmount%
3E%0A%20%20%20%20%20%20%20%20%20%20%3CCurrencyCode%3EUSD%3C%2FCurrencyCode%3E%0A
%20%20%20%20%20%20%20%20%20%20%3C%2FCharge%3E%0A%20%20%20%20%20%20%20%20%20%20%3C%2FCo
mponent%3E%0A%20%20%20%20%20%20%20%20%20%20%3CComponent%3E%0A%20%20%20%20%20%20%20%20%20%20%
20%20%3CType%3EShipping%3C%2FType%3E%0A%20%20%20%20%20%20%20%20%20%20%20%20%3CCharge%3E%0A%2
0%20%20%20%20%20%20%20%20%20%20%20%20%3CAmount%3E0.0%3C%2FAmount%3E%0A%20%20%20%20%20%20%
20%20%20%20%20%20%20%20%3CCurrencyCode%3EUSD%3C%2FCurrencyCode%3E%0A%20%20%20%20%20%20%20%20
%20%20%20%20%3C%2FCharge%3E%0A%20%20%20%20%20%20%20%20%20%20%3C%2FComponent%3E%0A%20%20%20%2
```

```
0%20%20%20%20%20%20%3CComponent%3E%0A%20%20%20%20%20%20%20%20%20%20%20%3CType%3ETax%3C%2F
Type%3E%0A%20%20%20%20%20%20%20%20%20%20%20%3CCharge%3E%0A%20%20%20%20%20%20%20%20%20%20%
20%20%20%20%3CAmount%3E0.0%3C%2FAmount%3E%0A%20%20%20%20%20%20%20%20%20%20%20%3CCur
rencyCode%3EUSD%3C%2FCurrencyCode%3E%0A%20%20%20%20%20%20%20%20%20%20%20%3C%2FCharge%3E%0
A%20%20%20%20%20%20%20%20%20%3C%2FComponent%3E%0A%20%20%20%20%20%20%20%20%20%3CCompone
nt%3E%0A%20%20%20%20%20%20%20%20%20%20%20%3CType%3EShippingTax%3C%2FType%3E%0A%20%20%20%2
0%20%20%20%20%20%20%20%20%3CCharge%3E%0A%20%20%20%20%20%20%20%20%20%20%20%3CAmount%
3E0.0%3C%2FAmount%3E%0A%20%20%20%20%20%20%20%20%20%20%20%3CCurrencyCode%3EUSD%3C%2F
CurrencyCode%3E%0A%20%20%20%20%20%20%20%20%20%20%20%3C%2FCharge%3E%0A%20%20%20%20%20%20%2
0%20%20%20%3C%2FComponent%3E%0A%20%20%20%20%20%20%20%20%20%3CComponent%3E%0A%20%20%20%20%
20%20%20%20%20%20%20%3CType%3EPrincipalPromo%3C%2FType%3E%0A%20%20%20%20%20%20%20%20%20%2
0%20%20%3CCharge%3E%0A%20%20%20%20%20%20%20%20%20%20%20%3CAmount%3E0.0%3C%2FAmount%
3E%0A%20%20%20%20%20%20%20%20%20%20%20%3CCurrencyCode%3EUSD%3C%2FCurrencyCode%3E%0A
%20%20%20%20%20%20%20%20%20%20%20%3C%2FCharge%3E%0A%20%20%20%20%20%20%20%20%20%3C%2FCo
mponent%3E%0A%20%20%20%20%20%20%20%20%20%3CComponent%3E%0A%20%20%20%20%20%20%20%20%20%20%
20%20%3CType%3EShippingPromo%3C%2FType%3E%0A%20%20%20%20%20%20%20%20%20%20%20%3CCharge%3E
%0A%20%20%20%20%20%20%20%20%20%20%20%20%3CAmount%3E0.0%3C%2FAmount%3E%0A%20%20%20%20%2
0%20%20%20%20%20%20%20%20%3CCurrencyCode%3EUSD%3C%2FCurrencyCode%3E%0A%20%20%20%20%20%20%
20%20%20%20%20%20%3C%2FCharge%3E%0A%20%20%20%20%20%20%20%20%20%3C%2FComponent%3E%0A%20%20
%20%20%20%20%20%20%3C%2FItemCharges%3E%0A%20%20%20%20%20%20%3C%2FProcessedOrderItem%3E%0A%20
%20%20%20%3C%2FProcessedOrderItems%3E%0A%20%20%3C%2FProcessedOrder%3E%0A%3C%2FNewOrderNotifi
cation%3E
```

The key is in plain text (in red) and the value is URL-encoded; each key-value pair is delimited using the "&" character.

The following section explains the six components in the Instant Order Processing Notification API request you receive and how you can use them to build your response.

#### 4.1.1 1. The UUID

The `UUID` (Universally Unique Identifier) is a string created by AWS (Amazon Web Services).

Here is an example of the UUID in the request you receive:

```
UUID=d6a58609-d9ea-415c-95c6-d7c2528fca09
```

You'll use this UUID later when you calculate the `Signature` to validate the request. (If you did not specify that you accept only signed carts, you will not receive the `UUID` in your merchant request.)

#### 4.1.2 2. The Timestamp

The `Timestamp` is the date and time the request was sent from our servers.

Here is an example of the `Timestamp` in the request you receive:

```
Timestamp=2008-11-14T20%3A57%3A07.146Z
```

This evaluates to 14-Nov-2008 at 20:57:07.146 Greenwich Mean Time, or 14-Nov-2008 12:57:07.146 Pacific, or 15 Nov 2008 02:27:07.146 IST.

You'll use this `Timestamp` later when you calculate the `Signature` to validate the request. (If you did not specify that you accept only signed carts, you will not receive the `Timestamp` in your merchant request.)

#### 4.1.3 3. The Signature

To create the `Signature`, we concatenate the `UUID` and `Timestamp`, apply a standard signature algorithm using your Seller Central Secret Access Key, and then URL encode it. (If you did not specify that you accept only signed carts, you will not receive the `Signature` in your merchant request.) For information on generating and viewing your Seller Central Secret Access Key, click the **Access Key** option from the **Integration** menu in Seller Central.

Here is an example of the signature in the request you receive.

```
Signature=MRYUB4bk%2B3ehw0BAbzk6mp6SxXw%3D
```

You'll use this `Signature` later when you calculate the `Signature` using your Seller Central Secret Access Key to validate the request.

#### 4.1.4 4. The NotificationType

The `NotificationType` identifies the type of notification. Its values are:

- NewOrderNotification
- OrderReadyToShipNotification

#### 4.1.5 5. OrderCancelledNotification

Here is an example of the `NotificationType` in the request you receive:

```
NotificationType=NewOrderNotification
```

#### 4.1.6 6. The NotificationData

The `NotificationData` is the URL-encoded content of the cart.

Here is an example of the order notification data using URL encoding.

```
NotificationData=%3c%3fxml+version%3d%221.0%22+encoding%3d%22UTF-8%22%3f%3e%3cNewOrderNotifi
cation+xmlns%3d%22http%3a%2f%2fpayments.amazon.com%2fcheckout%2f2009-05-15%2f%22%3e%3cNotifi
cationReferenceId%3eae51d3a6-7843-4cbb-ad1d-ee8cc591e10d%3c%2fNotificationReferenceId%3e%3cP
rocessedOrder%3e%3cOrderChannel%3eAmazon+Checkout+(Live)%3c%2fOrderChannel%3e%3cAmazonOrderI
D%3e101-1234567-9876543%3c%2fAmazonOrderID%3e%3cOrderDate%3e2009-08-31%3c%2fOrderDate%3e%3cB
uyerInfo%3e%3cBuyerName%3eKelly+Green%3c%2fBuyerName%3e%3cBuyerEmailAddress%3esomeone%40amaz
on.com%3c%2fBuyerEmailAddress%3e%3c%2fBuyerInfo%3e%3cShippingAddress%3e%3cName%3eKelly+Green
%3c%2fName%3e%3cAddressFieldOne%3e123+Oak+Avenue+SE%3c%2fAddressFieldOne%3e%3cAddressFieldTw
o%3eApt.+221-B%3c%2fAddressFieldTwo%3e%3cCity%3eSeattle%3c%2fCity%3e%3cState%3eWA%3c%2fState
%3e%3cPostalCode%3e98104-1234%3c%2fPostalCode%3e%3cCountryCode%3eUSA%3c%2fCountryCode%3e%3c%
2fShippingAddress%3e%3cShippingServiceLevel%3eStandard%3c%2fShippingServiceLevel%3e%3cProces
sedOrderItems%3e%3cProcessedOrderItem%3e%3cAmazonOrderItemCode%3e12345%3c%2fAmazonOrderItemC
ode%3e%3cMerchantId%3eAEIOU1234AEIOU%3c%2fMerchantId%3e%3cSKU%3eABC123%3c%2fSKU%3e%3cTitle%3
eRed+Fish%3c%2fTitle%3e%3cDescription%3eA+red+fish+packed+in+spring+water.%3c%2fDescription%
3e%3cClientRequestId%3e12345%3c%2fClientRequestId%3e%3cCartId%3emiq%3a%2f%2frose%3a1.0%2fcar
t%2fcba%3a1.0%2fcart%3a2.0%2fAEIOU1234AEIOU%2fAEIOU1234AEIOU%2f1-1232431180394-57EEF607E6D40
DB5%3c%2fCartId%3e%3cIntegratorId%3eVWXYZ98765VWXYZ%3c%2fIntegratorId%3e%3cIntegratorName%3e
www.IntegratorWorld.com%3c%2fIntegratorName%3e%3cPrice%3e%3cAmount%3e5.0%3c%2fAmount%3e%3cCu
rrencyCode%3eUSD%3c%2fCurrencyCode%3e%3c%2fPrice%3e%3cQuantity%3e1%3c%2fQuantity%3e%3cWeight
%3e%3cAmount%3e1.0%3c%2fAmount%3e%3cUnit%3elb%3c%2fUnit%3e%3c%2fWeight%3e%3cFulfillmentNetwo
```

```
rk%3eMERCHANT%3c%2fFulfillmentNetwork%3e%3cItemCharges%3e%3cComponent%3e%3cType%3ePrincipal%
3c%2fType%3e%3cCharge%3e%3cAmount%3e5.0%3c%2fAmount%3e%3cCurrencyCode%3eUSD%3c%2fCurrencyCod
e%3e%3c%2fCharge%3e%3c%2fComponent%3e%3cComponent%3e%3cType%3eShipping%3c%2fType%3e%3cCharge
%3e%3cAmount%3e0.0%3c%2fAmount%3e%3cCurrencyCode%3eUSD%3c%2fCurrencyCode%3e%3c%2fCharge%3e%3
c%2fComponent%3e%3cComponent%3e%3cType%3eTax%3c%2fType%3e%3cCharge%3e%3cAmount%3e0.0%3c%2fAm
ount%3e%3cCurrencyCode%3eUSD%3c%2fCurrencyCode%3e%3c%2fCharge%3e%3c%2fComponent%3e%3cCompone
nt%3e%3cType%3eShippingTax%3c%2fType%3e%3cCharge%3e%3cAmount%3e0.0%3c%2fAmount%3e%3cCurrency
Code%3eUSD%3c%2fCurrencyCode%3e%3c%2fCharge%3e%3c%2fComponent%3e%3cComponent%3e%3cType%3ePri
ncipalPromo%3c%2fType%3e%3cCharge%3e%3cAmount%3e0.0%3c%2fAmount%3e%3cCurrencyCode%3eUSD%3c%2
fCurrencyCode%3e%3c%2fCharge%3e%3c%2fComponent%3e%3cComponent%3e%3cType%3eShippingPromo%3c%2
fType%3e%3cCharge%3e%3cAmount%3e0.0%3c%2fAmount%3e%3cCurrencyCode%3eUSD%3c%2fCurrencyCode%3e
%3c%2fCharge%3e%3c%2fComponent%3e%3c%2fItemCharges%3e%3c%2fProcessedOrderItem%3e%3c%2fProces
sedOrderItems%3e%3c%2fProcessedOrder%3e%3c%2fNewOrderNotification%3e
```

Here is the same information expressed as readable XML for a NewOrderNotification. Please see Appendix A – The Instant Order Processing Notification API Schema for `OrderReadyToShip` and `OrderCancelled` notifications.

```
<?xml version="1.0" encoding="UTF-8"?>
<NewOrderNotification xmlns="http://payments.amazon.com/checkout/2009-05-15/">
  <NotificationReferenceId>ae51d3a6-7843-4cbb-ad1d-ee8cc591e10d</NotificationReferenceId>
  <ProcessedOrder>
     <OrderChannel>Amazon Checkout (Live)</OrderChannel>
     <AmazonOrderID>101-1234567-9876543</AmazonOrderID>
     <OrderDate>2009-08-31</OrderDate>
     <BuyerInfo>
        <BuyerName>Kelly Green</BuyerName>
        <BuyerEmailAddress>someone@amazon.com</BuyerEmailAddress>
     </BuyerInfo>
     <ShippingAddress>
        <Name>Kelly Green</Name>
        <AddressFieldOne>123 Oak Avenue SE </AddressFieldOne>
        <AddressFieldTwo>Apt. 221-B </AddressFieldTwo>
        <City>Seattle</City>
        <State>WA</State>
        <PostalCode>98104-1234</PostalCode>
        <CountryCode>USA</CountryCode>
     </ShippingAddress>
     <ShippingServiceLevel>Standard</ShippingServiceLevel>
     <ProcessedOrderItems>
        <ProcessedOrderItem>
           <AmazonOrderItemCode>12345</AmazonOrderItemCode>
           <MerchantId>AEIOU1234AEIOU</MerchantId>
           <SKU>ABC123</SKU>
           <Title>Red Fish</Title>
           <Description>A red fish packed in spring water.</Description>
           <ClientRequestId>12345</ClientRequestId>
           <CartId>miq://rose:1.0/cart/cba:1.0/cart:2.0/
           AEIOU1234AEIOU/AEIOU1234AEIOU/1-1232431180394-57EEF607E6D40DB5</CartId>
           <IntegratorId>VWXYZ98765VWXYZ</IntegratorId>
           <IntegratorName>www.IntegratorWorld.com</IntegratorName>
           <Price>
              <Amount>5.0</Amount>
              <CurrencyCode>USD</CurrencyCode>
           </Price>
           <Quantity>1</Quantity>
           <Weight>
              <Amount>1.0</Amount>
              <Unit>lb</Unit>
           </Weight>
           <FulfillmentNetwork>MERCHANT</FulfillmentNetwork>
           <ItemCharges>
```

```
            <Component>
               <Type>Principal</Type>
               <Charge>
                  <Amount>5.0</Amount>
                  <CurrencyCode>USD</CurrencyCode>
               </Charge>
            </Component>
            <Component>
               <Type>Shipping</Type>
               <Charge>
                  <Amount>0.0</Amount>
                  <CurrencyCode>USD</CurrencyCode>
               </Charge>
            </Component>
            <Component>
               <Type>Tax</Type>
               <Charge>
                  <Amount>0.0</Amount>
                  <CurrencyCode>USD</CurrencyCode>
               </Charge>
            </Component>
            <Component>
               <Type>ShippingTax</Type>
               <Charge>
                  <Amount>0.0</Amount>
                  <CurrencyCode>USD</CurrencyCode>
               </Charge>
            </Component>
            <Component>
               <Type>PrincipalPromo</Type>
               <Charge>
                  <Amount>0.0</Amount>
                  <CurrencyCode>USD</CurrencyCode>
               </Charge>
            </Component>
            <Component>
               <Type>ShippingPromo</Type>
               <Charge>
                  <Amount>0.0</Amount>
                  <CurrencyCode>USD</CurrencyCode>
               </Charge>
            </Component>
         </ItemCharges>
      </ProcessedOrderItem>
   </ProcessedOrderItems>
</ProcessedOrder>
</NewOrderNotification>
```

The `NotificationData` contains the `NotificationReferenceId` (a unique ID generated for each Notification request), and `ProcessedOrders` (order details for the current order for which notification is sent).

The `ProcessedOrders` includes details (for example, SKU, `Title`, `Description`, `Quantity`, `Price`, and so on) from the order XML submitted from your website. The `ProcessedOrders` also includes the charges calculated at the `Item` level. The charges are `Principal` (`Item Price` multiplied by `Quantity`), Shipping (shipping charge attributed to a single `Item`), `Tax` (sales tax calculated on the `Principal` after the promotion discount is applied), `ShippingTax` (sales tax

calculated on the shipping charge after the promotion discount is applied), `PrincipalPromo`, and `ShippingPromo` (discounts applied to `Principal` and `Shipping` charges).

**Notes**

We generate a new `NotificationReferenceId` each time we send a request with a new order. When we send a request retry for the same order, we use the same `NotificationReferenceId`.

Do not ship an order until you receive the Order Ready-to-Ship notification.

#### 4.1.7 7. The AWSAccessKeyId

The AWSAccessKeyId is the other part of your access key pairs (Secret Access Key and Access Key ID). This access key is returned in the notification. For information on generating and viewing your Seller Central Secret Access Key, click the **Access Key** option from the **Integration** menu in Seller Central.

# 5 Processing the Notification Request

You use the information in the notification request (the XML content as well as the other information in the request) to perform the following actions:

1. Parse the request data.
2. Verify that the notification request is valid.
3. Process the data.
4. Respond to the request

Your application must parse this request to verify that the notification is genuinely from Amazon Payments, process the information for the merchant calculation, and prepare a response.

### 5.1.1 1. Parse the Request Data

Your first step is to parse the request data. Note that the ampersand "&" character separates the values, and that request is URL-encoded.

### 5.1.2 2. Verify that the Notification Request Is Valid

After you receive the data and parse it, you should verify that the request is valid by verifying the Signature and the Timestamp. Verifying the data helps to prevent responding to requests that might incorrectly claim to be from Amazon Payments.

> **Notes**
>
> You can skip this step if you did not specify in Seller Central that you accept only signed carts.
>
> We recommend you accept only signed carts to prevent spoofed or invalid orders.

#### *5.1.2.1 2a. Verify the Signature*

To verify that the request is genuine, you compare the `Signature` we generate with the `Signature` you generate. To calculate the `Signature`, you decode the URL-encoded `UUID` and `Timestamp` from the original request; concatenate the `UUID` and `Timestamp`; then use a standard HMAC SHA1 algorithm and your Seller Central Secret Access Key (your private key). For information on generating and viewing your Seller Central Secret Access Key, click the **Access Key** option from the **Integration** menu in Seller Central.

> **Notes**
>
> Be sure you use the Seller Central Secret Access Key to validate and generate signatures for requests and responses – this is the key you find in Seller Central on the **Integration** > **MWS Access Key** page.

#### *5.1.2.2 2b. Verify the Timestamp*

To avoid replay attacks, compare the `Timestamp` in the notification you receive with your system clock. If the difference is more than 15 minutes, do not process the notification.

**Notes**

We recommend that you synchronize your system clock with a standard clock, such as the NIST clock (http://www.time.gov/). If you system clock is not synchronized, you might drop valid requests.

We recommend you record the notifications you receive so that you can compare the `UUID` and `Timestamp` to verify the uniqueness of the notification.

After you verify the `Timestamp`, you can proceed with processing the XML document.

#### 5.1.3 3. Process the Data

Now that you have the notification data, you can log it, pass it to inventory management, and use it for your internal order fulfillment systems.

If you receive an `OrderReadyToShip` notification, you must confirm shipment with Amazon.

#### 5.1.4 4. Respond to the Request

For successful notification reception, Pay with Amazon expects the HTTP response status code "200 OK." Other responses result in retries of the notification request.

# 6 Using Custom Data Fields

You can use your own data definitions and data fields in your Order XML feed. For example, if you want to allow customers to personalize the mugs and cups you sell, you can pass the custom text field "`cust_name`" in your Order XML. When we send the notification to you, we will include this information.

## 6.1 Prerequisites for Using Custom Data Fields

To use custom data fields in your feeds, your XML-based applications must meet the following conditions:

1. You must be using the XMLNS named "**http://payments.amazon.com**/checkout/2009-05-15/" or later.
2. You must be using the `order.xsd` version with a file date of 2009-03-04 or later. (Download the latest version here.)
3. You must be using the `iopn.xsd` version with a file date of 2009-03-04 or later. (Download the latest version here.)

## 6.2 How We Process the Custom Data Fields

When we receive your order XML, your order notification response, or your callback response which contains your custom data, we simply store it until we return it to you as part of the Instant Order Processing Notification.

We do not parse the data or attempt to interpret it. As long as the content is syntactically correct (tags spelled the same for both opening and closing tags, no tags improperly nested, and so on), we accept the content as-is, and return it to you in the same way.

Because we do not parse the custom data you send that's marked with the `ItemCustomData` or `CartCustomData` tags, you do not need to send us an XSD or otherwise define your custom data fields.

## 6.3 Modifying Your Order XML to Use Custom Data Fields

To use custom data fields in your order, you add a new section in your order XML tagged with `ItemCustomData` or `CartCustomData`, as shown in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<Order xmlns="http://payments.amazon.com/checkout/2009-05-15/">
  <Cart>
     <Items>
        <Item>
           <SKU>JKL909</SKU>
           <MerchantId>AEIOU1234AEIOU</MerchantId>
           <Title>Calvin and Hobbes Reliquary</Title>
           <Price>
              <Amount>29.99</Amount>
              <CurrencyCode>USD</CurrencyCode>
           </Price>
```

```
            <Quantity>1</Quantity>
            <Weight>
               <Amount>8.5</Amount>
               <Unit>lb</Unit>
            </Weight>
            <Category>Books</Category>
            <ItemCustomData>
               <CustomText>
                  <Text>Charles River Sports Academy</Text>
                  <Color>Malachite</Color>
                  <StyleID>10090</StyleID>
               </CustomText>
            </ItemCustomData>
         </Item>
      </Items>
      <CartCustomData>
         <CartNumber>0525470948</CartNumber>
      </CartCustomData>
   </Cart>
...
</Order>
```

## 6.4 Using Custom Data Fields with the Instant Order Processing Notification API

To use custom data fields with order notifications, you enter the information (as shown above) in the `ItemCustomData` or `CartCustomData` areas.

When we send you the order notification XML, we include this information in the file, as shown in the example below:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NewOrderNotification xmlns="http://payments.amazon.com/checkout/2009-05-15/">
   <NotificationReferenceId>6e5fdbca-e5c1-4353-9942-87210203378d</NotificationReferenceId>
   <ProcessedOrder>
      <OrderChannel>Amazon Checkout</OrderChannel>
      <AmazonOrderID>102-4788713-2074908</AmazonOrderID>
      <OrderDate>2009-09-11T02:19:01.000Z</OrderDate>
      <BuyerInfo>
         <BuyerName>KC Brown</BuyerName>
         <BuyerEmailAddress>someone@amazon.com</BuyerEmailAddress>
      </BuyerInfo>
      <ShippingAddress>
         <Name>KC Brown</Name>
         <AddressFieldOne>22 Twain Ave. South</AddressFieldOne>
         <City>Northampton</City>
         <State>MA</State>
         <PostalCode>01065</PostalCode>
         <CountryCode>US</CountryCode>
      </ShippingAddress>
      <ShippingServiceLevel>Standard</ShippingServiceLevel>
      <ProcessedOrderItems>
         <ProcessedOrderItem>
            <AmazonOrderItemCode>44837685294413</AmazonOrderItemCode>
            <MerchantId>AEIOU1234AEIOU</MerchantId>
            <SKU>JKL909</SKU>
            <Title>Calvin and Hobbes Reliquary</Title>
            <Description>By Bill Watterson</Description>
            <ClientRequestId>123457</ClientRequestId>
            <CartId>miq://rose:1.0/cart/cba:1.0/cart:2.0/
               AEIOU1234AEIOU/VWXYZ98765VWXYZ/
               2-1328561625299-323F62C3E26F95E9</CartId>
```

```
            <Price>
               <Amount>29.99</Amount>
               <CurrencyCode>USD</CurrencyCode>
            </Price>
            <Quantity>1</Quantity>
            <Weight>
               <Amount>8.5</Amount>
               <Unit>lb</Unit>
            </Weight>
            <Category>Books</Category>
            <FulfillmentNetwork>MERCHANT</FulfillmentNetwork>
            <ItemCharges>
               <Component>
                  <Type>Principal</Type>
                  <Charge>
                     <Amount>2.5</Amount>
                     <CurrencyCode>USD</CurrencyCode>
                  </Charge>
               </Component>
               <Component>
                  ...
               </Component>
            </ItemCharges>
            <CartCustomData>
               <CartNumber>0525470948</CartNumber>
            </CartCustomData>
            <ItemCustomData>
               <CustomText>
                  <Text>Charles River Sports Academy</Text>
                  <Color>Malachite</Color>
                  <StyleID>10090</StyleID>
               </CustomText>
            </ItemCustomData>
         </ProcessedOrderItem>
      </ProcessedOrderItems>
   </ProcessedOrder>
</NewOrderNotification>
```

When you receive this data, you can manipulate it using your own application logic.

## 6.5 Using a Custom Namespace as Part of the Custom Data Entry

You can provide your own namespace as part of your custom data entry.

Note the added XMLNS in the example below, listed as `CustomText xmlns="http://mydomain.com"`:

```
<?xml version="1.0" encoding="UTF-8"?>
<Order xmlns="http://payments.amazon.com/checkout/2009-05-15/">
   <Cart>
      <Items>
         <Item>
            <SKU>JKL909</SKU>
            <MerchantId>AEIOU1234AEIOU</MerchantId>
            <Title>Calvin and Hobbes Reliquary</Title>
            <Price>
               <Amount>29.99</Amount>
               <CurrencyCode>USD</CurrencyCode>
            </Price>
            <Quantity>1</Quantity>
            <Weight>
```

```
            <Amount>8.5</Amount>
            <Unit>lb</Unit>
         </Weight>
         <Category>Books</Category>
         <ItemCustomData>
            <mynamespace:CustomText xmlns:mynamespace="http://mydomain.com">
               <mynamespace:Text>Charles River Sports Academy</mynamespace:Text>
               <mynamespace:Color>Malachite</mynamespace:Color>
               <mynamespace:StyleID>10090</mynamespace:StyleID>
            </mynamespace:CustomText>
         </ItemCustomData>
      </Item>
   </Items>
   <CartCustomData>
      <CartNumber>0525470948</CartNumber>
      Customer <CName>John Smith</CName> wants delivery only in the evenings
   </CartCustomData>
</Cart>
...
</Order>
```

Because you added this custom XMLNS, the responses are modified, too.

Here's an example of the resulting order notification:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NewOrderNotification xmlns="http://payments.amazon.com/checkout/2009-05-15/">
  <NotificationReferenceId>6e5fdbca-e5c1-4353-9942-87210203378d</NotificationReferenceId>
  <ProcessedOrder>
     <OrderChannel>Amazon Checkout</OrderChannel>
     <AmazonOrderID>102-4788713-2074908</AmazonOrderID>
     <OrderDate>2009-09-11T02:19:01.000Z</OrderDate>
     <BuyerInfo>
        <BuyerName>KC Brown</BuyerName>
        <BuyerEmailAddress>someone@amazon.com</BuyerEmailAddress>
     </BuyerInfo>
     <ShippingAddress>
        <Name>KC Brown</Name>
        <AddressFieldOne>22 Twain Ave. South</AddressFieldOne>
        <City>Northampton</City>
        <State>MA</State>
        <PostalCode>01065</PostalCode>
        <CountryCode>US</CountryCode>
     </ShippingAddress>
     <ShippingServiceLevel>Standard</ShippingServiceLevel>
     <ProcessedOrderItems>
        <ProcessedOrderItem>
           <AmazonOrderItemCode>44837685294413</AmazonOrderItemCode>
           <MerchantId>AEIOU1234AEIOU</MerchantId>
           <SKU>JKL909</SKU>
           <Title>Calvin and Hobbes Reliquary</Title>
           <Description>By Bill Watterson</Description>
           <ClientRequestId>123457</ClientRequestId>
           <CartId>miq://rose:1.0/cart/cba:1.0/cart:2.0/
              AEIOU1234AEIOU/VWXYZ98765VWXYZ/
              2-1328561625299-323F62C3E26F95E9</CartId>
           <Price>
              <Amount>29.99</Amount>
              <CurrencyCode>USD</CurrencyCode>
           </Price>
           <Quantity>1</Quantity>
           <Weight>
```

```
            <Amount>8.5</Amount>
            <Unit>lb</Unit>
         </Weight>
         <Category>Books</Category>
         <FulfillmentNetwork>MERCHANT</FulfillmentNetwork>
         <ItemCharges>
            <Component>
               <Type>Principal</Type>
               <Charge>
                  <Amount>2.5</Amount>
                  <CurrencyCode>USD</CurrencyCode>
               </Charge>
            </Component>
            <Component>
               ...
            </Component>
         </ItemCharges>
         <CartCustomData>
            <CartNumber>0525470948</CartNumber>
            Customer <CName>John Smith</CName> wants delivery only in the evenings
         </CartCustomData>
         <ItemCustomData>
            <mynamespace:CustomText xmlns:mynamespace="http://mydomain.com">
               <mynamespace:Text>Charles River Sports Academy</mynamespace:Text>
               <mynamespace:Color>Malachite</mynamespace:Color>
               <mynamespace:StyleID>10090</mynamespace:StyleID>
            </mynamespace:CustomText>
         </ItemCustomData>
      </ProcessedOrderItem>
   </ProcessedOrderItems>
</ProcessedOrder>
</NewOrderNotification>
```

# 7 Appendix A—the Instant Order Processing Notification API Schema

Here is the schema for Instant Order Processing Notification API. You can also download the http://amazonpayments.s3.amazonaws.com/documents/iopn.xsd file.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://payments.amazon.com/checkout/2008-11-30/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://payments.amazon.com/checkout/2008-11-30/"
elementFormDefault="qualified">

  <!-- IOPN Events -->
  <xs:element name="NewOrderNotification" type="tns:NewOrderNotification"/>
  <xs:element name="OrderReadyToShipNotification" type="tns:OrderReadyToShipNotification"/>
  <xs:element name="OrderCancelledNotification" type="tns:OrderCancelledNotification"/>

  <!-- Basic types -->
  <xs:simpleType name="ShippingServiceLevel">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Standard"/>
      <xs:enumeration value="Expedited"/>
      <xs:enumeration value="OneDay"/>
      <xs:enumeration value="TwoDay"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="NonNegativeDouble">
    <xs:restriction base="xs:double">
      <xs:minInclusive value="0"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="PositiveDouble">
    <xs:restriction base="xs:double">
      <xs:minExclusive value="0"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="Price">
    <xs:sequence>
      <xs:element name="Amount" type="tns:NonNegativeDouble"/>
      <xs:element name="CurrencyCode" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="Weight">
    <xs:sequence>
      <xs:element name="Amount" type="tns:NonNegativeDouble"/>
      <xs:element name="Unit" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="FulfillmentNetwork">
    <xs:restriction base="xs:string">
      <xs:enumeration value="MERCHANT"/>
      <xs:enumeration value="AMAZON_NA"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="ShippingAddress">
    <xs:sequence>
```

```
      <xs:element name="Name" type="xs:string" minOccurs="0"/>
      <xs:element name="AddressFieldOne" type="xs:string" minOccurs="0"/>
      <xs:element name="AddressFieldTwo" type="xs:string" minOccurs="0"/>
      <xs:element name="AddressFieldThree" type="xs:string" minOccurs="0"/>
      <xs:element name="City" type="xs:string" minOccurs="0"/>
      <xs:element name="State" type="xs:string" minOccurs="0"/>
      <xs:element name="PostalCode" type="xs:string" minOccurs="0"/>
      <xs:element name="CountryCode" type="xs:string" minOccurs="0"/>
      <xs:element name="PhoneNumber" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="BillingAddress">
    <xs:sequence>
      <xs:element name="Name" type="xs:string" minOccurs="0"/>
      <xs:element name="AddressFieldOne" type="xs:string" minOccurs="0"/>
      <xs:element name="AddressFieldTwo" type="xs:string" minOccurs="0"/>
     <xs:element name="AddressFieldThree" type="xs:string" minOccurs="0"/>
      <xs:element name="City" type="xs:string" minOccurs="0"/>
      <xs:element name="State" type="xs:string" minOccurs="0"/>
      <xs:element name="PostalCode" type="xs:string" minOccurs="0"/>
      <xs:element name="CountryCode" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- IOPN Event Contents -->
  <xs:complexType name="NewOrderNotification">
    <xs:sequence>
      <xs:element name="NotificationReferenceId" type="xs:string"/>
      <xs:element name="ProcessedOrder" type="tns:ProcessedOrder"/>
      <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="OrderReadyToShipNotification">
    <xs:sequence>
      <xs:element name="NotificationReferenceId" type="xs:string"/>
      <xs:element name="ProcessedOrder" type="tns:ProcessedOrder"/>
      <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="OrderCancelledNotification">
    <xs:sequence>
      <xs:element name="NotificationReferenceId" type="xs:string"/>
      <xs:element name="ProcessedOrder" type="tns:ProcessedOrder"/>
      <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <!-- Common Types -->
  <xs:complexType name="ProcessedOrder">
    <xs:sequence>
      <xs:element name="OrderChannel" type="xs:string"/>
      <xs:element name="AmazonOrderID" type="xs:string"/>
      <xs:element name="OrderDate" type="xs:dateTime"/>
      <xs:element name="BuyerInfo">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="BuyerName" type="xs:string"/>
            <xs:element name="BuyerEmailAddress" type="xs:string"/>
            <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
```

```
        </xs:complexType>
      </xs:element>
      <xs:element name="BillingAddress" type="tns:BillingAddress" minOccurs="0"/>
      <xs:element name="ShippingAddress" type="tns:ShippingAddress"/>
      <xs:element name="ShippingServiceLevel" type="tns:ShippingServiceLevel"
minOccurs="0"/>
      <xs:element name="ProcessedOrderItems" type="tns:ProcessedOrderItems"/>
      <xs:element name="DisplayableShippingLabel" type="xs:string"/>
      <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="ProcessedOrderItems">
    <xs:sequence>
      <xs:element name="ProcessedOrderItem" type="tns:ProcessedOrderItem"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="ProcessedOrderItem">
    <xs:sequence>
      <xs:element name="AmazonOrderItemCode" type="xs:string"/>
      <xs:element name="MerchantId" type="xs:string"/>
      <xs:element name="SKU" type="xs:string" minOccurs="0"/>
      <xs:element name="Title" type="xs:string"/>
      <xs:element name="Description" type="xs:string" minOccurs="0"/>
      <xs:element name="ClientRequestId" type="xs:string" minOccurs="0"/>
      <xs:element name="CartId" type="xs:string" minOccurs="0"/>
      <xs:element name="IntegratorId" type="xs:string" minOccurs="0"/>
      <xs:element name="IntegratorName" type="xs:string" minOccurs="0"/>
      <xs:element name="Price" type="tns:Price"/>
      <xs:element name="Quantity" type="xs:nonNegativeInteger" minOccurs="0"/>
      <xs:element name="Weight" type="tns:Weight" minOccurs="0"/>
      <xs:element name="Category" type="xs:string" minOccurs="0"/>
      <xs:element name="Condition" type="xs:string" minOccurs="0"/>
      <xs:element name="FulfillmentNetwork" type="tns:FulfillmentNetwork" minOccurs="0"/>
      <xs:element name="ItemCharges" type="tns:Charges" minOccurs="0"/>
      <xs:element name="CartCustomData" type="tns:MerchantCustomDataXML" minOccurs="0"/>
      <xs:element name="ItemCustomData" type="tns:MerchantCustomDataXML" minOccurs="0"/>
      <xs:element name="ShippingCustomData" type="tns:MerchantCustomDataXML" minOccurs="0"/>

     <!--
        Note: The element below is included for future enhancements. This field is not
        populated in the current version of IOPN.
      -->
      <xs:element name="ItemAttributes" type="tns:ItemAttributes" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="Charges">
    <xs:sequence>
      <xs:element name="Component" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Type" type="tns:ComponentType"/>
            <xs:element name="Charge" type="tns:Price"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  <!-- In the current version, Giftwrap and Other charge types are not supported.
       All amounts are positive. The following mathematical formula can be used to
```

```
       calculate the total item charge:

       Total Item Charge = (Principal - PrincipalPromo) + (Shipping - ShippingPromo) + Tax +
ShippingTax
   -->
  <xs:simpleType name="ComponentType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Principal"/>
      <xs:enumeration value="Shipping"/>
      <xs:enumeration value="GiftWrap"/>
      <xs:enumeration value="Other"/>
      <xs:enumeration value="Tax"/>
      <xs:enumeration value="ShippingTax"/>
      <xs:enumeration value="GiftWrapTax"/>
      <xs:enumeration value="OtherTax"/>
      <xs:enumeration value="PrincipalPromo"/>
      <xs:enumeration value="ShippingPromo"/>
      <xs:enumeration value="GiftWrapPromo"/>
      <xs:enumeration value="OtherPromo"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- Item Attributes is a reserved type for future extensions. New information about order
items
       will be included in this structure.
   -->
  <xs:complexType name="ItemAttributes">
    <xs:sequence>
      <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <!-- This is a reserved type and will contain any type of private data that is passed from
the Seller
       to Amazon along with the cart.
   -->
  <xs:complexType name="MerchantCustomDataXML" mixed="true">
    <xs:sequence>
      <xs:any namespace="##any" processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

</xs:schema>
```

# 8 Appendix B—Frequently Asked Questions

## 8.1 What is the Instant Order Processing Notification API?

With the Instant Order Processing Notification API, you can get instant updates when you get a new new orders; you can reserve your inventory (remove it from availability to be sold) as soon as you get an Order Ready-to-Ship notification; and you can release your inventory (make it available to be sold) when you receive an Order Canceled notification.

## 8.2 What are order notifications?

Order notifications are HTTP requests that inform merchants about new orders or status changes in existing orders placed using Pay with Amazon. These notifications can be passed to merchant's internal order management system (OMS).

## 8.3 How does the Instant Order Processing Notification API benefit me?

Instead of going to Seller Central to see order status changes, you can use Instant Order Processing Notification API to receive instant updates and then automatically process them using your order management system (OMS). With the Instant Order Processing Notification API, you can be instantly notified of new orders; you can reserve your inventory (remove it from availability to be sold) as soon as you get an Order Ready-to-Ship notification; and you can release your inventory (make it available to be sold) when you receive an Order Canceled notification.

## 8.4 What are the prerequisites for using the Instant Order Processing Notification API?

Before you can use the Instant Order Processing Notification API, your systems must meet the following prerequisites.

1. You must be using the `order.xsd` version 2009-05-15 or later. (Download this version here.)
2. You must be using the `iopn.xsd` version 2008-11-30 or later. (Download this version here.)
3. You must be running a web service that can receive HTTPS POST requests made to your endpoint using port 443.

To learn more about enabling Instant Order Processing Notification API, please see Enabling Order Notifications, above.

## 8.5 Where are Instant Order Processing Notification API messages sent?

Instant Order Processing Notification API messages go to the Merchant URL or Integrator URL endpoint you specified in Seller Central.

The Merchant URL is the URL on your website where you (the merchant) want to receive notifications as the orders are processed. Your web service must read and process notifications at this URL. If you leave the Merchant URL blank, you (the merchant) will not receive notifications.

If you provide the Integrator URL, the notification will go to that URL.

You must enable at least one URL endpoint, whether it is the Merchant URL or the Integrator URL. You set the endpoint URLs in Seller Central.

## 8.6 What messages can I receive and when will I receive them?

You can receive three type of notifications:

1. `NewOrderNotification` You receive this notification when a new order is placed from your site. This notification means that the Amazon Payments internal order processing has started.
2. `OrderReadyToShipNotification` You receive this notification when the order is ready to be shipped according to Pay with Amazon order pipeline.
3. `OrderCancelledNotification` You get this notification when the order is canceled during the order processing, whether by the buyer or the merchant. You also receive this notification if Amazon Payments cancels the order due to buyer fraud, order expiration, failed authorization, and so on.

## 8.7 Why am I not receiving an Instant Order Processing Notification API message?

To enable Instant Order Processing Notification API messages, you must provide the endpoint URL in Seller Central. Please see Enabling Order Notifications, above.

If you provide the Merchant URL (endpoint), verify that the URL is using `https`:// for Production accounts, using port 443. You can use either `http://` or `https://` for Sandbox accounts.

If you provide the integrator URL (endpoint), please verify that URL is using `https`:// for Production accounts, using port 443. You can use either `http`:// or `https`:// for Sandbox accounts.

## 8.8 Why did I receive an Order Canceled Notification or Order Ready-to-Ship notification before the New Order notification for the same order?

If the endpoint is unavailable when we send the initial New Order notification but is available soon after when we send an Order Canceled notification or Order Ready-to-Ship notification, then the notifications are unsynchronized.

If this happens you can follow one of these procedures:

1. Do not acknowledge the Order Canceled or Order Ready-To-Ship notification until you receive the New Order notification message. Instead, send an INTERNAL_ERROR (500) message, as the message is out of sequence. If you do not acknowledge a notification, Amazon Payments keeps retrying the notification request; notifications will synchronize after you get the New Order notification.
2. Because each notification contains the full cart and order detail, you can update your order management system (OMS) with the order information sent in the Order Canceled or Order Ready-To-Ship notification and ignore the New Order notification message you receive later.

## 8.9 Why did I receive an Order Canceled notification before an Order Ready-to-Ship notification for the same order?

If endpoint is unavailable when we send the initial Order Ready-to-Ship notification but is available soon after when we send an Order Canceled notification, then the notifications are unsynchronized.

If it happens, please ignore the `OrderReadyToShip` Notification as `OrderCancelled` notification is the final state for an order.

## 8.10 What do I do if I get a notification request with a duplicate UUID?

The `UUID` is unique for each request. If you get a second notification with a duplicate `UUID`, please ignore it as the notification might be due to external replay attacks.

Note that the `NotificationReferenceID` is the same for each retried notification, even though we create a new `UUID`.

## 8.11 What happens if my service or endpoint is unavailable?

If the notification fails because the endpoint URL is unavailable or the merchant discards the request, Pay with Amazon follows a 14-days Exponential Back-off Retry Policy.

## 8.12 What is the 14-day Exponential Back-off Retry Policy?

If the notification fails because the merchant endpoint URL is unavailable or the merchant discards the request, Pay with Amazon follows a 14-day Exponential Back-off Retry Policy, where the time interval between two retries doubles after every retry. For example, we make the first retry after one minute. We make the second retry after 2 minutes, the third after 4 minutes, and so on. We retry sending these notification for 14 days. If the request does not get through in 14 days, the request is dropped.

## 8.13 What should I do if I cannot process the notification I receive?

If you cannot process the received notification, you must respond with proper error codes, as listed below:

| Name | Use | W3C Definition |
|---|---|---|
| **INTERNAL_ERROR (500)** | Respond with this code when the internal service handling the notification cannot process the request. | The server encountered an unexpected condition which prevented it from fulfilling the request. |
| **SERVICE_UNAVAILABLE (503)** | Respond with this code when your internal service cannot process the request or when you want Amazon Payments to retry sending the notification. Retries are sent following the Exponential Back-off Retry Policy. | The server is currently unable to handle the request due to a temporary overloading or maintenance of the server. |
| **PERMISSION_DENIED (403)** | Respond with this code when you are not able to verify the signature | The server understood the request, but is refusing to fulfill it. |

*Definitions from Hypertext Transfer Protocol – HTTP/1.1 definitions (RFC2616 Fielding, et al.).*

## 8.14 How do I authenticate order notifications?

After you receive the data and parse it, you should verify that the request is valid by verifying the Signature and the Timestamp. Please see Verify that the Notification Request Is Valid, above.

## 8.15 Why did I receive an order notification with items referring to multiple carts?

Amazon consolidates 1-Click orders that are placed within 90-minute window into a single order. When consolidation occurs, you receive a single notification of the consolidated order. The items in the consolidated order originate from the multiple 1-Click order submissions. Each item, therefore, refers to its originating cart.

## 8.16 Why did I receive an order notification with fewer items than were in the submitted cart?

If the submitted cart contains items with different shipping method or if merchant-fulfilled items are combined with Fulfillment by Amazon items, the cart is split into multiple Amazon orders. When this occurs, you receive a separate notification for each Amazon order. Each notification includes the different set of items (that is, items grouped based on the common shipping method) from the same cart.

## 8.17 I received a New Order Notification. Should I ship the order now?

No, do not ship the order when you receive the New Order notification. This notification lets you know that the order has been placed, but payment for the order might still be processing. Wait until you receive the Order Ready-to-Ship notification before you ship the order.