Pay with Amazon Integration Guide

Contents

Pay with Amazon Integration Guide

Last updated: 2014-08-19

Introduction to Pay with Amazon

Pay with Amazon provides millions of Amazon buyers a secure, trusted, and convenient way to pay for their purchases on your site. Buyers simply select a shipping address and payment method stored in their Amazon account to complete their purchase.

There are two ways to set up Pay with Amazon on your site:

- Checkout by Amazon: Activate a complete checkout flow on your site with a simple implementation. Includes a hosted checkout experience and integrated order management features.
- Advanced Payment APIs: Activate a pure payment solution into your own site using a complete set of flexible APIs.

You can learn more about these options on the UK Amazon Payments web site , or the DE Amazon Payments web site

This guide provides details on the API-based integration, called Advanced Payment APIs, which provides a greater degree of flexibility and customization to meet your payment needs.

Before you start - Important Information

This document aims at guiding you through the Integration of the Amazon Advanced Payment APIs regardless of the programming language that you are working with. However, for the most commonly used programming languages (C#, Java, and PHP) we provide SDKs including libraries and examples that will significantly reduce the complexity of your integration. Our libraries have been tested extensively and thus are very reliable. In consequence, we strongly recommend you to utilize our SDKs if your programming language is supported.

- PHP Client Library
- C# Client Library
- Java Client Library

In case you use one of these languages, please have a look at the corresponding SDK, download it and read the included documentation, especially the readme file. In addition you should also be aware of the Off-Amazon Payments API section and of the general Amazon MWS Developer Guide

Important Advanced Payment APIs prerequisites

The following are the prerequisites for the Advanced Payment APIs service:

- Previous experience using HTML, JavaScript, and XML for application development.
- Previous experience using web service APIs.
- Familiarity with the Seller Central portal.
- A Seller ID (also called a Merchant ID) for an Amazon Seller account. The Amazon Seller account must have the Advanced Payment APIs service provisioned. You can create an Amazon Seller account or provision an existing Seller account for Amazon Payments by signing up for Advanced Payment APIs from the Amazon Payments website (UK Amazon Payments web site, or DE Amazon Payments web site).
- Access to your Amazon Marketplace Web Service (Amazon MWS) Access Key and Secret Key. To get these keys, register with Amazon MWS or authorize a developer to make calls on your behalf using their keys. For more information, see "Registering to use Amazon MWS" in the Amazon MWS Developer Guide.
- The buyer’s browser must have JavaScript enabled.
- The buyer’s browser must have cookies enabled.

How does Pay with Amazon work?

Pay with Amazon is embedded directly into your existing web site, and all the buyer interactions with Pay with Amazon take place in embedded widgets so that the buyer never leaves your site. Buyers can log in using their Amazon account, select a shipping address and payment method, and then confirm their order.

Key concepts in Pay with Amazon

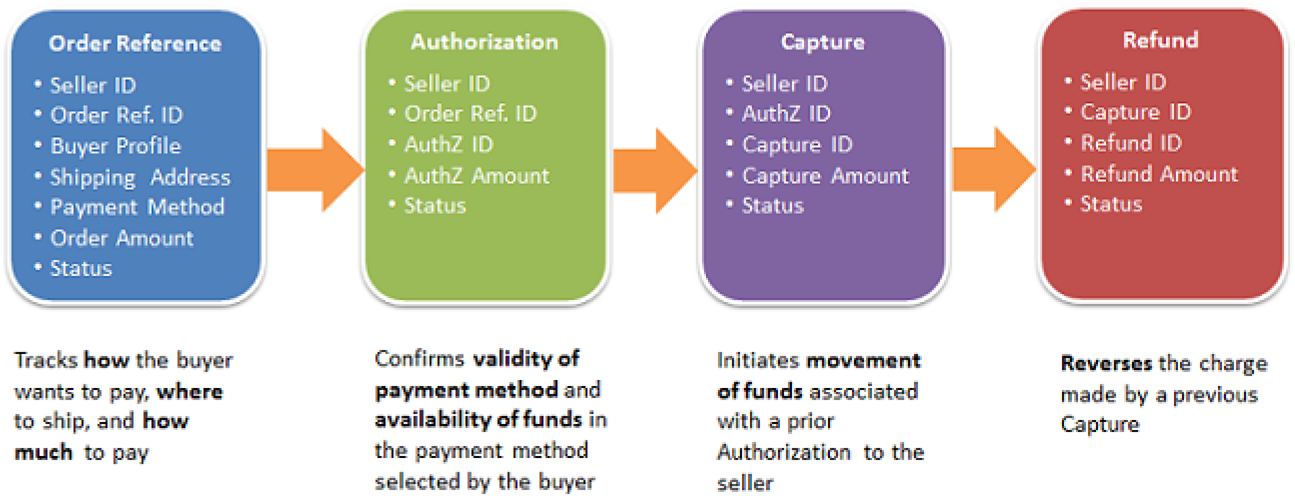
The end-to-end Pay with Amazon experience is designed to make checkout easy and secure for buyers, and payment processing easy and secure for you. Before you start Advanced Payment APIs integration, you should familiarize yourself with a few key concepts regarding this service.

Payment objects

Payment objects are the building blocks that support the Advanced Payment APIs payments process. The following objects are referenced throughout this guide:

- Order Reference object – This object is a record of key attributes necessary to process the buyer's payment. Some of these attributes will be determined by the buyer as they go through the checkout process (for example, shipping address and payment method) and you will set some of these attributes with information about the order (for example, the total amount and your order identifier).
- Authorization object – This object represents the availability of funds in the selected payment method and reserves them for future collection using a `Capture` request.
- Capture object – This object represents the movement of funds from the buyer's payment method to your Amazon Payments Seller account.
- Refund object – In the event of a refund, this object represents the movement of previously captured funds back to the buyer's account.

Each of these objects is identified by a unique identifier and has a status. You will learn more about using these objects in the next chapter of this guide.

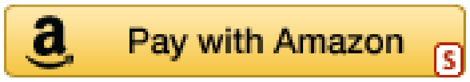


Widgets

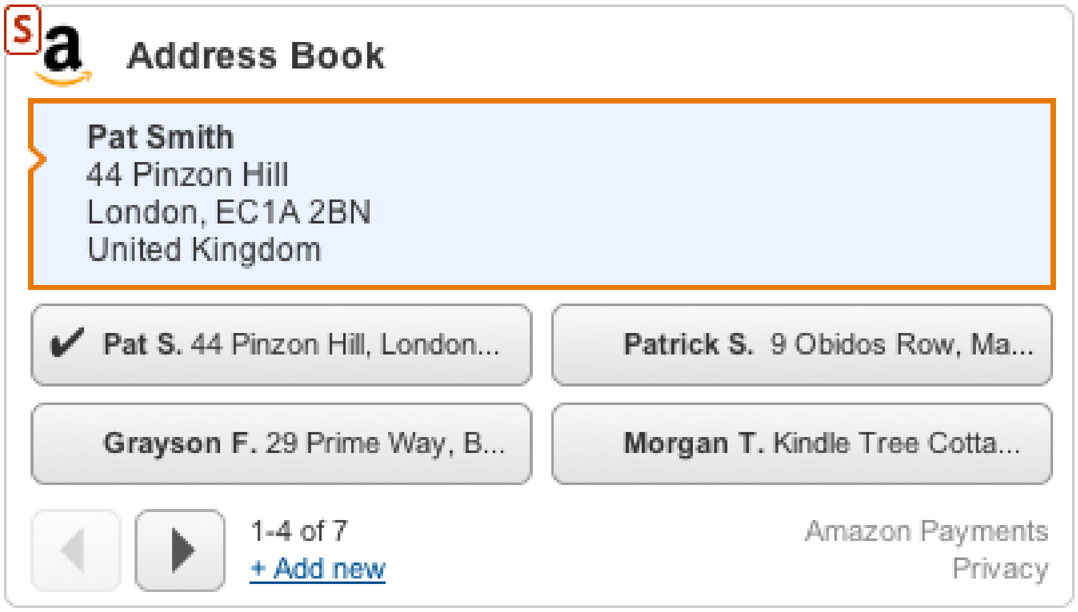
Widgets are Amazon-hosted controls that you can embed in your site. They allow the buyer to authenticate using their Amazon credentials, view the address and payment methods stored in their Amazon account, and select the ones that they want to use for their purchase. The selected address and payment method information is stored in the Order Reference object.

There are three widgets that you will use when integrating with Advanced Payment APIs:

- **Button** widget: This enables a buyer to choose Amazon Payments for their payment and to authenticate with Amazon. You typically place this on the page where a buyer initiates the checkout process, for example from your shopping cart or product detail page. The following image shows the **Button** widget:



- **AddressBook** widget: The **AddressBook** widget displays the addresses stored in a buyer's Amazon account. You can replace the manual entry forms that you use to collect addresses from the buyer with this widget. The following image shows the **AddressBook** widget:


Address Book

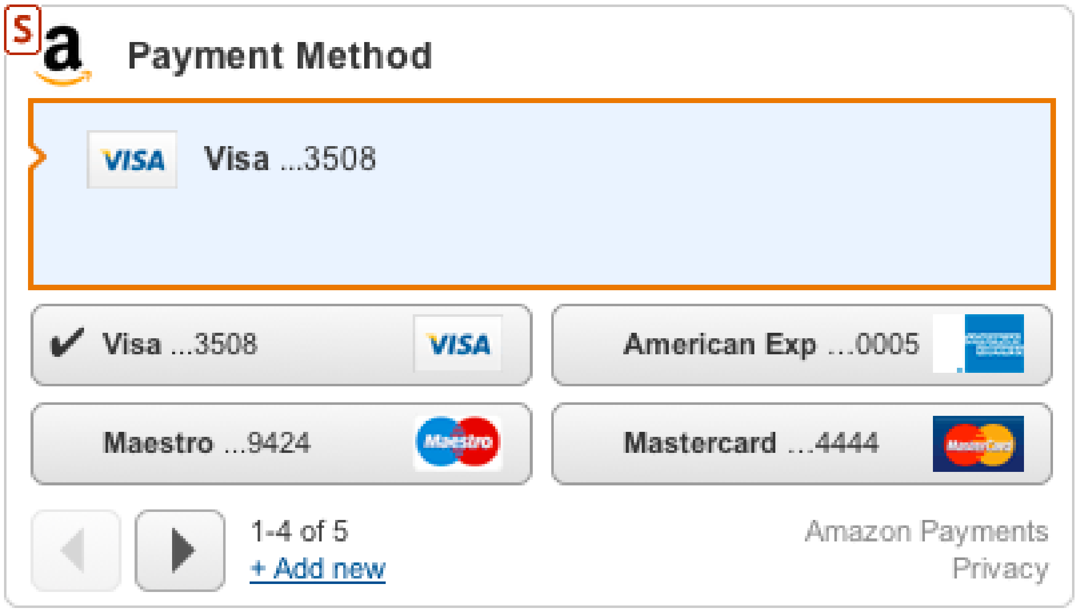
Pat Smith
44 Pinzon Hill
London, EC1A 2BN
United Kingdom

✓ Pat S. 44 Pinzon Hill, London...
Patrick S. 9 Obidos Row, Ma...
Grayson F. 29 Prime Way, B...
Morgan T. Kindle Tree Cotta...

1-4 of 7
+ Add new

Amazon Payments
Privacy

- **Wallet** widget: The **Wallet** widget displays the payment methods stored in a buyer's Amazon account. You can replace the manual entry forms that you use to collect payment method information from the buyer with this widget. The following image shows the **Wallet** widget:


Payment Method

Visa ...3508

✓ Visa ...3508
American Exp ...0005
Maestro ...9424
Mastercard ...4444

1-4 of 5
+ Add new

Amazon Payments
Privacy

Application Programming Interfaces (APIs)

You call the operations in the Off-Amazon Payments API section to exchange information and instructions between Amazon Payments and your internal systems. For example, you call these operations to request Amazon to charge the buyer, issue a refund, get the buyer's shipping information, or cancel an order reference. For more information, see the Off-Amazon Payments API section reference.

Instant Payment Notifications (IPNs)

By default, Amazon processes your payment requests (`Authorize`, `Capture`, and `Refund` requests) in an asynchronous manner. Whether using asynchronous or synchronous processing, after processing the request, Amazon sends a notification, called an Instant Payment Notification (IPN), which notifies you of the final status of the request.

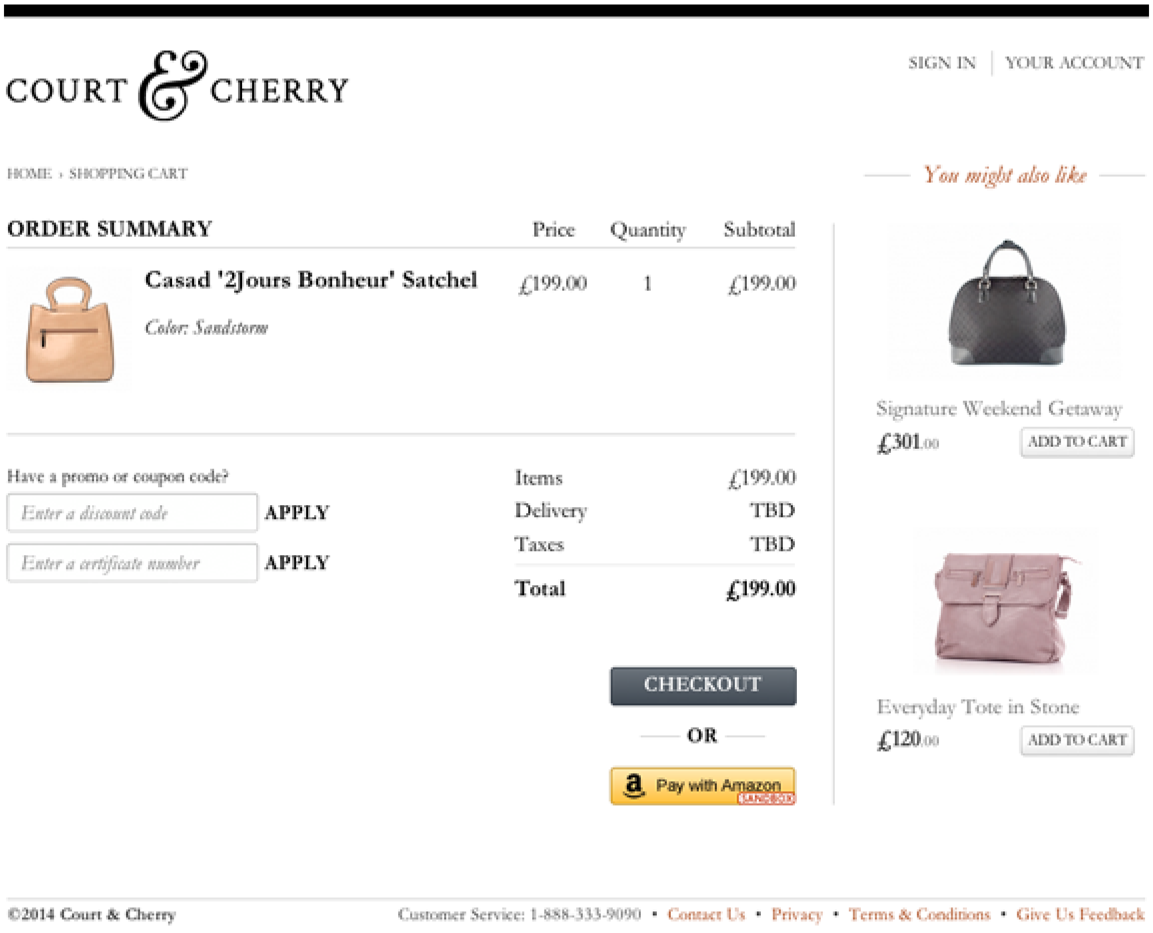
Additionally, the status of a payment object can change due to a request submitted by you or due to an internal Amazon business rule. In this case, Amazon will send an IPN to you so that you can keep your system in sync with Amazon Payments. For more information, see Synchronizing your systems with Amazon Payments.

Getting started

Overview of the buyer experience with Pay with Amazon

Pay with Amazon enables a buyer to complete a purchase on your site in just a few clicks. Following is an example of what the buyer experience with Amazon Payments looks like:

1. The buyer begins the checkout process by clicking the **Pay with Amazon** button.



ORDER SUMMARY

	Price	Quantity	Subtotal
Casad '2Jours Bonheur' Satchel *Color: Sandstorm*	£199.00	1	£199.00

Have a promo or coupon code?

Enter a discount code APPLY

Enter a certificate number APPLY

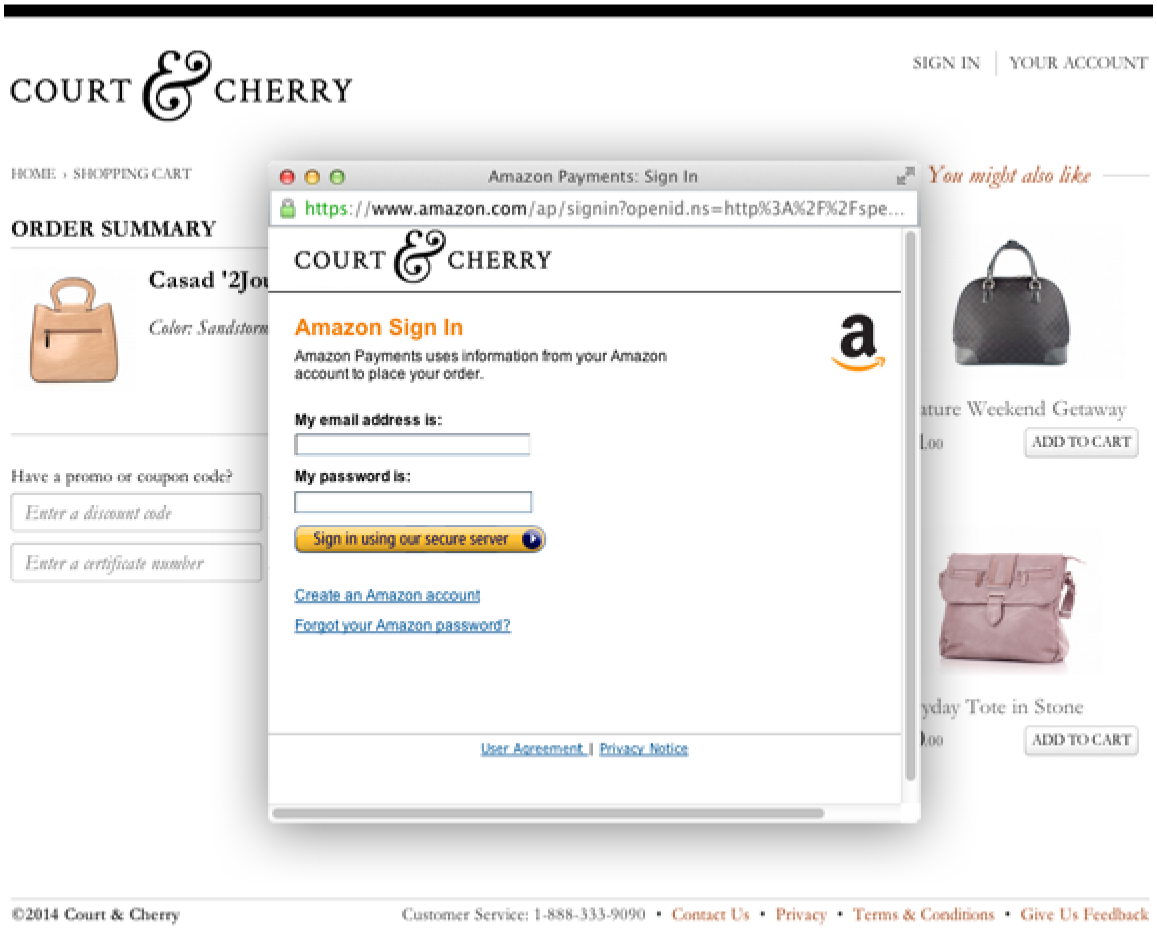
Items	£199.00
Delivery	TBD
Taxes	TBD
Total	**£199.00**

Signature Weekend Getaway
£301.00

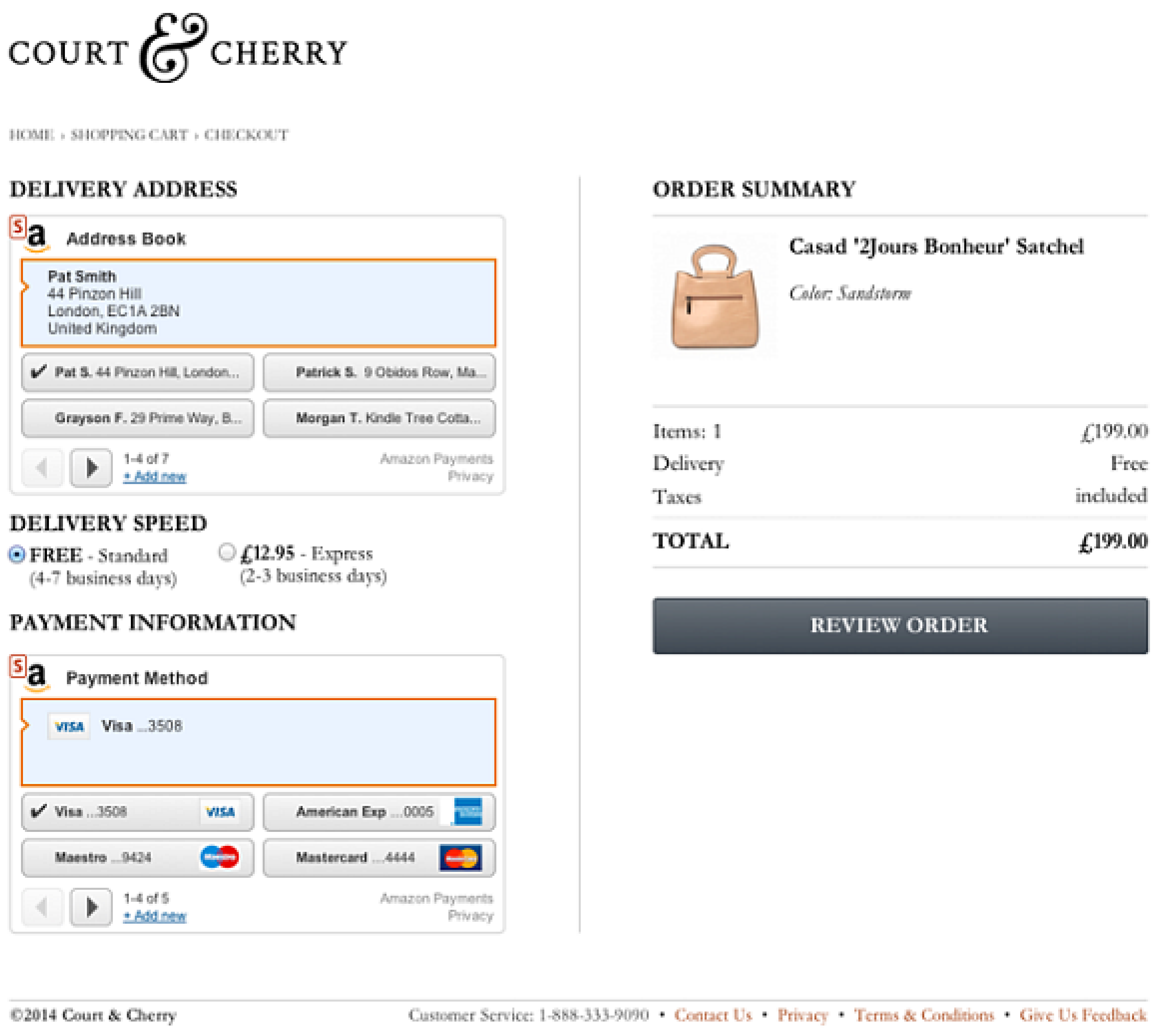
Everyday Tote in Stone
£120.00

©2014 Court & Cherry   Customer Service: 1-888-333-9090 • Contact Us • Privacy • Terms & Conditions • Give Us Feedback

2. The buyer signs in using his or her Amazon username and password.


COURT & CHERRY
SIGN IN | YOUR ACCOUNT
HOME › SHOPPING CART
ORDER SUMMARY
Casad '2Jou
Color: Sandstorm
Have a promo or coupon code?
Enter a discount code
Enter a certificate number
Amazon Payments: Sign In
https://www.amazon.com/ap/signin?openid.ns=http%3A%2F%2Fspe...
COURT & CHERRY
Amazon Sign In
Amazon Payments uses information from your Amazon account to place your order.
My email address is:
My password is:
Sign in using our secure server
Create an Amazon account
Forgot your Amazon password?
User Agreement | Privacy Notice
You might also like
ature Weekend Getaway
ADD TO CART
yday Tote in Stone
ADD TO CART
©2014 Court & Cherry
Customer Service: 1-888-333-9090 • Contact Us • Privacy • Terms & Conditions • Give Us Feedback

3. The buyer selects a shipping address from the **AddressBook** widget and a payment method from the **Wallet** widget.



COURT & CHERRY

HOME › SHOPPING CART › CHECKOUT

DELIVERY ADDRESS

Address Book

Pat Smith
44 Pinzon Hill
London, EC1A 2BN
United Kingdom

✓ Pat S. 44 Pinzon Hill, London...
Patrick S. 9 Obidos Row, Ma...
Grayson F. 29 Prime Way, B...
Morgan T. Kindle Tree Cotta...

1-4 of 7
+ Add new

Amazon Payments
Privacy

DELIVERY SPEED

[✓ FREE - Standard (4-7 business days)] £12.95 - Express (2-3 business days)

PAYMENT INFORMATION

Payment Method

VISA Visa ...3508

✓ Visa ...3508
American Exp ...0005
Maestro ...9424
Mastercard ...4444

1-4 of 5
+ Add new

Amazon Payments
Privacy

ORDER SUMMARY

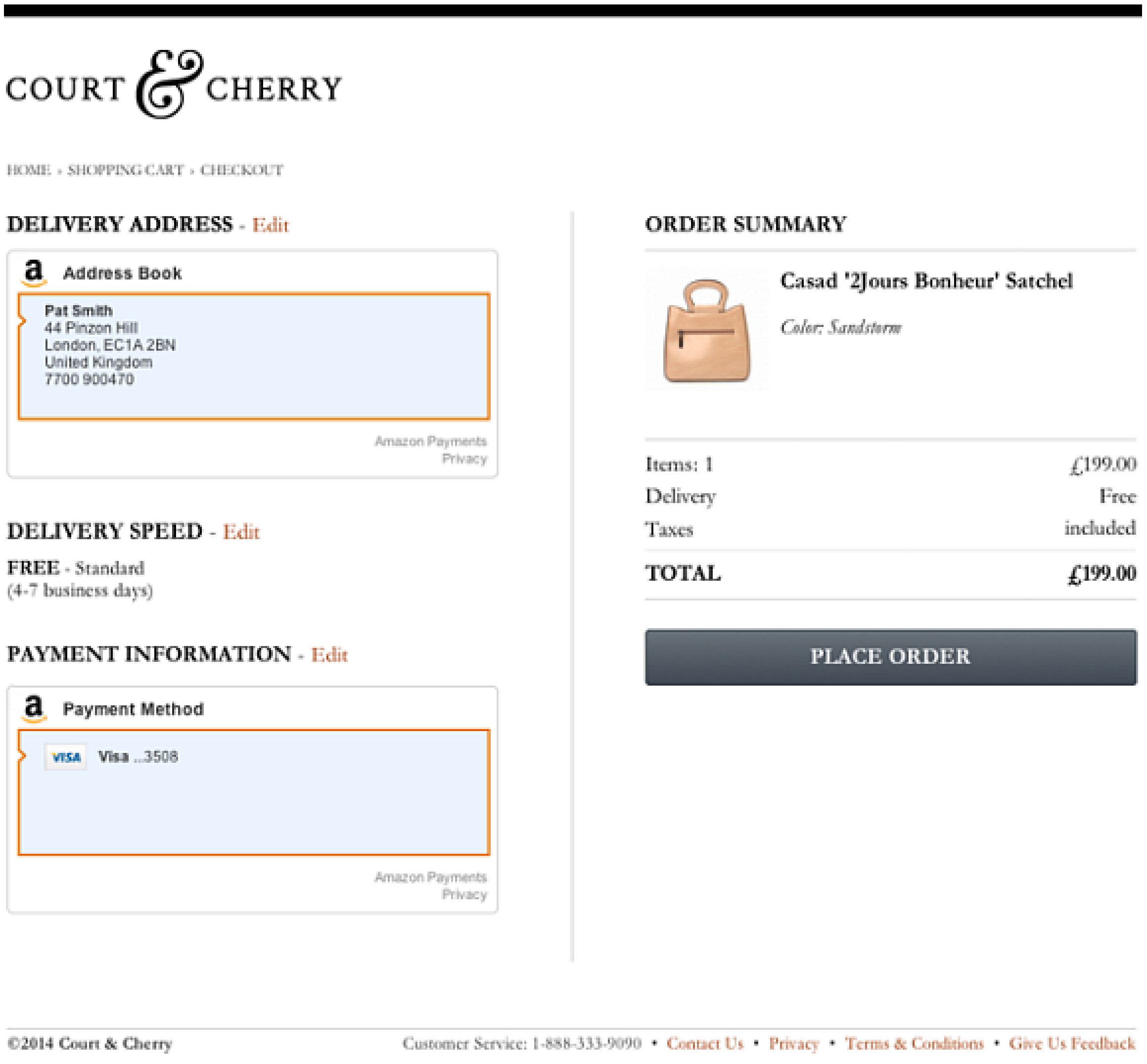
Casad '2Jours Bonheur' Satchel
Color: Sandstorm

Items: 1	£199.00
Delivery	Free
Taxes	included
TOTAL	**£199.00**

REVIEW ORDER

©2014 Court & Cherry Customer Service: 1-888-333-9090 • Contact Us • Privacy • Terms & Conditions • Give Us Feedback

4. The buyer reviews their order.



COURT & CHERRY

HOME › SHOPPING CART › CHECKOUT

DELIVERY ADDRESS - Edit

Address Book

Pat Smith
44 Pinzon Hill
London, EC1A 2BN
United Kingdom
7700 900470

Amazon Payments
Privacy

DELIVERY SPEED - Edit

FREE - Standard
(4-7 business days)

PAYMENT INFORMATION - Edit

Payment Method

VISA Visa ...3508

Amazon Payments
Privacy

ORDER SUMMARY

Casad '2Jours Bonheur' Satchel

Color: Sandstorm

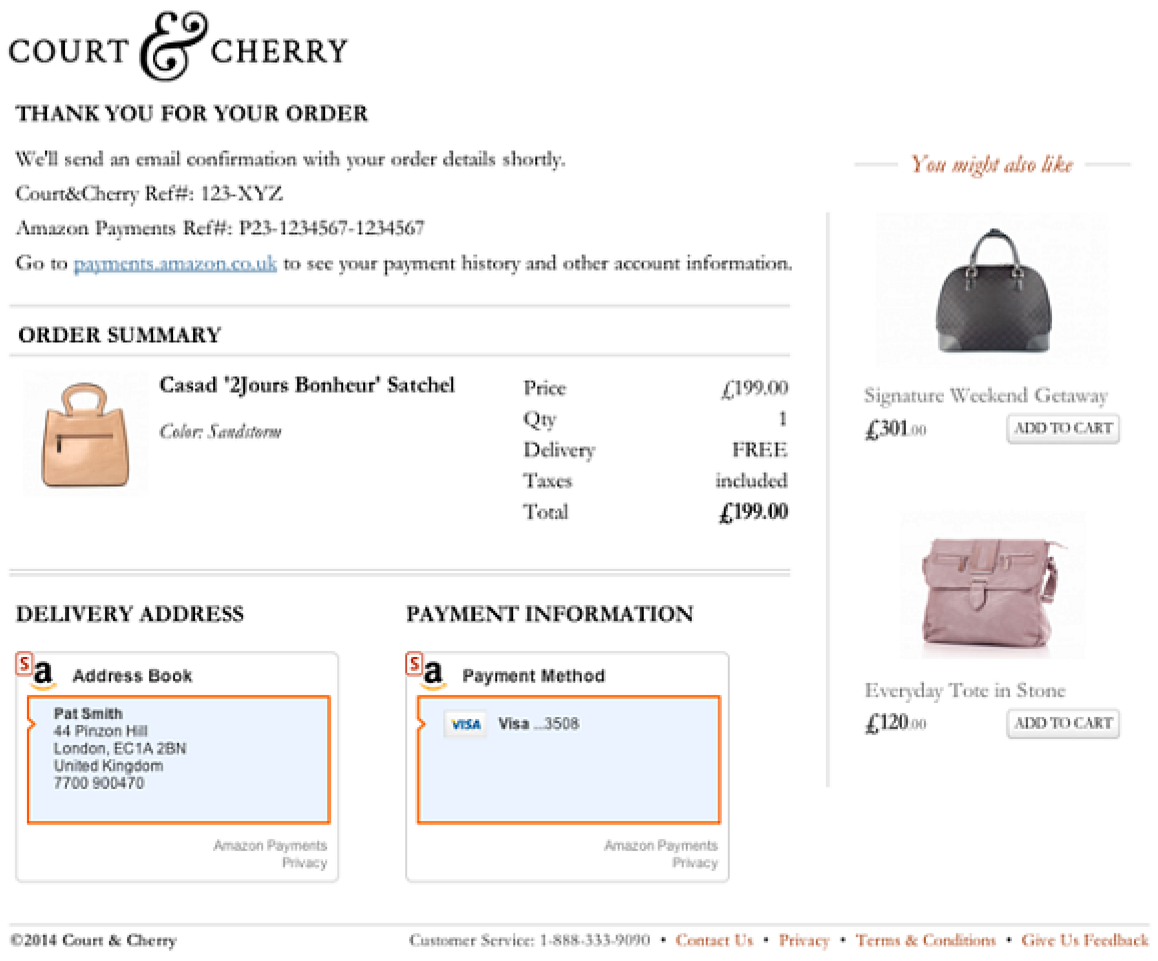
Items: 1	£199.00
Delivery	Free
Taxes	included
TOTAL	**£199.00**

PLACE ORDER

©2014 Court & Cherry

Customer Service: 1-888-333-9090 • Contact Us • Privacy • Terms & Conditions • Give Us Feedback

5. The buyer receives an order confirmation from you.


COURT & CHERRY

THANK YOU FOR YOUR ORDER

We'll send an email confirmation with your order details shortly.

Court&Cherry Ref#: 123-XYZ

Amazon Payments Ref#: P23-1234567-1234567

Go to payments.amazon.co.uk to see your payment history and other account information.

ORDER SUMMARY

Casad '2Jours Bonheur' Satchel

Color: Sandstorm

Price	£199.00
Qty	1
Delivery	FREE
Taxes	included
Total	£199.00

DELIVERY ADDRESS

Address Book

Pat Smith
44 Pinzon Hill
London, EC1A 2BN
United Kingdom
7700 900470

Amazon Payments
Privacy

PAYMENT INFORMATION

Payment Method

VISA Visa ...3508

Amazon Payments
Privacy

You might also like

Signature Weekend Getaway
£301.00 ADD TO CART

Everyday Tote in Stone
£120.00 ADD TO CART

©2014 Court & Cherry

Customer Service: 1-888-333-9090 • Contact Us • Privacy • Terms & Conditions • Give Us Feedback

Overview of integration

The following is an overview of the typical process to set up Amazon Payments on your web site and collect payment for a simple scenario. The steps are grouped into two parts: enabling a purchase and processing payment.

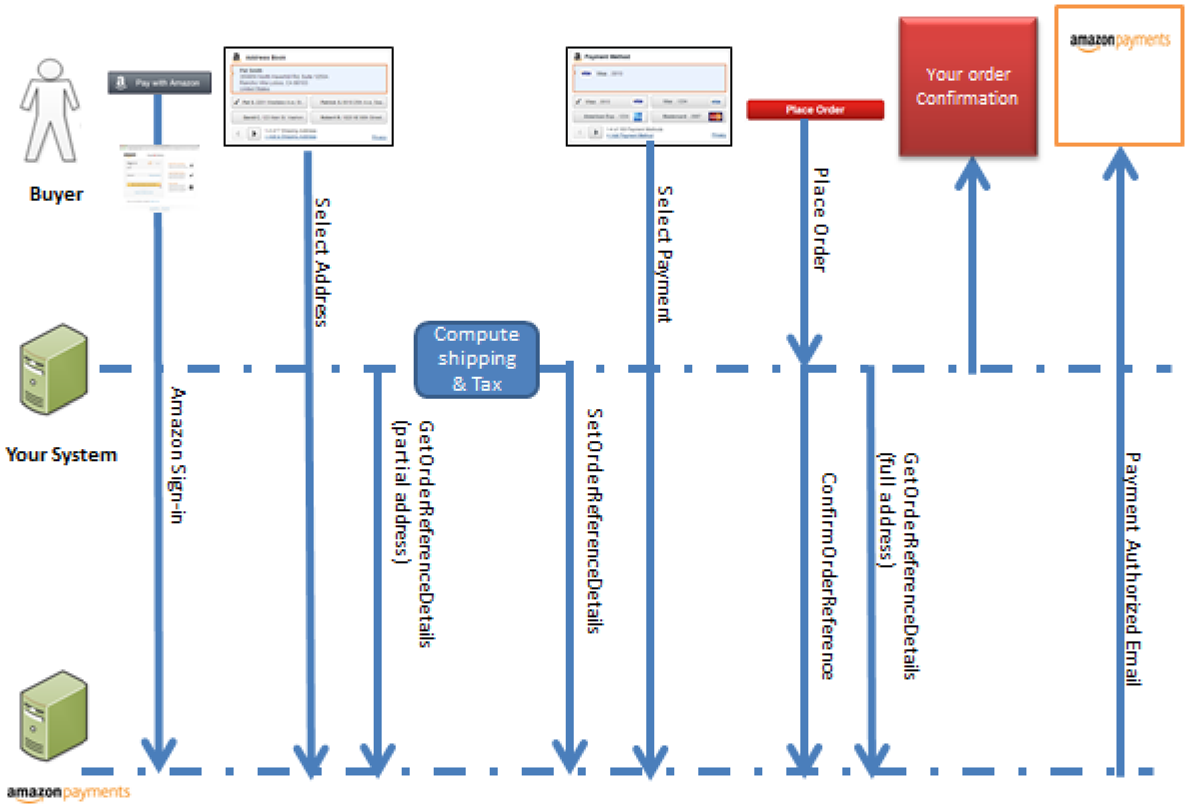
Enabling a purchase

1. Add the **Pay with Amazon** button on the page where the buyer starts the checkout flow (for example, the shopping cart or product detail page). This button enables a buyer to authenticate with Amazon.
2. After the buyer has successfully authenticated, display the Amazon **AddressBook** widget if you need the shipping address to fulfill your order. You should display the widget on the page where you normally collect the shipping address.
3. If you have collected the address using the Amazon **AddressBook** widget, get the partial shipping address (city, state, postal code, and country) by calling the `GetOrderReferenceDetails` operation to compute taxes and shipping costs or possible applicable shipping speed, and options. Display the shipping and tax amount to the buyer.
4. Provide the order total and other optional information to Amazon by calling the `SetOrderReferenceDetails` operation.
5. Display the Amazon **Wallet** widget on the page where you collect payment information.
6. Display the read-only version of the **AddressBook**, if applicable, and **Wallet** widgets together with a summary of the order details on your order review page.

7. After the buyer has placed the order, confirm the order to Amazon by calling the `ConfirmOrderReference` operation.
8. Get the full shipping address from Amazon by calling the `GetOrderReferenceDetails` operation. Ensure that you can ship to this address.
9. Thank the buyer for placing the order. Send an order confirmation e-mail to the buyer. Amazon also sends a "Payment Authorized" e-mail to the buyer.

Note: The "Payment Authorized" e-mail is sent to the buyer when you confirm the order reference to Amazon. This informs the buyer that they have authorized payment for a purchase made using Amazon Payments. This e-mail is not related to the Authorize request that you submit at the time of the order or later.

The following image shows how to enable a purchase:



Processing payment

1. Request Amazon to authorize the buyer's payment with by calling the `Authorize` operation.
2. Listen for the **Authorize** Instant Payment Notification (IPN) for the processing status of your authorization request.

Note: Amazon provides both a synchronous and an asynchronous mode for the Authorize operation.

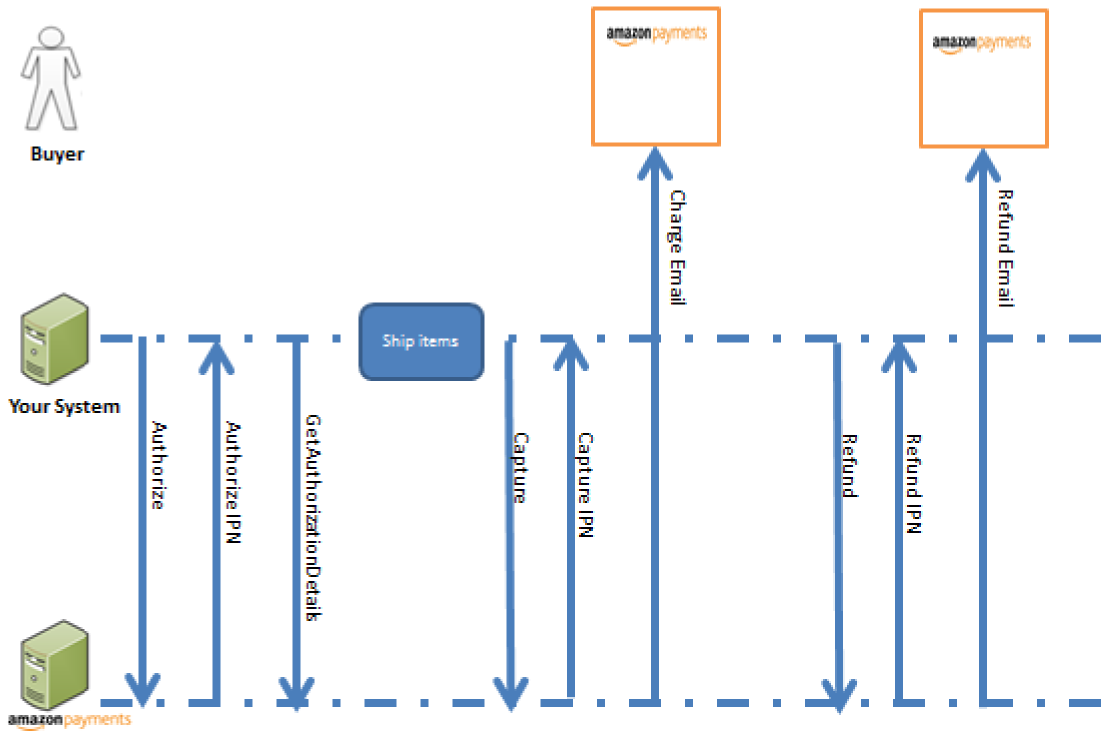
With the synchronous mode, you receive an immediate processing status of Open or Declined. In the asynchronous mode, you first receive a Pending status for the authorization object. After the asynchronous mode processing is complete, Amazon will notify you of the final processing status via an Instant Payment Notification (IPN).

If you are using an asynchronous mode, you should not make the buyer wait for order confirmation on your site until you receive the final processing status from Amazon. You should display the order confirmation page to the buyer immediately after confirming the order to Amazon (step 6 above).

Note: if you are a VAT registered seller, call `GetAuthorizationDetails` to get the billing address, when you receive the **Authorize** IPN.

3. Ship the order and capture payment by calling the `Capture` operation. A `Capture` operation initiated within 7 days of a successful `Authorize` is processed immediately, with a response of **Completed** or **Declined**. A later `Capture` will return a **Pending** state.
4. Listen to the **Capture** IPN for the processing status of your capture request. Amazon sends a "Charge Notification" e-mail to the buyer when the capture is successfully processed.
5. After you have captured the desired amount for the order, mark the Order Reference object as **Closed** by calling the `CloseOrderReference` operation. Amazon uses this information to indicate to the buyer that the payment for this order is complete
6. If you need to issue a refund, you can do so by calling the `Refund` operation.
7. Listen to the **Refund** IPN for the processing status of your refund request. Amazon sends a "Refund Notification" e-mail to the buyer when the refund is successfully processed.

The following image shows how to process a payment:



About the Getting started walkthrough

Note:

The instructions and code samples provided in this guide will help you to embed the Advanced Payment APIs widgets on your web site and prepare you for executing API calls in the Advanced Payment APIs Sandbox mode. At this point, please be reminded that Amazon provides SDKs for the following programming languages:

- PHP
- C#
- Java

This test environment enables you to immediately start testing your end-to-end integration without having to expose the Advanced Payment APIs buyer-facing elements until you are ready to go live. For more information about the Advanced Payment APIs Sandbox mode, see Testing your integration with the Sandbox environment. To switch from the Sandbox environment to the Production environment, make the following changes:

Parameter	Type / Location	Value
Amazon MWS service endpoints	Sandbox / DE	https://mws-eu.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01/
	Production / DE	https://mws-eu.amazonservices.com/OffAmazonPayments/2013-01-01/
	Sandbox / UK	https://mws-eu.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01/
	Production / UK	https://mws-eu.amazonservices.com/OffAmazonPayments/2013-01-01/
Widgets.js URL	Sandbox / DE	https://static-eu.payments-amazon.com/OffAmazonPayments/de/sandbox/js/Widgets.js
	Production / DE	https://static-eu.payments-amazon.com/OffAmazonPayments/de/js/Widgets.js
	Sandbox / UK	https://static-eu.payments-amazon.com/OffAmazonPayments/uk/sandbox/js/Widgets.js
	Production / UK	https://static-eu.payments-amazon.com/OffAmazonPayments/uk/js/Widgets.js
Pay with Amazon button URL	Sandbox / DE	https://payments-sandbox.amazon.de/gp/widgets/button
	Production / DE	https://payments.amazon.de/gp/widgets/button
	Sandbox / UK	https://payments-sandbox.amazon.co.uk/gp/widgets/button
	Production / UK	https://payments.amazon.co.uk/gp/widgets/button

Getting ready to use the Advanced Payment APIs widgets

To properly render the embedded widgets on your site, you must add a reference to the Advanced Payment APIs JavaScript file, Widgets.js, to the source code for your web page. You must add a reference to the Advanced Payment APIs JavaScript file from any web page where you want to render Advanced Payment APIs widgets.

To add a reference to the Advanced Payment APIs JavaScript file, add a few lines of code as shown in the following example code. As a best practice, Amazon Payments recommends placing this code in the **head** section of your web pages. In addition you should not make a local copy or modify the content of the Amazon-provided JavaScript. This can cause unintended consequences such as error in the rendering or functionality of the widgets.

```
<head>
// your head section here
<script type='text/javascript'
  src='https://static-eu.payments-amazon.com/OffAmazonPayments/xx/sandbox/js/
Widgets.js
       ?sellerId=YOUR_SELLER_ID_HERE'>
</script>
// your head section here
</head>
```

In the above code sample, "xx" in the URL would be "de" or "uk" as appropriate. Please be aware that the URL is case sensitive.

After the reference to the Advanced Payment APIs JavaScript file has been added to your source code, you are ready to start embedding the Advanced Payment APIs widgets.

Step 1 - Add the Button widget for buyer authentication

The first widget that you need to embed and present to your buyers is the Advanced Payment APIs **Button** widget. This widget displays the button that lets buyers know they can use their Amazon account to pay for their order.

Assuming that you have added the Widgets.js JavaScript file (as described earlier in this section), you can embed the button by adding the following code:

For UK:

```
<div id="payWithAmazonDiv">
  <img src="https://payments-sandbox.amazon.co.uk/gp/widgets/button
            ?sellerId=YOUR_SELLER_ID_HERE&size=large&color=orange"
```

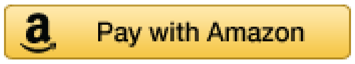
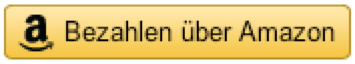
```
       style="cursor: pointer;"/>
</div>
```

For DE:

```
<div id="payWithAmazonDiv">
  <img src="https://payments-sandbox.amazon.de/gp/widgets/button
            ?sellerId=YOUR_SELLER_ID_HERE&size=large&color=orange"
       style="cursor: pointer;"/>
</div>
```

The code sample above defines a division in your web page which is used to render the button. You should place this code in the specific spot where you want to show the button. Replace the text "YOUR_SELLER_ID_HERE" with your own Seller ID that you received during registration. If you are not sure what your Seller ID is, log in to Seller Central and visit the **Integration Settings** section under the **Settings** tab to view your Seller ID.

In addition, you can optionally specify button size and color parameters. The following table lists the supported properties and the defaults which are applied if no properties are specified:

Parameter	**Description**
Button - UK	
Button - DE	
Language	Automatically set, based on marketplace location
Color	• Orange • Tan Default: Orange
Size	• medium (126px x 24px) • large (151px x 27px) • x-large (173px x 27px) Default: medium

To render the Amazon Payments authentication window successfully, you will need to add the following code to your web site:

```
<script>
 var amazonOrderReferenceId;
 new OffAmazonPayments.Widgets.Button ({
   sellerId: 'YOUR_SELLER_ID_HERE',
   onSignIn: function(orderReference) {
     amazonOrderReferenceId = orderReference.getAmazonOrderReferenceId();
     window.location = 'https://yoursite.com/redirect_page?session='
        + amazonOrderReferenceId;
   },
   onError: function(error) {
    // your error handling code
   }
 }).bind("payWithAmazonDiv");
 </script>
```

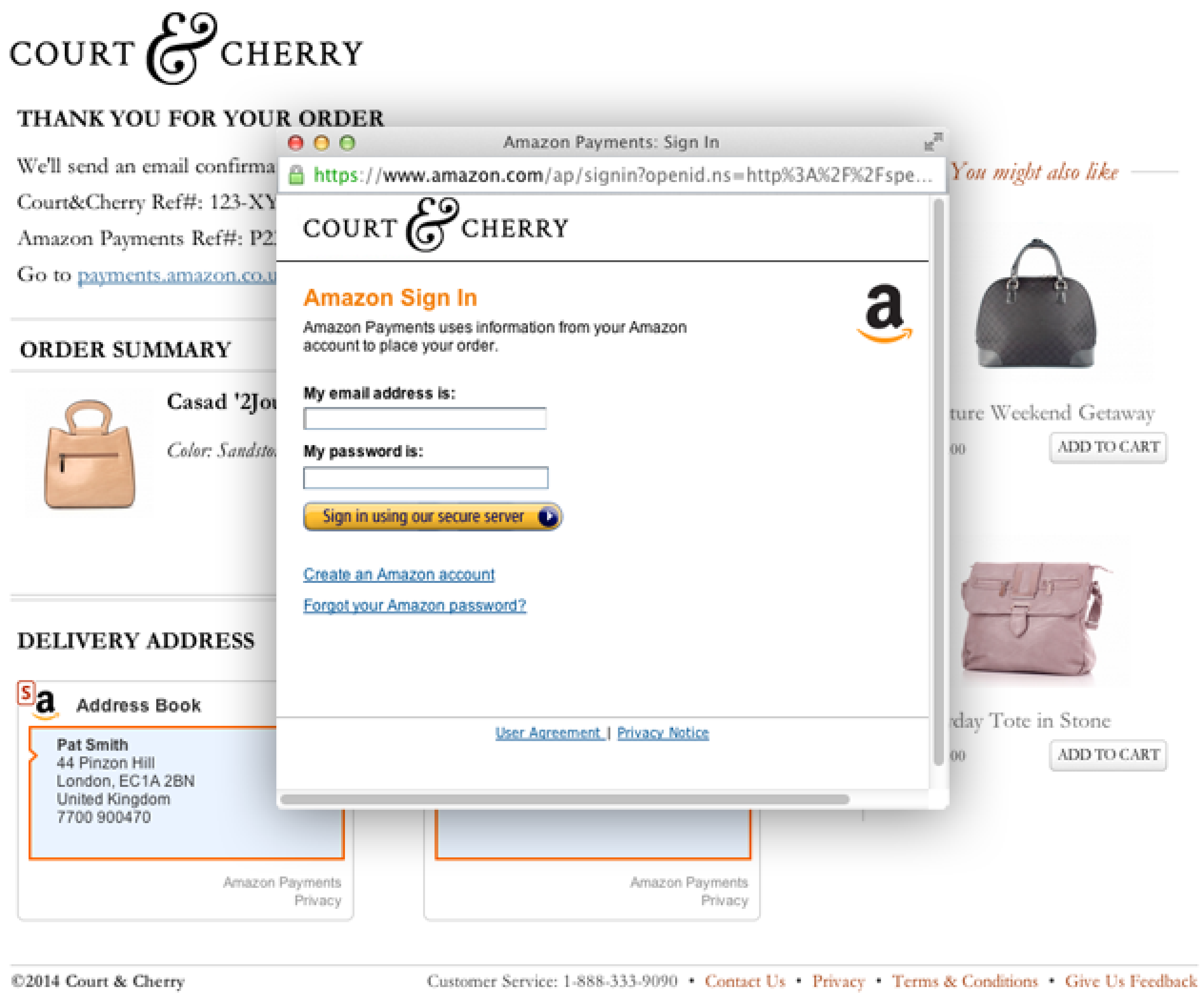
After the code in the previous example is added, clicking on the button will launch the Amazon Payments authentication window, where your buyer will be asked for their Amazon account e-mail address and password.

After your buyer successfully signed in, the JavaScript function specified in the `onSignIn` parameter is called. Amazon recommends that you use the `onSignIn` function to redirect the window location to your web page where the buyer will continue the checkout process. For example, after successful authentication you might want to take the buyer to the next page in your checkout pipeline to collect shipping address and/or payment method details.

You can use the `onSignIn` callback function to perform or call any function required by your checkout flow. The parameter provided in the `.bind()` function call must be the same as the name of the div where you rendered the button. The `orderReference` object (the parameter of the callback function) has a method to get the `amazonOrderReferenceId`. The sample code provided shows you how to get the `amazonOrderReferenceId` and use it to construct the name of the page to redirect to after sign in. You must store the `amazonOrderReferenceId` as you will need it throughout your Advanced Payment APIs integration.

Amazon also recommends that you implement an `onError` handler in your code. In the previous example, the `onError` code is optional, but it can save you considerable effort with troubleshooting integration issues. For more information, see Testing your integration with the Sandbox environment.

After you have embedded the button and added the JavaScript code, a window displays where buyers can authenticate without ever leaving your web site:



COURT & CHERRY

THANK YOU FOR YOUR ORDER

We'll send an email confirma
Court&Cherry Ref#: 123-XY
Amazon Payments Ref#: P2
Go to payments.amazon.co.u

ORDER SUMMARY

Casad '2Jou
Color: Sandsto

DELIVERY ADDRESS

Address Book

Pat Smith
44 Pinzon Hill
London, EC1A 2BN
United Kingdom
7700 900470

Amazon Payments Privacy

Amazon Payments: Sign In

https://www.amazon.com/ap/signin?openid.ns=http%3A%2F%2Fspe...

COURT & CHERRY

Amazon Sign In

Amazon Payments uses information from your Amazon account to place your order.

My email address is:

My password is:

Sign in using our secure server

Create an Amazon account

Forgot your Amazon password?

User Agreement | Privacy Notice

You might also like

ture Weekend Getaway

ADD TO CART

day Tote in Stone

ADD TO CART

©2014 Court & Cherry

Customer Service: 1-888-333-9090 • Contact Us • Privacy • Terms & Conditions • Give Us Feedback

As you can see in the previous screen shot, you can add your company logo to the authentication window. You can do this from the **Integration Settings** page under the **Settings** tab on Seller Central.

Note: After a customer is successfully authenticated, Pay with Amazon creates an Order Reference object in the **Draft** state (see Step 5 in this section for more details about the Order Reference object); the **Draft** Order Reference object will be canceled if you do not complete Step 4 - Set purchase details and confirm the purchase within three hours of a buyer clicking the button on your site.

Step 2 - Add the AddressBook and Wallet widgets

After a buyer has successfully authenticated, you can render the Advanced Payment APIs **AddressBook** and **Wallet** widgets in the location of your choosing. The **AddressBook** and **Wallet** widgets will display the shipping addresses and payment methods the buyer has saved in their Amazon account, respectively.

To embed the **AddressBook** widget on your web site, add the following example code to your web page:

```
<!-- please put the style below inside your CSS file -->
<style type="text/css">
  #addressBookWidgetDiv{width: 400px; height: 228px;}
</style>
```

```
<div id="addressBookWidgetDiv">
</div>

<script>
new OffAmazonPayments.Widgets.AddressBook({
  sellerId: 'YOUR_SELLER_ID_HERE',
  amazonOrderReferenceId: amazonOrderReferenceId,
  // amazonOrderReferenceId obtained from Button widget
  onAddressSelect: function(orderReference) {
    // Replace the following code with the action you that want to perform
    // after the address is selected.
    // The amazonOrderReferenceId can be used to retrieve
    // the address details by calling the GetOrderReferenceDetails
    // operation.
    // If rendering the AddressBook and Wallet widgets on the same page, you
    // should wait for this event before you render the Wallet widget for
    // the first time.
    // The Wallet widget will re-render itself on all subsequent
    // onAddressSelect events, without any action from you. It is not
    // recommended that you explicitly refresh it.
  },
  design: {
     designMode: 'responsive'
  },
  onError: function(error) {
   // your error handling code
  }
}).bind("addressBookWidgetDiv");
</script>
```

To embed the **Wallet** widget on your web site, add the following example code to your web page:

```
<!-- please put the style below inside your CSS file -->
<style type="text/css">
  #walletWidgetDiv{width: 400px; height: 228px;}
</style>
```

```
<div id="walletWidgetDiv">
</div>
<script>

new OffAmazonPayments.Widgets.Wallet({
  sellerId: 'YOUR_SELLER_ID_HERE',
  amazonOrderReferenceId: amazonOrderReferenceId,
  // amazonOrderReferenceId obtained from Button widget
  onPaymentSelect: function(orderReference) {
    // Replace this code with the action that you want to perform
    // after the payment method is selected.
  },
  design: {
    designMode: 'responsive'
  },
  onError: function(error) {
    // your error handling code
  }
}).bind("walletWidgetDiv");
</script>
```

You can control the widget height and width in your CSS as long as you provide a height and width range within our allowed parameters. You can also use other CSS unit measurements like percent, REM, and EM as long as the minimum and maximum height, and the width of the element, are within the provided ranges.

If you have a responsive site please follow these steps:

1. Add a meta tag to the head section of each of your smartphone-optimized pages. This meta tag will cause the widgets to scale to the size of the available screen so that is is readable on a smartphone without requiring the user to manually zoom the page:

```
<meta name="viewport" content="width-device-width, initial-scale=1.0,
 maximum-scale=1.0"/>
```

2. You can use media query with a CSS to build a mobile first design with example below:

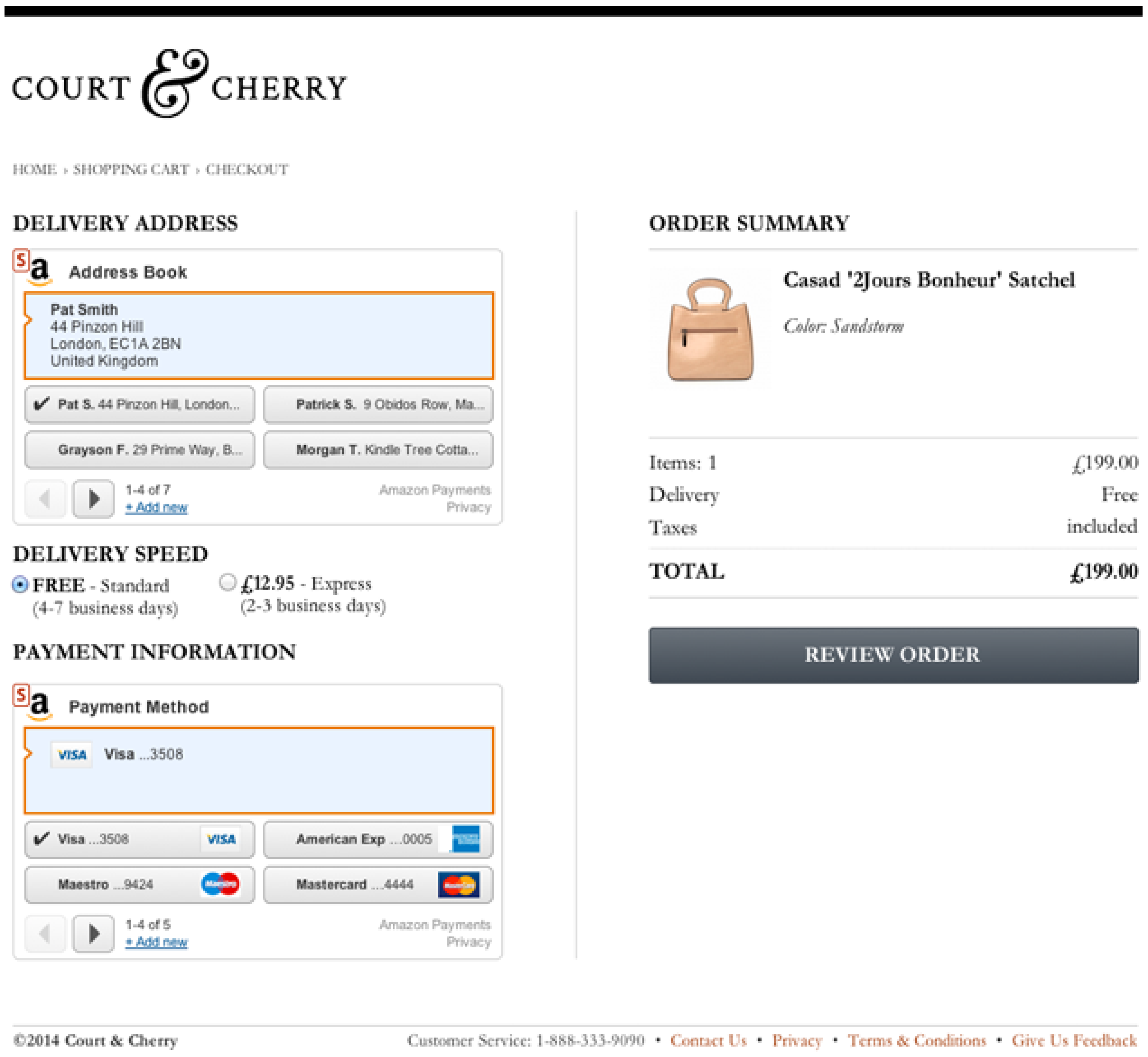
```
<style type="text/css">
/* Please include the min-width, max-width, min-height and max-height
 */
/* if you plan to use a relative CSS unit measurement to make sure the */
/* widget renders in the optimal size allowed.          */

#addressBookWidgetDiv {min-width: 300px; max-width: 600px; min-height:
228px; max-height: 400px;}
#walletWidgetDiv {min-width: 300px; max-width:600px; min-height: 228px;
max-height: 400px;}

/* Smartphone and small window */
#addressBookWidgetDiv {width: 100%; height: 228px;}
#walletWidgetDiv {width: 100%; height: 228px;}

/* Desktop and tablet */
@media only screen and (min-width: 768px) {
    #addressBookWidgetDiv {width: 400px; height: 228px;}
    #walletWidgetDiv {width: 400px; height: 228px;}
}
```

Notice that in the example code you must specify the `amazonOrderReferenceId` when rendering the **AddressBook** and **Wallet** widgets. If the `amazonOrderReferenceId` is not specified, the widget will display an error message.

When creating the Advanced Payment APIs **AddressBook** and **Wallet** widgets, you must specify `width` and `height` parameters; otherwise, the widgets will not render. The valid and recommended `width` and `height` parameters for the one- and two-column widgets are specified in the following table:

	Two-Column Widget		**One-Column Widget**	
Parameter	**Recommended Dimension**	**Valid Dimensions**	**Recommended Dimension**	**Valid Dimensions**
width	*400px*	*400px - 600px*	*300px*	*300px - 399px*
height	*228px*	*228px - 400px*	*228px*	*228px - 400px*

Following a successful authentication, the buyer will see the **AddressBook** widget where you specified the widget to render, similar to the following screen shot:



COURT & CHERRY

HOME › SHOPPING CART › CHECKOUT

DELIVERY ADDRESS

Address Book

Pat Smith
44 Pinzon Hill
London, EC1A 2BN
United Kingdom

✓ Pat S. 44 Pinzon Hill, London...
Patrick S. 9 Obidos Row, Ma...
Grayson F. 29 Prime Way, B...
Morgan T. Kindle Tree Cotta...

1-4 of 7
+ Add new
Amazon Payments
Privacy

DELIVERY SPEED

● FREE - Standard (4-7 business days)
○ £12.95 - Express (2-3 business days)

PAYMENT INFORMATION

Payment Method

VISA Visa ...3508

✓ Visa ...3508
American Exp ...0005
Maestro ...9424
Mastercard ...4444

1-4 of 5
+ Add new
Amazon Payments
Privacy

ORDER SUMMARY

Casad '2Jours Bonheur' Satchel
Color: Sandstorm

Items: 1	£199.00
Delivery	Free
Taxes	included
TOTAL	**£199.00**

REVIEW ORDER

©2014 Court & Cherry
Customer Service: 1-888-333-9090 • Contact Us • Privacy • Terms & Conditions • Give Us Feedback

Please note that in the previous screen shot, the **AddressBook** and **Wallet** widgets render directly on your web site.

Amazon recommends that you disable the checkout workflow until you are sure that the buyer has selected an address and payment method from the respective widgets. Amazon notifies you of the address and payment method selection through the `onAddressSelect` and `onPaymentSelect` callback functions, respectively. Based on these notifications, you can enable the next step in your checkout flow. For example, you can enable the continue button on your web page if you have an order review page that follows.

Note: If you render the **AddressBook** and **Wallet** widgets on the same page, listen to the `onAddressSelect` event before you render the **Wallet** widget for the first time.

Note: A buyer must select both an address and a payment method before you can confirm the order reference. If the buyer changes their address selection after making an initial address and payment method selection, they must then reselect a payment method. This means that you must listen for an `onPaymentSelect` notification following each address selection (`onAddressSelect` notification). If you do not need to collect address information from the **AddressBook** widget, please see Integrating without the AddressBook widget.

Getting the shipping address from Amazon

In many cases you will need to obtain the buyer's shipping address to calculate shipping charges and taxes. After receiving the `onAddressSelect` notification, you can call the `GetOrderReferenceDetails` operation to support a number of business rules around shipping and taxes. You can also provide your buyers with immediate feedback if they selected an unsupported address from the widget. In this situation, you should display a message to the buyer that they need to choose a different shipping address from the widget.

To retrieve the address that the buyer has selected, use Ajax to call the `GetOrderReferenceDetails` operation after the `onAddressSelect` notification has fired.

Note: Calling the `GetOrderReferenceDetails` operation before the Order Reference is confirmed (see Step 4) returns only the **City**, **StateOrRegion**, **CountryCode**, and **PostalCode** elements of the address. This information can be used to calculate shipping and taxes, or to compare the selected address to internal business rules about shipping restrictions. After you confirm the order reference by calling the `ConfirmOrderReference` operation, the `GetOrderReferenceDetails` operation returns the full shipping address that the buyer selected as well as buyer's name, e-mail address, and phone number (if available). You can use this information to fulfill your orders and contact the buyer for any customer service related issues.

Note: All of your API calls must be signed for them to work properly. To sign your API requests you will need Amazon MWS keys. You can get these keys by logging into Seller Central and viewing the **MWS Access Key** section under the **Integration** tab. This Amazon MWS key must be set in the **AWSAccessKeyId** request parameter of the Off-Amazon Payments API section operations.

Amazon offers predefined libraries and example code for PHP, C#, and Java. To simplify integration you may download the corresponding SDK from one of the following locations:

- PHP Client Library
- C# Client Library
- Java Client Library

Amazon strongly recommends using these libraries, since they not only make the integration easier for you, but also reduce the risk for programming errors. In case you use a programming language that is not supported by our currently available SDKs, you will have to implement the calls yourself, which is explained below.

Please note that the signing your calls is required. For more information about how to configure your services to make signed API calls, see the Amazon MWS Developer Guide.

The following example shows how to call the `GetOrderReferenceDetails` operation:

```
https://mws-eu.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01
?AWSAccessKeyId=AKIAJKYFSJU7PEXAMPLE
&Action=GetOrderReferenceDetails
&AmazonOrderReferenceId=S23-1234567-1234567
&MWSAuthToken=amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE
&OrderReferenceAttributes.OrderTotal.Amount=199.00
&OrderReferenceAttributes.OrderTotal.CurrencyCode=EUR
&OrderReferenceAttributes.SellerNote= For%20your%20order%20of%20Casad
%20%272Jours%20Bonheur%27%20Satchel%20in%20Sandstorm
&OrderReferenceAttributes.SellerOrderAttributes.SellerOrderId=123-XYZ
&OrderReferenceAttributes.SellerOrderAttributes.StoreName=CourtAndCherry.co.uk
&SellerId=YOUR_SELLER_ID_HERE
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2012-10-03T19%3A01%3A11Z
&Version=2013-01-01
&Signature=CLZOdtJGjAo81IxaLoE7af6HqK0EXAMPLE
```

How you set the parameters in the Order Reference details will affect the detail appearing in the Payment Authorized email. For more information see "The Payment Authorized email" section under "Controlling the data that appears in emails".

For more information about the `GetOrderReferenceDetails` operation, including the request parameters and response elements, see "GetOrderReferenceDetails" in the Off-Amazon Payments API section reference.

Recommendations for re-rendering Wallet widgets

In most cases you do not need to refresh the **Wallet** widget as it will automatically refresh when a buyer selects an address from the **AddressBook** widget. Frequent re-rendering of the **Wallet** widget will affect website performance and cause a poor buyer experience.

However, there are times when certain constraints are returned in the checkout flow and the re-rendering of the **Wallet** widget is actually required. For example, Amazon Payments may not allow all payment methods for a given transaction. In our scenario, a buyer selects Direct Debit as a payment method for a high-value purchase. You pass the amount of the transaction to the SetOrderReferenceDetails operation.

Since the OrderTotal is calculated after the buyer has selected the payment method from the Wallet widget, the SetOrderReferenceDetails operation might result in the PaymentMethodNotAllowed constraint. .

To resolve this constraint, you will need to re-render the **Wallet** widget and request the buyer to select a different payment method.

For more information on how to simulate constraints for integration testing see Sandbox Simulation of Constraints, or for more information about constraints see "Order Reference Constraints" in the Off-Amazon Payments API section reference.

Step 3 - Display the selected shipping address and payment method (optional)

An optional but highly recommended step after the buyer has selected both their shipping address and payment method from the Amazon Payments widgets is to render read-only versions of the **AddressBook** and **Wallet** widgets to give the buyer one last chance to review their selections before confirming their purchase.

If you decide to have an order review page and display the read-only widgets, Amazon recommends that you verify that the buyer has selected an address and payment method before you show a confirmation page. You can do this by calling the `GetOrderReferenceDetails` operation and checking if the response contains a **Constraints** element. Each child constraint will provide you with the information you need in order to take appropriate corrective action.

For example, if you receive either a **ShippingAddressNotSet** or a **PaymentPlanNotSet** constraint, you might want to let the buyer know that they still need to select a shipping address or payment method. Or, you might choose to integrate your site to automatically redirect the buyer to the appropriate page to make a selection.

If the buyer has not selected a shipping address or payment method, the read-only widgets will also inform the buyer that they need to make a selection. If an order reference returns no constraints then it means that a `ConfirmOrderReference` call will succeed. For more information about constraints, see the "GetOrderReferenceDetails" and "Order Reference Constraints" sections in the Off-Amazon Payments API section reference. After the buyer confirms their purchase, you can proceed with fulfilling the order and collecting the payment.

When creating the read-only Advanced Payment APIs **AddressBook** and **Wallet** widgets, you must specify `width` and `height` parameters; otherwise, the widgets will not render. The valid and recommended `width` and `height` parameters are specified in the following table:

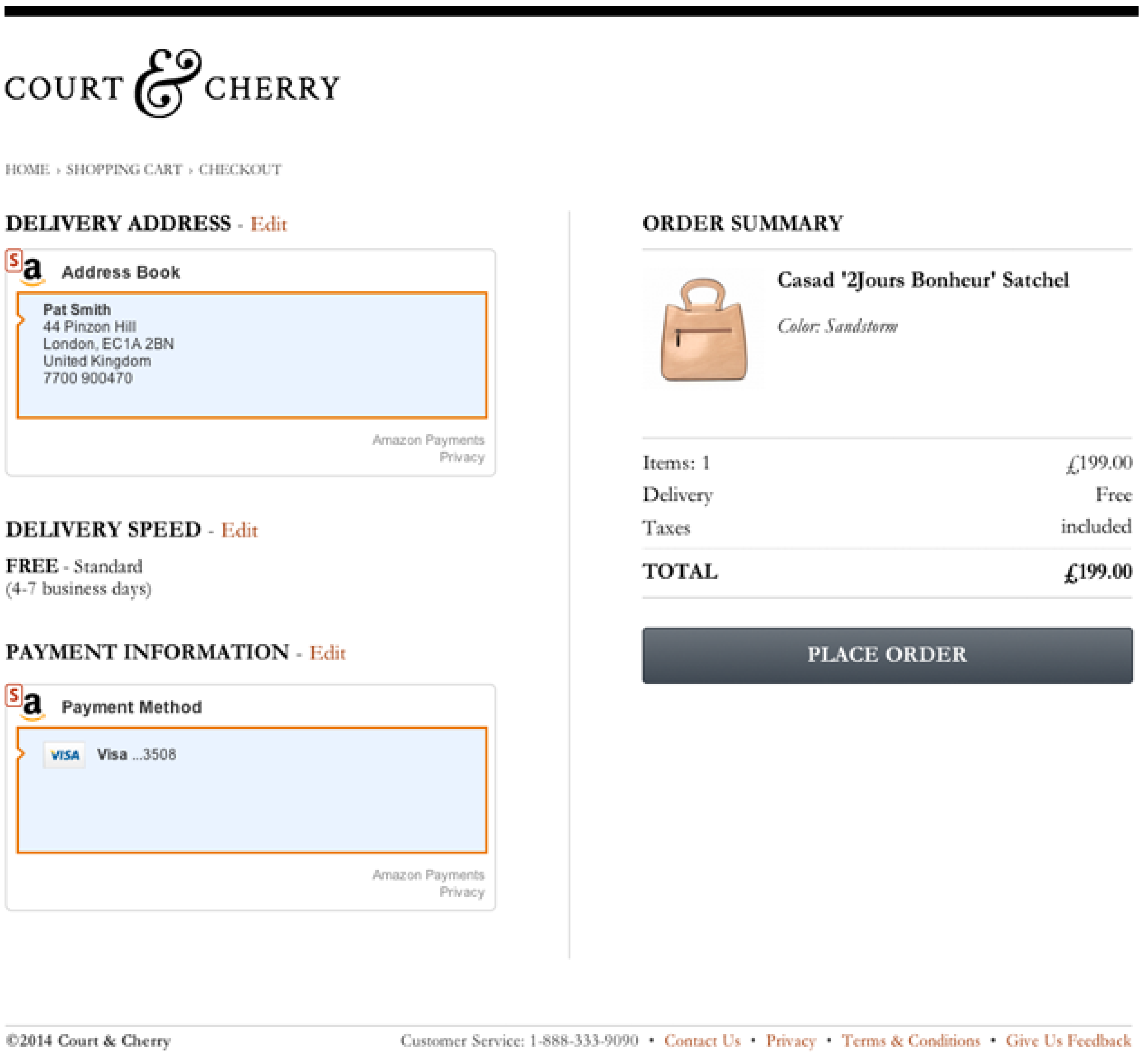
Parameter	Recommended Dimension	Valid Dimensions
width	*400px*	*300px - 600px*
height	*185px*	*185px*

To display the read-only versions of the widgets, you can use the same code that originally rendered the widgets on your site with the added parameter `displayMode: "Read"` inside the code. If you need to display the read-only widget on the same page as the editable **AddressBook** or **Wallet** widgets, subscribe to the `onAddressSelect` or `onPaymentSelect` notifications and redraw the read-only widget at that time. Otherwise the read-only widget will not display the correct information. The following is an example of the code with the added parameter:

```
<!-- please put the style below inside CSS file -->
<style type="text/css">
  #readOnlyAddressBookWidgetDiv {width: 400px; height: 185px;}
  #readOnlyWalletWidgetDiv {width: 400px; height: 185px;}
</style>
```

```
<div id="readOnlyAddressBookWidgetDiv">
</div>

<div id="readOnlyWalletWidgetDiv">
</div>

<script>
new OffAmazonPayments.Widgets.AddressBook({
  sellerId: 'YOUR_SELLER_ID_HERE',
  amazonOrderReferenceId: amazonOrderReferenceId,
  // amazonOrderReferenceId obtained from Button widget
  displayMode: "Read",
  design: {
     designMode: 'responsive'
  },
  onError: function(error) {
   // your error handling code
  }
}).bind("readOnlyAddressBookWidgetDiv");
</script>
<script>
new OffAmazonPayments.Widgets.Wallet({
  sellerId: 'YOUR_SELLER_ID_HERE',
  amazonOrderReferenceId: amazonOrderReferenceId,
  // amazonOrderReferenceId obtained from Button widget
  displayMode: "Read",
  design: {
     designMode: 'responsive'
  },
  onError: function(error) {
   // your error handling code
  }
}).bind("readOnlyWalletWidgetDiv");
</script>
```

When the extra parameter is added, the read-only version of the widgets displays on your web site similar to the following screen shot:



DELIVERY ADDRESS - Edit

Address Book

Pat Smith
44 Pinzon Hill
London, EC1A 2BN
United Kingdom
7700 900470

Amazon Payments
Privacy

DELIVERY SPEED - Edit

FREE - Standard
(4-7 business days)

PAYMENT INFORMATION - Edit

Payment Method

VISA Visa ...3508

Amazon Payments
Privacy

ORDER SUMMARY

Casad '2Jours Bonheur' Satchel
Color: Sandstorm

Items: 1	£199.00
Delivery	Free
Taxes	included
TOTAL	£199.00

©2014 Court & Cherry Customer Service: 1-888-333-9090 • Contact Us • Privacy • Terms & Conditions • Give Us Feedback

Step 4 - Set purchase details and confirm the purchase

The Order Reference object

As soon as your buyer authenticates with Amazon as shown in Step 1, Advanced Payment APIs creates a system object called the Order Reference object. This object stores all the attributes associated with the payment that the buyer is going to make using Pay with Amazon.

At this point in the Pay with Amazon checkout process, the Order Reference object has been populated with the buyer’s shipping address and payment method selections. You must now set additional attributes in the Order Reference object that will help associate the Pay with Amazon payment with any other details that you are tracking. You must set the purchase amount. You can also set additional details such as the order number that you have specified for the transaction.

After you set these attributes by calling the `SetOrderReferenceDetails` operation, you must confirm the purchase to Amazon by calling the `ConfirmOrderReference` operation. After you confirm the purchase, Amazon will inform the buyer about the payment that they have authorized. You must confirm the order reference before you can process the payment for the order.

Setting the Order Reference details

To set the specifics of the Order Reference object, you call the `SetOrderReferenceDetails` operation. During this call, you specify the total amount of the purchase, and it enables you to provide any additional information about the purchase that you would like included in the Order Reference object. This additional information is used in both buyer-facing communications from Amazon as well as in payment processing information returned to you.

The Order Reference Object contains the **AmazonOrderReferenceId**, which is required to close the order reference after all processing is completed, via the `CloseOrderReference` operation, and for processing any associated notifications.

The following example shows how to call the `SetOrderReferenceDetails` operation:

```
https://mws-eu.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01
?AWSAccessKeyId=0GS7553JW74RRM612K02EXAMPLE
&Action=SetOrderReferenceDetails
&AmazonOrderReferenceId=S23-1234567-1234567
&MWSAuthToken=amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE
&OrderReferenceAttributes.OrderTotal.Amount=199
&OrderReferenceAttributes.OrderTotal.CurrencyCode=EUR
&OrderReferenceAttributes.SellerNote=For%20your%20order%20of%20Casad
%20%272Jours%20Bonheur%27%20Satchel%20in%20Sandstorm
&OrderReferenceAttributes.SellerOrderAttributes.SellerOrderId=123-XYZ
&OrderReferenceAttributes.SellerOrderAttributes.StoreName=CourtAndCherry.co.uk
&SellerId=YOUR_SELLER_ID_HERE
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2013-11-16T19%3A01%3A11Z
&Version=2013-01-01
&Signature=2RPzkOgQmDybUjk0dA54maCEXAMPLE
```

How you set the parameters in the Order Reference details will affect the detail appearing in the Payment Authorized email. For more information see The Payment Authorized email section under Controlling the data that appears in emails.

For more information about the `SetOrderReferenceDetails` operation, including the request parameters and response elements, see "SetOrderReferenceDetails" in the Off-Amazon Payments API section reference.

Note: You must set the purchase amount at this step in the Advanced Payment APIs integration process. If the purchase amount will change based upon the selected address, call the `GetOrderReferenceDetails` operation after each `onAddressSelect` event and reset the purchase amount accordingly.

Order Reference Constraint

Amazon Payments may not always allow all payment methods for a given transaction. Depending on various factors, some payment methods may not be available. To determine the available payment methods, the amount of the transaction must be passed to the `SetOrderReferenceDetails` operation.

To ensure that only allowed payment methods are shown to a buyer, Amazon requires that you set the **OrderTotal** using the `SetOrderReferenceDetails` operation before displaying the **wallet** widget as suggested in the section Overview of integration. Performing this operation enables Amazon to determine which payment methods are allowed for this Order Reference.

If you set the **OrderTotal** *after* the buyer has selected the payment method from the **Wallet** widget, you should inspect the result of the `SetOrderReferenceDetails` operation for the presence of the **PaymentMethodNotAllowed** constraint. If this constraint is returned, it means that the payment method selected by the buyer is not allowed for this Order Reference, and you will not be able to confirm the Order Reference until this constraint is resolved. To resolve this constraint, re-render the **Wallet** widget and request the buyer to select a different payment method.

Confirming the purchase details

After setting the attributes for the Order Reference object (including any additional attributes that you would like to add through the `SetOrderReferenceDetails` operation call), you must confirm the Order Reference object. This step informs Amazon that the buyer has placed the order on your site. Confirming the Order Reference object does not guarantee funds or put funds on hold. For this, you will still need to authorize the payment and capture funds.

Note: You cannot confirm an order reference if the buyer has not been able to select an address or payment instrument or if you have not set an amount. When an order reference cannot be confirmed, it will have constraints on it. If you are using an order review page, call the `GetOrderReferenceDetails` operation before rendering the order review page to check for constraints. If there are address or payment related constraints, you should disable the confirm button on your website until the buyer has picked an address and/or payment instrument. If you attempt to confirm an order reference that has constraints on it, you will get an error response showing the specific constraint violations. You should not display the order confirmation page to your buyers unless the `ConfirmOrderReference` operation was successful and without constraints. For more information on constraints, see the "GetOrderReferenceDetails" and "Order Reference Constraints" sections in the Off-Amazon Payments API section reference.

You confirm the Order Reference object by calling the `ConfirmOrderReference` operation.

The following example shows how to call the `ConfirmOrderReference` operation:

```
https://mws-eu.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01
?AWSAccessKeyId=AKIAJKYFSJU7PEXAMPLE
&Action=ConfirmOrderReference
&AmazonOrderReferenceId=S23-1234567-1234567
&MWSAuthToken=amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE
&SellerId=YOUR_SELLER_ID_HERE
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2013-11-16T19%3A01%3A11Z
&Version=2013-01-01
&Signature=CLZOdtJGjAo81IxaLoE7af6HqK0EXAMPLE
```

For more information about the `ConfirmOrderReference` operation, including the request parameters and response elements, see "ConfirmOrderReference" in the Off-Amazon Payments API section reference.

After successfully calling the `ConfirmOrderReference` operation, the Order Reference object moves from the **Draft** state to the **Open** state . For more information on state transitions, see Synchronizing your systems with Amazon Payments.

Before you confirm the Order Reference object by calling the `ConfirmOrderReference` operation, only the partial shipping address is displayed to you. You can use this partial address to compute taxes and shipping costs. Amazon recognizes that you need further details about the customer and the full shipping address to process your order and offer customer service. After you confirm an Order Reference object by calling the `ConfirmOrderRefence` operation, you should get the full shipping address by calling the `GetOrderReferenceDetails` operation, and you should verify that you can ship to that address.

The following customer information is made available to you after the Order Reference object is confirmed:

Buyer Detail	How and When to Obtain
Buyer name	Call the `GetOrderReferenceDetails` operation after you successfully call the `ConfirmOrderReference` operation.
Buyer e-mail address	Call the `GetOrderReferenceDetails` operation after you successfully call the `ConfirmOrderReference` operation.
Buyer phone number (if available)	Call the `GetOrderReferenceDetails` operation after you successfully call the `ConfirmOrderReference` operation.
Recipient name	Call the `GetOrderReferenceDetails` operation after you successfully call the `ConfirmOrderReference` operation.

Buyer Detail	How and When to Obtain
Shipping address	Partial address information (**City**, **StateOrRegion**, **PostalCode** and **CountryCode**) can be obtained by calling the `GetOrderReferenceDetails` operation immediately after an address is selected in the **AddressBook** widget. Full address information (including street information) is obtained by calling the `GetOrderReferenceDetails` operation after you successfully call the `ConfirmOrderReference` operation. **Note:** For EU, the StateOrRegion field is always empty.
Recipient phone number (if available)	Call the `GetOrderReferenceDetails` operation after you successfully call the `ConfirmOrderReference` operation.

At this point, Amazon highly recommends that you trigger your existing buyer-facing order confirmation business processes (confirmation e-mail, confirmation landing page, etc.) as the following steps of authorizing and capturing the payment method will take additional time to process.

After the Order Reference object is in the **Open** state, you can proceed with processing your buyer’s payment.

Buyer e-mail for confirmed orders

As soon as the Order Reference object is in the **Open** state, Amazon will send an e-mail to the buyer notifying them that they have agreed to make a payment to you for the total amount of the Order Reference object.

Note: An Order Reference object will remain in the **Open** state for 180 days in Production mode and 180 days in Sandbox mode if you make a successful call to the `ConfirmOrderReference` operation. After this time, the Order Reference object will move to the **Closed** state. You can request an authorization only while the Order Reference object is in the **Open** state.

Note: The "Payment Authorization" e-mail is sent to the buyer when you confirm the order reference to Amazon. This informs the buyer that they have authorized payment for a purchase made using Amazon Payments. This e-mail is not related to the `Authorize` request that you submit at the time of the order or later.

To see a sample Payment Authorized email see The Payment Authorized email.

Step 5 - Requesting an authorization

Now that you have confirmed the purchase and have successfully created an Order Reference object in the **Open** state, you can start the process of collecting the payment. You must get an authorization on the payment method that the buyer chose during checkout by first calling the `Authorize` operation to create an Authorization object.

The Authorization object

In Advanced Payment APIs, an Authorization object represents the availability of funds in the payment method(s) associated with the Order Reference object. At any time, an Authorization object can be in one of the following four states:

- **Pending** – The Authorization object functions in two modes - synchronous and asynchronous. The **Pending** status applies only if you select the asynchronous mode. For more information see Synchronous and Asynchronous Modes of Authorization. In the asynchronous mode, Amazon does not process your authorizations in real time. The Authorization object is always in the **Pending** state when you first submit the authorization request. You will receive an immediate response that indicates the operation status response from the Authorize operation that indicates this status. The Authorization object remains in the **Pending** state until it is processed by Amazon. The processing time varies and can be a minute or more. After processing is complete, Amazon will notify you of the final processing status. For more information on how to receive the final authorization status, see Step 6 - Getting the authorization status.

- **Open** – The Authorization object is in the **Open** state as soon as the payment method is successfully authorized. You can now request a capture against this authorization for up to 30 days.
- **Declined** – The Authorization object is in the **Declined** state when Amazon declines the authorization request. For more information, see Handling payment declines.
- **Closed** – The Authorization object can be in this state for a variety of reasons, including:
  - The object was in the **Open** state for 30 days in Production mode and two days in Sandbox mode.
  - You have completed a capture for the full or a partial amount against the open authorization. Amazon allows only one capture against a single authorization. An authorization will be closed immediately after your capture request is processed successfully.
  - You have marked the authorization as **Closed**.
  - You have marked the Order Reference object as **Canceled**.
  - Amazon has closed the authorization.

For more information about the Authorization object's states and the transitions among these states, see Synchronizing your systems with Amazon Payments.

The following example shows how to call the `Authorize` operation:

```
https://mws-eu.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01
?AWSAccessKeyId=AKIAFBM3LG5JEEXAMPLE
&Action=Authorize
&AmazonOrderReferenceId=S23-1234567-1234567
&AuthorizationAmount.Amount=94.50
&AuthorizationAmount.CurrencyCode=EUR
&AuthorizationReferenceId=test_authorize_1
&MWSAuthToken=amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE
&SellerAuthorizationNote=Authorization%20for%20Blue%20Shoes
&SellerId=YOUR_SELLER_ID_HERE
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2013-11-16T19%3A01%3A11Z
&TransactionTimeout=60
&Version=2013-01-01
&Signature=WlQ708aqyHXMkoUBk69Hjxj8qdh3aDcqpY71hVgEXAMPLE
```

For more information about the `Authorize` operation, including the request parameters and response elements, see the "Authorize" in the Off-Amazon Payments API section reference.

Step 6 - Getting the authorization status

The response to the `Authorize` call includes the **AuthorizationStatus** response element, which will be always be set to **Pending** if you have selected the asynchronous mode of operation. You will later receive the final status of the authorization request (for example, **Open** or **Declined**) from Amazon through the Instant Payment Notification service. For more information about the Instant Payment Notification service, see Synchronizing your systems with Amazon Payments.

You can then optionally query details of the Authorization object by calling the `GetAuthorizationDetails` operation using the **AmazonAuthorizationId** that was returned with the authorization response. You will also need this unique identifier to capture funds against a successful authorization, to process any associated notifications, and to close the authorization. For more information, see "GetAuthorizationDetails" in the Off-Amazon Payments API section reference.

Note: After you have successfully submitted the `Authorize` operation, you have 30 days to capture funds against the authorization in Production mode before it is automatically closed. In Sandbox mode you only have two days to capture funds.

Note: You can use the **TransactionTimeout** parameter in the `Authorize` request to specify the maximum time that you are willing to wait for an authorization to process. Any authorizations that are not processed in this time (that remain in the **Pending** state) will be declined by Amazon. You will receive a declined message through the Instant Payment Notification service with a reason code of *TransactionTimedOut*.

If you set the **TransactionTimeout** to 0, then the operation will always return an **Open** or **Declined** status for the Authorization. In this case, any Authorization that cannot be processed synchronously will be declined with a reason code of *TransactionTimedOut*. For more information see Synchronous and Asynchronous Modes of Authorization.

VAT registered sellers - Obtaining the Billing Address

If you are a VAT registered seller and if your Amazon Seller registration contains your VAT information, then it is possible to get the buyer's billing address after a successful call to the `Authorize` operation.

If the **AuthorizationStatus** in the `Authorize` response is in the **Open** state, or **Closed** with the **MaxCapturesProcessed** reason code, the billing address can be obtained. The billing address is not available for authorizations in a **Pending** or **Declined** state.

To obtain either the **AuthorizationStatus**, or the buyer's billing address, call the `GetAuthorizationDetails` operation. After you call the `GetAuthorizationDetails` operation, you will receive a success or failure notification. If the **AuthorizationStatus** is in the **Open** state, or **Closed** with the **MaxCapturesProcessed** reason code, the buyers billing address is provided in the **BillingAddress** element in the **AuthorizationDetails** element of the `GetAuthorizationDetails` response. The following code example includes the billing address in the response:

```
<GetAuthorizationDetailsResponse xmlns=" https://mws-eu.amazonservices.com/
schema/OffAmazonPayments/2013-01-01">
  <AuthorizationDetails>
    <AmazonAuthorizationId>
      SD0-3878800-6705015-A078460
    </AmazonAuthorizationId>
    <AuthorizationAmount>
      <CurrencyCode>GBP</CurrencyCode>
      <Amount>100.00</Amount>
    </AuthorizationAmount>
    <AuthorizationBillingAddress>
      <AddressLine1>87 Terrick Rd</AddressLine1>
      <City>EILEAN DARACH</City>
      <CountryCode>GB</CountryCode>
      <Name>Amber Kelly</Name>
      <PostalCode>IV23 2TW</PostalCode>
    </AuthorizationBillingAddress>
    <AuthorizationFee>
      <CurrencyCode>GBP</CurrencyCode>
      <Amount>0.00</Amount>
    </AuthorizationFee>
    <AuthorizationReferenceId>AuthReference7883758</AuthorizationReferenceId>
    <AuthorizationStatus>
      <State>Open</State>
      <LastUpdateTimestamp>2012-12-10T19%3A01%3A11Z</LastUpdateTimestamp>
    </AuthorizationStatus>
    <CaptureNow>false</CaptureNow>
    <CapturedAmount>
      <CurrencyCode>GBP</CurrencyCode>
      <Amount>0.00</Amount>
    </CapturedAmount>
    <CreationTimestamp>2012-12-10T19%3A01%3A11Z</CreationTimestamp>
    <ExpirationTimestamp>2013-01-10T19:10:16Z</ExpirationTimestamp>
    <SellerAuthorizationNote>Authorize Test</SellerAuthorizationNote>
  <AuthorizationDetails>
  <ResponseMetadata>
    <RequestId>b4ab4bc3-c9ea-44f0-9a3d-67cccef565c6</RequestId>
  </ResponseMetadata>
</GetAuthorizationDetailsResponse>
```

Step 7 - Requesting a capture

After you have successfully called the `Authorize` operation and the Authorization object is in the **Open** state, you can capture funds against that authorization.

The Capture object

In Advanced Payment APIs, a Capture object represents the movement of funds from the buyer to your account. Every Capture object is associated with an Authorization object that was previously created and is still valid. At any time, the Capture object can be in one of the following four states:

- **Pending** – The Capture object is the **Pending** state until it is processed by Amazon. If you submit a capture request within seven days of a successful authorization, Amazon immediately processes it. The initial response to the `Capture` operation indicates the **Completed** or **Declined** state. If you request a capture more than seven days from the date of successful authorization, Amazon will not process it immediately. The initial response to the `Capture` operation returns a **Pending** state, and it remains in this state until the request is processed by Amazon. The processing time for a capture varies and can be an hour or more. After processing is complete, Amazon will notify you of the processing status. For more information on how to receive the final capture status, see Step 8 - Getting the capture status.
- **Declined** – The Capture object is in the **Declined** state when Amazon declines the capture request. For more information, see Handling payment declines.
- **Completed** – The Capture object is in the **Completed** state when the buyer has been successfully charged.
- **Closed** – The Capture object moves to the **Closed** state if Amazon identifies a problem with the buyer's account, if the maximum amount of the capture has already been refunded, or if you have requested 10 partial refunds against the Capture object.

  The maximum amount that you can refund is the lesser of 15% or:

  - 75£ above the captured amount in the UK
  - 75€ above the captured amount in Germany

  The refund limit applies against the capture object, and not the sum of captures. No refunds are allowed against the capture after it is moved to the **Closed** state.

For more information about the Capture object’s states and the transitions among these states, see Synchronizing your systems with Amazon Payments.

To collect the funds that were reserved in the Authorization object, you must make a `Capture` operation call within 30 days of a successful authorization in Production mode and within two days in Sandbox mode. Amazon strongly recommends that you capture funds within seven days of authorization to reduce the likelihood of declines.

The following example shows how to call the `Capture` operation:

```
https://mws-eu.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01
?AWSAccessKeyId=AKIAFBM3LG5JEEXAMPLE
&Action=Capture
&AmazonAuthorizationId=S23-1234567-1234567-0000001
&CaptureAmount.Amount=199.00
&CaptureAmount.CurrencyCode=EUR
&CaptureReferenceId=test_capture_1
&MWSAuthToken=amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE
&SellerCaptureNote=For%20your%20order%20of%20Casad%20%272Jours%20Bonheur
%27%20Satchel%20in%20Sandstorm
&OrderReferenceAttributes.SellerOrderAttributes.SellerOrderId=123-XYZ
&OrderReferenceAttributes.SellerOrderAttributes.StoreName=CourtAndCherry.co.uk
&SellerId=YOUR_SELLER_ID_HERE
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2013-11-19T19%3A01%3A11Z
&Version=2013-01-01
```

```
&Signature=WlQ708aqyHXMkoUBk69Hjxj8qdh3aDcqpY71hVgEXAMPLE
```

How you set the parameters in the capture operation will affect the detail appearing in the Charge Notification email. See The Charge Notification email section under Controlling the data that appears in emails.

For more information about the `Capture` operation, including the request parameters and response elements, see "Capture" in the Off-Amazon Payments API section reference.

Step 8 - Getting the capture status

If you submit a `Capture` request within seven days of a successful authorization, Amazon immediately processes it. The initial response to your `Capture` request will indicate the **Completed** or **Declined** state. If you request a capture more than seven days from the date of a successful authorization, Amazon will not process it immediately. The initial response to your `Capture` request will return a **Pending** state and it will remain in this state until the request is processed by Amazon. The processing time varies and can be an hour or more.

After processing is complete, Amazon will notify you of the processing state through the Instant Payment Notification service. In the event that Amazon cannot complete your capture request, you will receive a **Declined** message. For more information about declines, see Handling payment declines. For more information about the Instant Payment Notification service, see Synchronizing your systems with Amazon Payments.

You can then optionally query details of the Capture object by calling the `GetCaptureDetails` operation using the **AmazonCaptureId** that was returned with the Capture response. You will also need this unique identifier to refund funds against a completed capture or to properly process associated notifications. For more information, see "GetCaptureDetails" in the Off-Amazon Payments API section reference.

Buyer e-mail for successful captures

Every time a capture is successful, Amazon sends the buyer a confirmation e-mail with details about the capture. The e-mail lists all of the Seller and Payment information for their purchase, and it informs the buyer that the Seller has charged them for their purchase. To see a sample confirmation emails see The Charge Notification email.

Step 9 - Marking the order reference as Closed

After you have charged the desired amount for the purchase and you no longer want to request any more authorizations, you should mark the Order Reference object as **Closed** by calling the `CloseOrderReference` operation. Note that once the Order Reference object is marked as **Closed**:

- You can still perform captures against any open authorizations, but you cannot create any new authorizations on the Order Reference object.
- You can still execute refunds against the Order Reference object.

For more information, see "CloseOrderReference" in the Off-Amazon Payments API section reference.

Step 10 - Requesting a refund

You can issue full or partial refunds against previously successful `Capture` operations by calling the `Refund` operation. The capture process generates an **AmazonCaptureId** that must be included in the call to the `Refund` operation.

The Refund object

In Advanced Payment APIs, a Refund object represents the movement of previously captured funds to the buyer's account from your account. At any time, the Refund object can be in one of the following three states:

- **Pending** – Amazon does not process your refunds in real time. The Refund object is always in the **Pending** state when you first submit the refund request. You will receive an initial response to the `Refund` operation that indicates this status. The Refund object remains in the **Pending** state until it is processed by Amazon. The processing time varies and can be several hours. After processing is complete, Amazon will notify you of the processing status. For more information on how to receive the final refund status, see Step 11 - Getting the refund status.

- **Declined** – The Refund object is in the **Declined** state if the request is declined by Amazon. For more information, see Handling payment declines.
- **Completed** – The Refund object is in the **Completed** state after the refund is successfully processed.

The following example shows how to call the `Refund` operation:

```
https://mws-eu.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01
?AWSAccessKeyId=AKIAFBM3LG5JEEXAMPLE
&Action=Refund
&AmazonCaptureId=S23-1234567-1234567-0000002
&MWSAuthToken=amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE
&RefundAmount.Amount=199.00
&RefundAmount.CurrencyCode=EUR
&RefundReferenceId=test_refund_1
&SellerRefundNote=For%20your%20returned%20item%2C%20Casad%20%272Jours
%20Bonheur%27%20Satchel%20in%20Sandstorm
&OrderReferenceAttributes.SellerOrderAttributes.SellerOrderId=123-XYZ
&OrderReferenceAttributes.SellerOrderAttributes.StoreName=CourtAndCherry.co.uk
&SellerId=YOUR_SELLER_ID_HERE
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2013-11-27T19%3A01%3A11Z
&Version=2013-01-01
&Signature=WlQ708aqyHXMkoUBk69Hjxj8qdh3aDcqpY71hVgEXAMPLE
```

How you set the parameters in the refund operation will affect the detail appearing in the Refund Notification email. See The Refund Notification email section under Controlling the data that appears in emails.

For more information about the `Refund` operation, including the request parameters and response elements, see "Refund" in the Off-Amazon Payments API section reference.

Step 11 - Getting the refund status

The response to the `Refund` operation includes a response element called the **RefundStatus** which will be always be set to **Pending**. You will later receive the final status of the refund request (for example, **Completed** or **Declined**) from Amazon through the Instant Payment Notification service. For more information about the Instant Payment Notification service, see Synchronizing your systems with Amazon Payments.

You can then optionally query details of the Refund object by calling the `GetRefundDetails` operation using the **AmazonRefundId** that was returned with the Refund response. For more information, see "GetRefundDetails" in the Off-Amazon Payments API section reference. The **Refund** response contains the **AmazonRefundId**, which is required for processing any associated notifications.

Buyer e-mail for successful refunds

Every time a successful refund takes place, Amazon sends the buyer a confirmation e-mail. The e-mail lists all of the Seller and payment information and it informs the buyer that a refund has been issued. To see a sample refund emails see The Refund Notification email.

Controlling data that appears in emails

How you set the parameters, when calling various operations, affects the details that appear in the emails sent to the buyer.

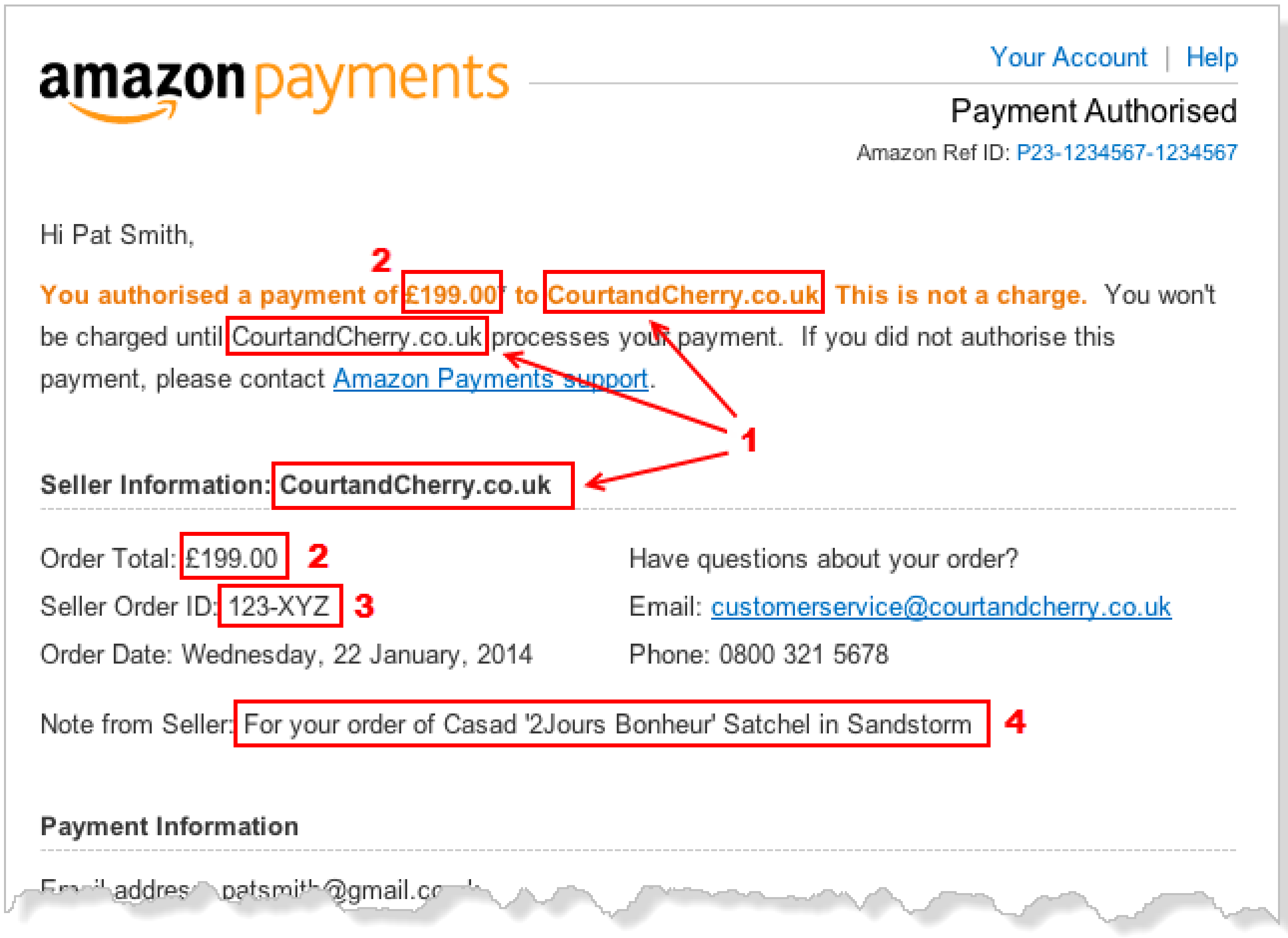
1. The Payment Authorized email displays fields from the SetOrderReferenceDetails operation.
2. The Refund or Charge Notification emails have fields that are populated by the Refund and Capture operations, respectively, as well as parameters from the Order Reference and SetOrderReferenceDetails operation.
3. A Payment Information Updated email displays parameters similarly to the Refund and Charge Notification.

Examples below show the buyer emails and the parameters that populate them.

The Payment Authorized email

As soon as your buyer authenticates with Amazon, Login and Pay with Amazon creates a system object called the Order Reference object, and you then call the `SetOrderReferenceDetails` operation to set additional attributes in the Order Reference object.

Here is a part of an Payment Authorized email that is sent to a buyer. Note the numbered fields and the corresponding parameter in the Order Reference object that is used to populate that field.



Payment Authorised

Amazon Ref ID: P23-1234567-1234567

Hi Pat Smith,

You authorised a payment of £199.00 to CourtandCherry.co.uk This is not a charge. You won't be charged until CourtandCherry.co.uk processes your payment. If you did not authorise this payment, please contact Amazon Payments support.

Seller Information: CourtandCherry.co.uk

Order Total: £199.00

Seller Order ID: 123-XYZ

Order Date: Wednesday, 22 January, 2014

Have questions about your order?

Email: customerservice@courtandcherry.co.uk

Phone: 0800 321 5678

Note from Seller: For your order of Casad '2Jours Bonheur' Satchel in Sandstorm

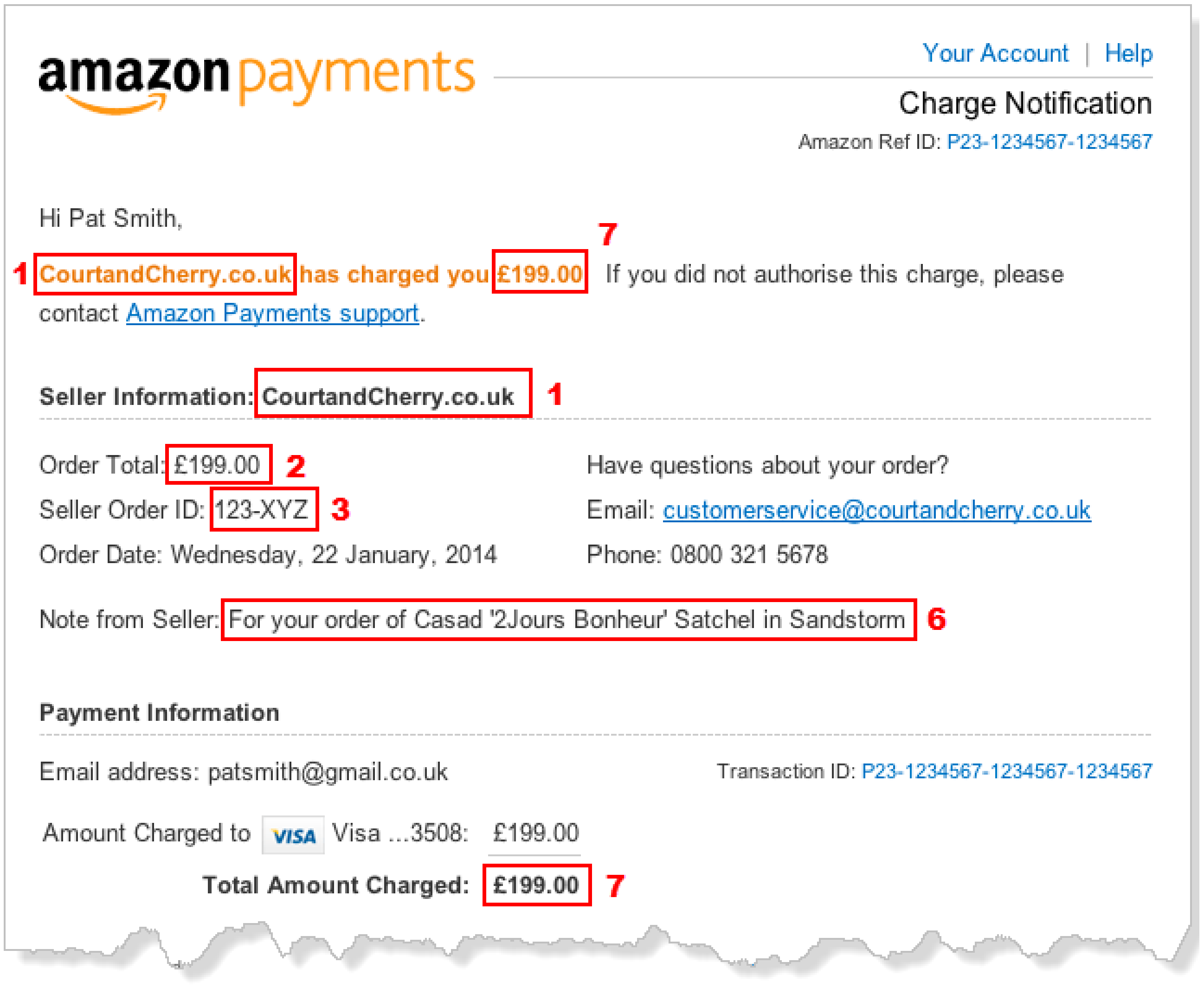
Payment Information

1	The **Seller Information:** field is populated with the Store Name:

	&OrderReferenceAttributes.SellerOrderAttributes.StoreName The store name will default to one specified during registration if none is passed in this attribute. Example: &OrderReferenceAttributes.SellerOrderAttributes.StoreName=CourtAndCherry.co.uk
2	The **Order Total:** is set by two parameters: &OrderReferenceAttributes.OrderTotal.Amount &OrderReferenceAttributes.OrderTotal.CurrencyCode Example: &OrderReferenceAttributes.OrderTotal.Amount=199 &OrderReferenceAttributes.OrderTotal.CurrencyCode=EUR
3	The **Seller Order ID:** &OrderReferenceAttributes.SellerOrderAttributes.SellerOrderId Example: &OrderReferenceAttributes.SellerOrderAttributes.SellerOrderId=123-XYZ
4	The **Note from the Seller:** &OrderReferenceAttributes.SellerNote Example: &OrderReferenceAttributes.SellerNote= For%20your%20order%20of%20Casad%20%272Jours%20Bonheur%27%20Satchel%20in %20Sandstorm

The Charge Notification email

This is an example of a Charge Notification email. Note that the Store Name, Order Total, and Seller Order ID are populated from the Order Reference, and the other values are from the call to the `Capture` operation.



Charge Notification

Amazon Ref ID: P23-1234567-1234567

Hi Pat Smith,

CourtandCherry.co.uk has charged you £199.00 If you did not authorise this charge, please contact Amazon Payments support.

Seller Information: CourtandCherry.co.uk

Order Total: £199.00

Seller Order ID: 123-XYZ

Order Date: Wednesday, 22 January, 2014

Have questions about your order?

Email: customerservice@courtandcherry.co.uk

Phone: 0800 321 5678

Note from Seller: For your order of Casad '2Jours Bonheur' Satchel in Sandstorm

Payment Information

Email address: patsmith@gmail.co.uk

Transaction ID: P23-1234567-1234567-1234567

Amount Charged to VISA Visa ...3508: £199.00

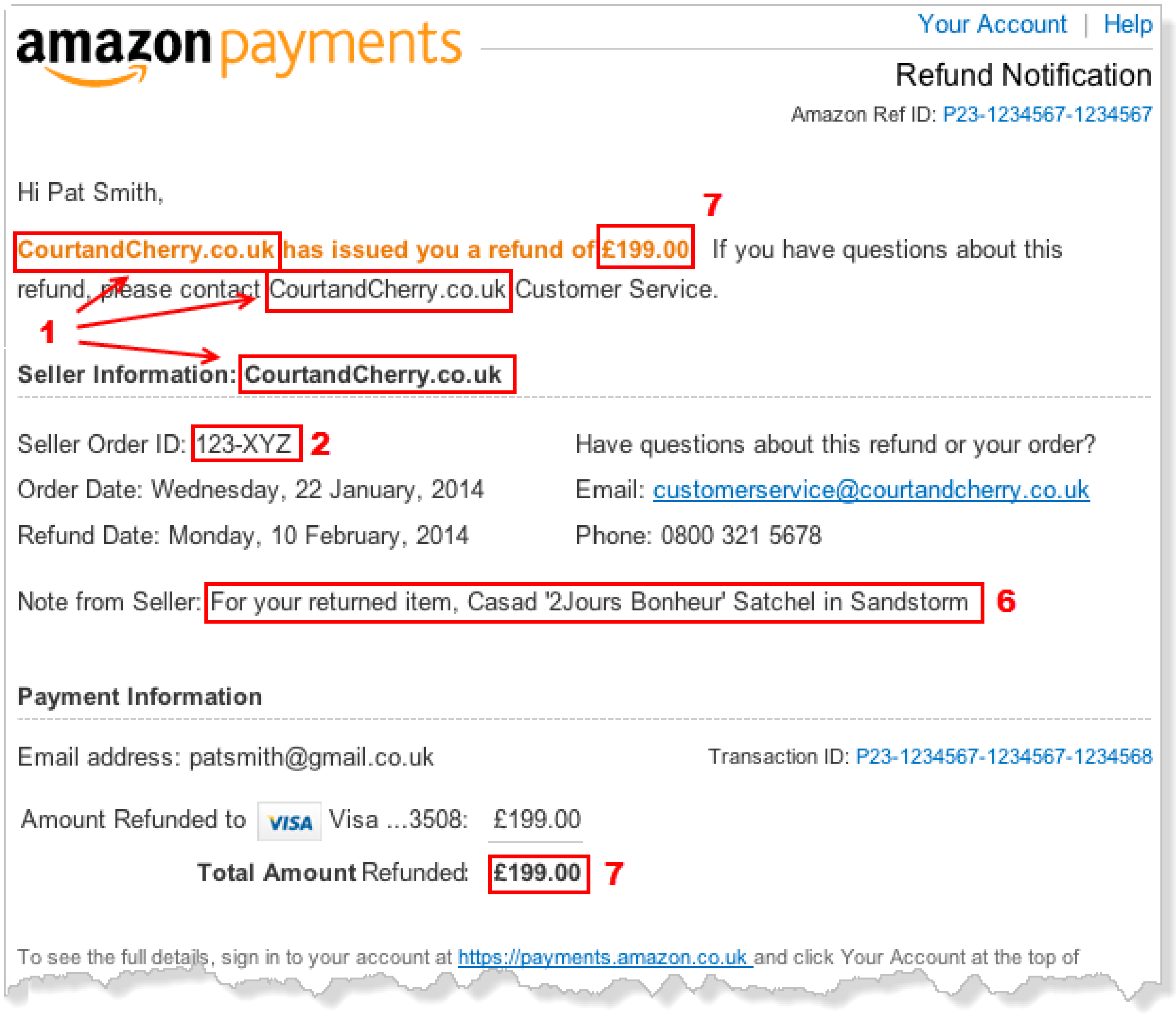
Total Amount Charged: £199.00

1	The **Seller Information:** field is populated with the Store Name: &OrderReferenceAttributes.SellerOrderAttributes.StoreName The store name will default to one specified during registration if none is passed in this attribute. Example: &OrderReferenceAttributes.SellerOrderAttributes.StoreName=CourtAndCherry.co.uk
2	The **Order Total:** is set by two parameters: &OrderReferenceAttributes.OrderTotal.Amount &OrderReferenceAttributes.OrderTotal.CurrencyCode Example: &OrderReferenceAttributes.OrderTotal.Amount=199 &OrderReferenceAttributes.OrderTotal.CurrencyCode=EUR
3	The **Seller Order ID:** &OrderReferenceAttributes.SellerOrderAttributes.SellerOrderId Example:

	&OrderReferenceAttributes.SellerOrderAttributes.SellerOrderId=123-XYZ
6	The **Note from the Seller:** &SellerCaptureNote Example: &SellerCaptureNote=For%20your%20order%20of%20Casad%20%272Jours%20Bonheur%27%20Satchel%20in%20Sandstorm
7	The **Total Amount Charged:** displays in the opening sentence of the email as well as the Total Amount Charged, and is set by two parameters: &CaptureAmount.Amount &CaptureAmount.CurrencyCode Example: &CaptureAmount.Amount=199 &CaptureAmount.CurrencyCode=EUR

The Refund Notification email

Here is a Refund Notification email. Note that the Store Name and Seller Order ID are populated from the Order Reference, and the Note from the Seller, and the Total Amount Refunded are from the call to the `Refund` operation.



Refund Notification

Amazon Ref ID: P23-1234567-1234567

Hi Pat Smith,

CourtandCherry.co.uk has issued you a refund of £199.00 If you have questions about this refund, please contact CourtandCherry.co.uk Customer Service.

Seller Information: CourtandCherry.co.uk

Seller Order ID: 123-XYZ

Order Date: Wednesday, 22 January, 2014

Refund Date: Monday, 10 February, 2014

Have questions about this refund or your order?

Email: customerservice@courtandcherry.co.uk

Phone: 0800 321 5678

Note from Seller: For your returned item, Casad '2Jours Bonheur' Satchel in Sandstorm

Payment Information

Email address: patsmith@gmail.co.uk

Transaction ID: P23-1234567-1234567-1234568

Amount Refunded to VISA Visa ...3508: £199.00

Total Amount Refunded: £199.00

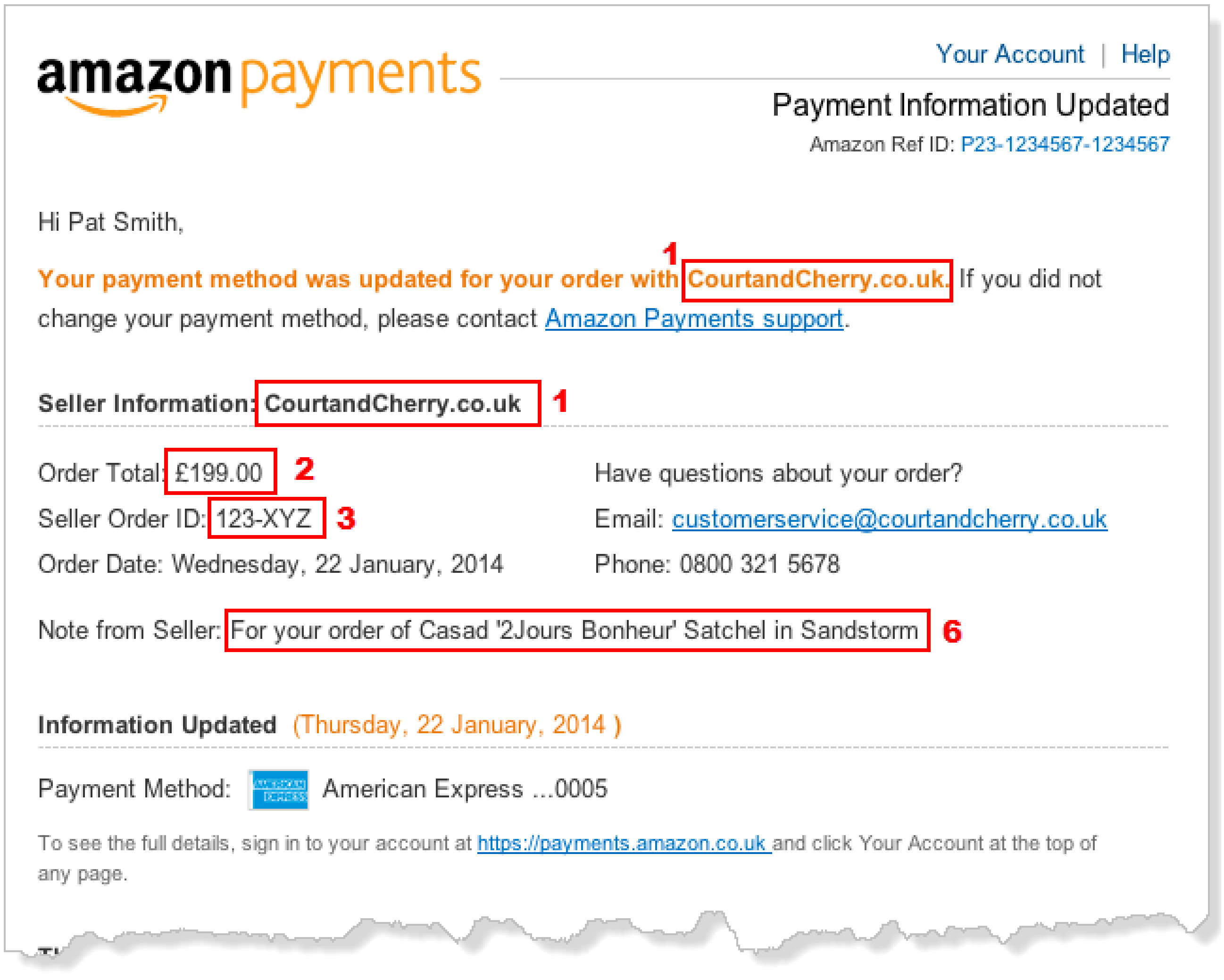
To see the full details, sign in to your account at https://payments.amazon.co.uk and click Your Account at the top of

1	The **Seller Information:** field is populated with the Store Name: &OrderReferenceAttributes.SellerOrderAttributes.StoreName The store name will default to one specified during registration if none is passed in this attribute. Example: &OrderReferenceAttributes.SellerOrderAttributes.StoreName=CourtAndCherry.co.uk
3	The **Seller Order ID:** &OrderReferenceAttributes.SellerOrderAttributes.SellerOrderId Example: &OrderReferenceAttributes.SellerOrderAttributes.SellerOrderId=123-XYZ
6	The **Note from the Seller:** **&SellerRefundNote** Example: **&SellerRefundNote**=For%20your%20returned%20item%2C%20Casad%20%272Jours%20Bonheur%27%20Satchel%20in%20Sandstorm
7	The **Total Amount Refunded:** displays in the opening sentence of the email as well as the Total Amount Refunded, and is set by two parameters: &RefundAmount.Amount

	&RefundAmount.CurrencyCode Example: &RefundAmount.Amount=199 &RefundAmount.CurrencyCode=EUR

The Payment Information Updated email

The Payment Information Updated email parameters are populated similarly where the Seller Information, Order Total, and Seller Order ID are populated from the Order Reference, and the Note from Seller is from the `Capture` operation.



Your Account | Help

Payment Information Updated

Amazon Ref ID: P23-1234567-1234567

Hi Pat Smith,

Your payment method was updated for your order with CourtandCherry.co.uk. If you did not change your payment method, please contact Amazon Payments support.

Seller Information: CourtandCherry.co.uk

Order Total: £199.00

Seller Order ID: 123-XYZ

Order Date: Wednesday, 22 January, 2014

Have questions about your order?

Email: customerservice@courtandcherry.co.uk

Phone: 0800 321 5678

Note from Seller: For your order of Casad '2Jours Bonheur' Satchel in Sandstorm

Information Updated (Thursday, 22 January, 2014)

Payment Method: American Express ...0005

To see the full details, sign in to your account at https://payments.amazon.co.uk and click Your Account at the top of any page.

1	The **Seller Information:** field is populated with the Store Name: &OrderReferenceAttributes.SellerOrderAttributes.StoreName The store name will default to one specified during registration if none is passed in this attribute. Example: &OrderReferenceAttributes.SellerOrderAttributes.StoreName=CourtAndCherry.co.uk
2	The **Order Total:** is set by two parameters: &OrderReferenceAttributes.OrderTotal.Amount

	&OrderReferenceAttributes.OrderTotal.CurrencyCode Example: &OrderReferenceAttributes.OrderTotal.Amount=199 &OrderReferenceAttributes.OrderTotal.CurrencyCode=EUR
3	The **Seller Order ID:** &OrderReferenceAttributes.SellerOrderAttributes.SellerOrderId Example: &OrderReferenceAttributes.SellerOrderAttributes.SellerOrderId=123-XYZ
6	The **Note from the Seller:** **&SellerCaptureNote** Example: **&SellerCaptureNote=**[R2] For%20your%20order%20of%20Casad%20%272Jours%20Bonheur%27%20Satchel%20in%20Sandstorm

Synchronous and Asynchronous Modes of Authorization

The `Authorize` operation functions in one of two modes:

1. Asynchronous mode – In this mode, the `Authorize` operation always returns a **Pending** status in API response and you must listen to our Instant Notification to obtain the final processing status (**Open** or **Declined**).
2. Synchronous mode – In this mode, the `Authorize` operation will always return an **Open** or **Declined** status in the API response.

For more information on the `Authorize` operation see Step 5 - Request an authorization, or Step 6 - Getting the authorization status.

Selecting the Mode of Operation

Selecting the mode of operation for the `Authorize` API depends on your business processes and technical systems.

Asynchronous mode

You can select the asynchronous mode of operation by not specifying the **TransactionTimeout** parameter, or specifying a non-zero value to the **TransactionTimeout** parameter of the `Authorize` API. For more information on non-zero values see `Authorize` in the Off-Amazon Payments API section reference.

The following code example shows how to enable the asynchronous mode for the Authorize API:

```
https://mws-eu.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01
?AWSAccessKeyId=AKIAFBM3LG5JEEXAMPLE
&Action=Authorize
&AmazonOrderReferenceId=S23-1234567-1234567
&AuthorizationAmount.Amount=94.50
&AuthorizationAmount.CurrencyCode=EUR
&AuthorizationReferenceId=test_authorize_1
&MWSAuthToken=amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE
&SellerAuthorizationNote=Authorization%20for%20Blue%20Shoes
&SellerId=YOUR_SELLER_ID_HERE
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2013-11-16T19%3A01%3A11Z
&TransactionTimeout=60
&Version=2013-01-01
&Signature=WlQ708aqyHXMkoUBk69Hjxj8qdh3aDcqpY71hVgEXAMPLE
```

You will receive an initial **Pending** status in the API response to your `Authorize` operation request, typically within 2-3 seconds. You will receive the final processing status of **Open** or **Declined** via Instant Payment Notification typically within 60 seconds. In rare cases, when Amazon is manually investigating a transaction, you may not receive the final processing status notification for up to the **TransactionTimeout** value specified by you, or for 24 hours if no value is specified. In the code example above the user has requested a **TransactionTimeout** of 60 minutes.

Amazon recommends using the asynchronous mode if your business processes and technical systems can hold an order or transaction for up to 24 hours. Since the final processing status is not available in real time, you can optimistically confirm the purchase to the buyer. If later, you receive a **Declined** status of the Authorization, you will need to notify the buyer of the failed transaction and request that they update the payment method from the Amazon Payments website, collect an alternative form of payment, or cancel the order based on the declined reason code.

For more information see Handling payment declines.

By selecting this mode, you will provide more time to Amazon to investigate transactions, usually resulting in a lower Authorization decline rate.

Synchronous mode

You can select the synchronous mode of operation by specifying the value of the **TransactionTimeout** parameter to be 0.

The following code example shows how to enable the synchronous mode for the `Authorize` API:

```
https://mws-eu.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01
?AWSAccessKeyId=AKIAFBM3LG5JEEXAMPLE
&Action=Authorize
&AmazonOrderReferenceId=S23-1234567-1234567
&AuthorizationAmount.Amount=94.50
&AuthorizationAmount.CurrencyCode=EUR
&AuthorizationReferenceId=test_authorize_1
&MWSAuthToken=amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE
&SellerAuthorizationNote=Authorization%20for%20Blue%20Shoes
&SellerId=YOUR_SELLER_ID_HERE
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2013-11-16T19%3A01%3A11Z
&TransactionTimeout=0
&Version=2013-01-01
&Signature=WlQ708aqyHXMkoUBk69Hjxj8qdh3aDcqpY71hVgEXAMPLE
```

In the synchronous mode of operation, you will receive an **Open** or **Declined** status in the API response to your `Authorize` request, typically within 6-8 seconds. If Amazon is unable to process the `Authorization` synchronously, we will return a **Declined** status with a ReasonCode of **TransactionTimeout**.

Amazon recommends using the synchronous mode if you have scenarios where you want to authorize and/or capture payments while the buyer is still on your site. For example, if you want to offer a digital download or confirm an expedited delivery.

By selecting this mode, you may observe a higher authorization decline rate as Amazon will convert some **Pending** authorizations to **Declined**. You can track these authorization declines using the ReasonCode **TransactionTimeout**.

Unlike the asynchronous mode, where you handle authorization declines through an offline process, in the synchronous mode you will be able to handle the declines while the buyer is on your site. For more information see Handling payment declines.

Integrating without the AddressBook widget

In some scenarios you might not want to collect the buyer's address from the **AddressBook** widget. You might do this if your checkout process does not require collecting a shipping address or if you want to collect the buyer's address on your own. If you want to render the **Wallet** widget without displaying the **AddressBook** widget, you must set the `useAmazonAddressBook` parameter to *false* when you create the Button widget in Step 1. You can only specify the `useAmazonAddressBook` parameter when the button is rendered. Set this parameter to *true* if you want to collect the shipping address from Amazon by using the **AddressBook** widget. If you do not specify a value for the parameter, it will default to *true* and the buyer will be required to select an address.

Note: If you do not use the Amazon **AddressBook** widget to collect an address from the buyer, the transaction will not be covered under the Amazon Payments Payment Protection Policy. For more information on Amazon Payments Payment Protection Policy, please review the Amazon Payments user agreement on the Amazon Payments web site.

The following example shows how to render the Button widget to support scenarios where you do not want to use the **AddressBook** widget:

```
<!-- put the style below inside your CSS file -->
<style type="text/css">
    #walletWidgetDiv {width: 400px; height: 228px;}
</style>
```

```
<div id="walletWidgetDiv">
</div>

<script>
var amazonOrderReferenceId;
new OffAmazonPayments.Widgets.Button ({
    sellerId: 'YOUR_SELLER_ID_HERE',
    useAmazonAddressBook: false,
    onSignIn: function(orderReference) {
        amazonOrderReferenceId = orderReference.getAmazonOrderReferenceId();
        window.location = 'https://yoursite.com/redirect_page?session=' +
            amazonOrderReferenceId;
    },
    design: {
        designMode: 'responsive'
    }
    onError: function(error) {
            // your error handling code
    }
}).bind("payWithAmazonDiv");
</script>
```

Just like in Step 2 - Add the AddressBook and Wallet widgets of the Getting Started section, you set the size of the widget via the CSS. The Wallet Widget will resize to fit the provided space. You can use media query if you want to build a responsive site where you want the widgets to resize based on the size of the browser window.

Using widgets with mobile devices

All Advanced Payment APIs widgets are optimized for mobile with no further integration changes required. The Advanced Payment APIs widgets work equally well on desktops, tablets, and smartphone devices. However, if you have a separate website that is optimized for smartphones, then Amazon recommends that you use the smartphone-optimized widgets on that website.

The smartphone-optimized widgets can be expanded and collapsed by the buyer. They are best for websites where you want to keep the space allocated to the widgets to a minimum and where you want to use the full screen area of the buyer's device. This differs from the standard widgets, which are of fixed height and width and cannot be expanded or collapsed by the buyer. The standard widgets are described in Getting started.

The smartphone-optimized **AddressBook** and **Wallet** widgets scale to fit the width of the smartphone screen. The smartphone-optimized widgets have a fixed height of 135 pixels when collapsed. The buyer can expand the widgets to change the pre-selected shipping address or payment method. In this case the widgets expand to the full height and width of the device. When the buyer makes a selection, the widgets collapse automatically and the buyer can continue checking out.

To integrate the smartphone-optimized widgets, follow these steps:

1. Add a meta tag to the head section of each of your smartphone-optimized pages. This meta tag will cause the widgets to scale to be readable on a smartphone without requiring the user to zoom the page:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0,
 maximum-scale=1.0"/>
```

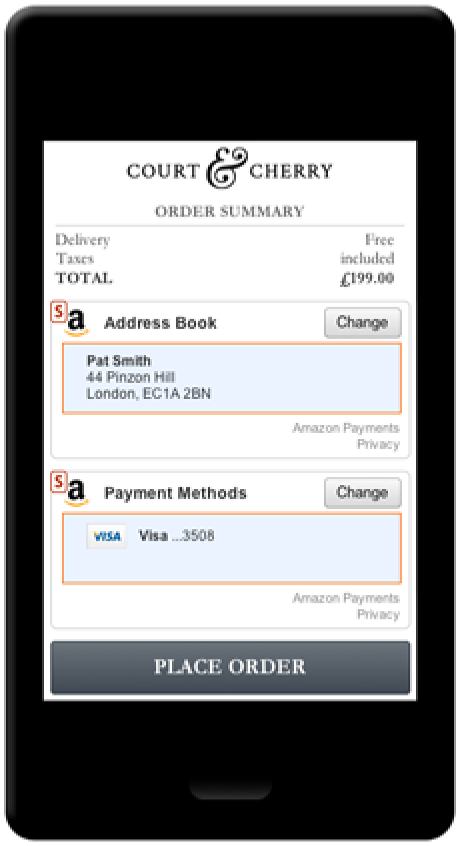
2. Modify your code that creates the **AddressBook** widget or **Wallet** widget to set the **designMode** parameter to *smartphoneCollapsible*, as in the following example:

```
new OffAmazonPayments.Widgets.AddressBook({
 sellerId : 'YOUR_SELLER_ID_HERE',
 amazonOrderReferenceId : orderReferenceId,
 design : {
  designMode: 'smartphoneCollapsible'
 },
 onAddressSelect : function(orderReference) {
 },
 onError : function(error) {
 }
}).bind("addressBookWidgetDiv");
```

The above example shows how to modify the **AddressBook** widget. You will need to make the same change for the **Wallet** widget and for the read-only **AddressBook** and **Wallet** widgets. You cannot specify the width and height of the smartphone-optimized widgets.

The following example shows how a buyer interacts with the smartphone widgets:

- Initially, the widgets are collapsed, as in Figure 1 below. The widgets show the default Shipping Address and Payment Method.
- If the buyer wants to modify the Shipping Address or Payment Method, the buyer clicks the appropriate **Change** button. The widget then expands to fill the entire screen, as in Figure 2 below.
- After the buyer selects a different Shipping Address or Payment Method, or hits the **Cancel** button, the widget collapses again, as in Figure 1.


COURT & CHERRY

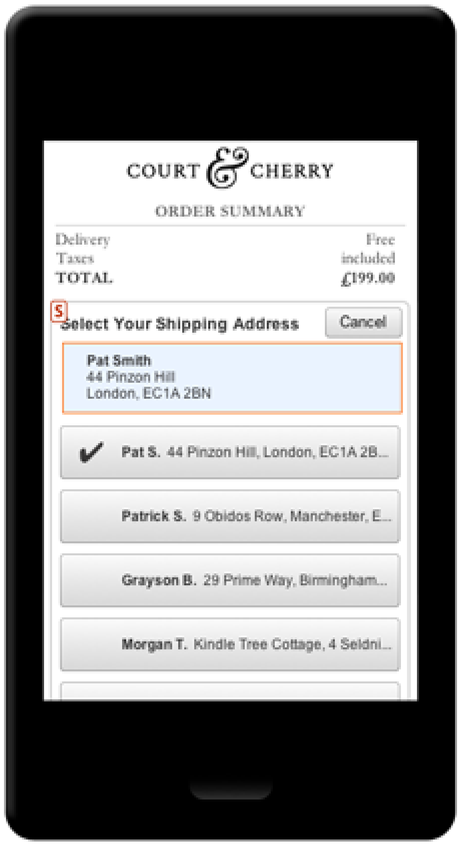
ORDER SUMMARY

Delivery	Free
Taxes	included
TOTAL	£199.00

Pat Smith
44 Pinzon Hill
London, EC1A 2BN

VISA Visa ...3508

Figure 1: Collapsed AddressBook and Wallet Widgets


COURT & CHERRY

ORDER SUMMARY

Delivery	Free
Taxes	included
TOTAL	£199.00

Pat Smith
44 Pinzon Hill
London, EC1A 2BN

✔ Pat S. 44 Pinzon Hill, London, EC1A 2B...

Patrick S. 9 Obidos Row, Manchester, E...

Grayson B. 29 Prime Way, Birmingham...

Morgan T. Kindle Tree Cottage, 4 Seldni...

Figure 2: Expanded AddressBook Widget

Handling complex payment scenarios

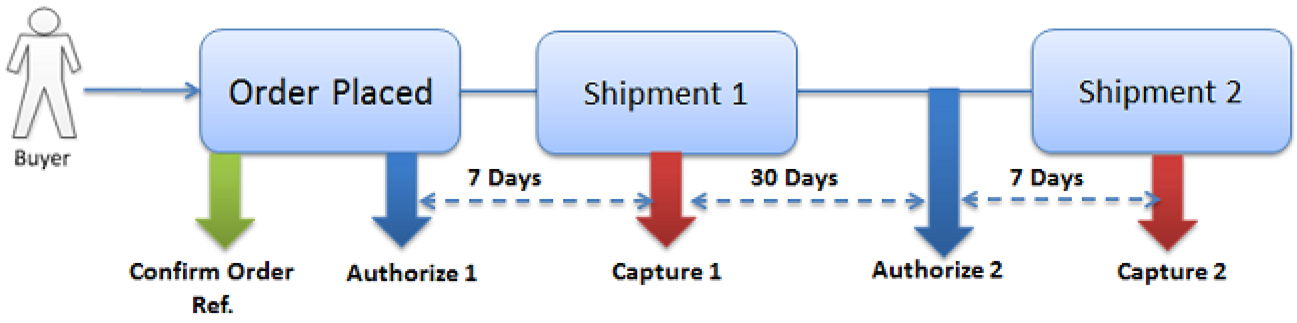
This section shows how to handle payment scenarios that are more complex than the payment scenario shown in the step-by-step walkthrough. The following payment scenarios are described:

- Split payments
- Deferred payments
- Immediate charge

Split payments

In a split payment scenario, a buyer spreads the total payment amount for the order across multiple individual payments. For example, if you provide unique, built-to-order items, you might want to get a partial payment up front and then capture the rest of the payment when the order is completed.

Another scenario where split payments are useful is where a buyer's order is sent out in multiple shipments and you capture funds as each shipment is sent out. This payment model is supported in Advanced Payment APIs as shown in the following illustration:



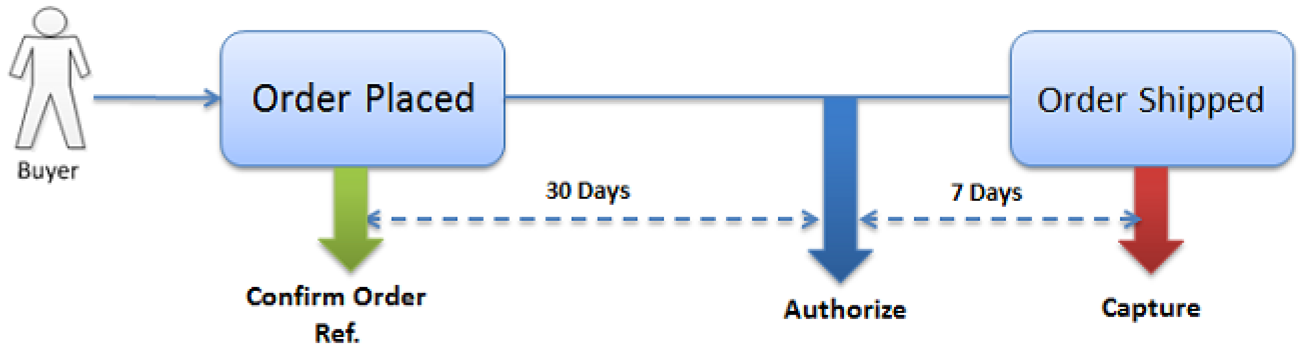
In this example, the Order Reference object is confirmed at the time of the order and an initial shipment is sent out within seven days. You can request the first capture when you send out the first shipment. After another 30 days pass, you call the `Authorize` operation again. Finally, you call the `Capture` operation a second time when the second shipment is sent to the buyer.

Advanced Payment APIs supports this split payment scenario by allowing you to submit up to 10 `Authorize` requests against the Order Reference object, allowing you to map these payment events to your internal business processes. Any **Declined** or **Closed** authorizations without any captures do not count toward the limit.

Deferred payments

In a deferred payment scenario, a buyer places an order with you online, but you do not capture a payment from them until a specific milestone is met. For example, if you allow buyers to pre-order items, or if an item is backordered, you might have a business policy that you do not collect the payment until the item ships.

Assuming that your company's policy is to collect the payment when a backordered item ships out, you do not execute the `Authorize` call until the item is available and then you call the `Capture` operation when the item is shipped. The deferred payment scenario looks something like the following illustration:



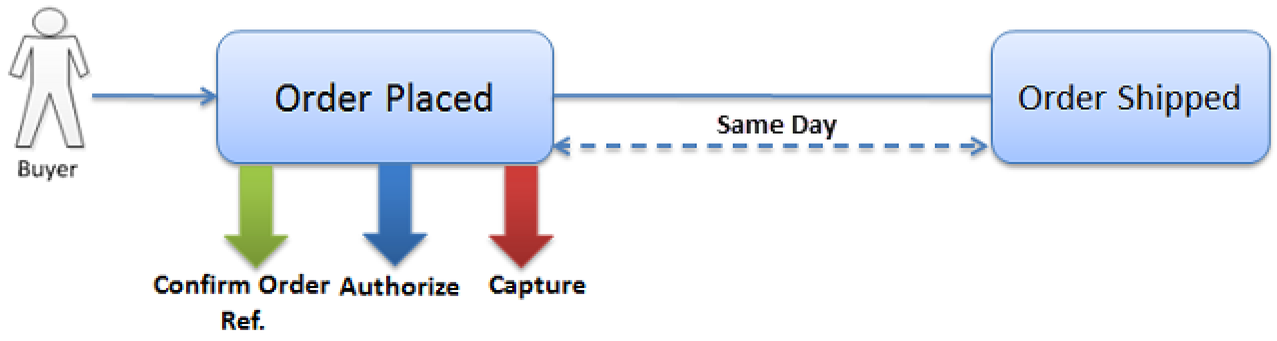
The illustration shows that the order was placed and the Order Reference object was confirmed, but the `Authorize` operation was not called until the item was available in stock and ready to be shipped. The call to the `Capture` operation is made when the pre-ordered item is shipped to the buyer.

Advanced Payment APIs supports these scenarios by keeping a confirmed Order Reference object open for 180 days in both Production mode and Sandbox mode. This 180-day period enables buyers to place their orders, but you do not have to make the `Authorize` operation call until you are ready to ship the order and collect the payment.

Immediate charge

When you want to immediately charge the buyer after the buyer places the order and you have confirmed the order reference to Amazon, you can immediately call the `Authorize` operation, and after the Authorization object moves to the **Open** state, call the `Capture` operation. Or, you can call the `Authorize` operation and set the **CaptureNow** request parameter to *true*. This approach is useful when you fulfill your items soon after taking the order, for example, on the same day.

This payment scenario looks like the following illustration:



The illustration shows that the charge occurred as soon as the buyer confirmed their order and the Order Reference is confirmed. The item is then immediately shipped to the buyer.

Note: The Amazon Payments policy states that you charge your buyer when you fulfill the items in the order. You should not collect funds prior to fulfilling the order.

Handling payment for post-purchase order modifications

Advanced Payment APIs enables you to make certain modifications to the specifics of a purchase after the buyer has successfully gone through the checkout process. The modification scenarios supported by Advanced Payment APIs include the following:

- Charging more than the original order amount
- Changing the shipping address
- Canceling or completing an order
- Partially fulfilling an order

Charging more than the original order amount

In some situations, you might need to make an adjustment and increase the total amount that you charge the buyer for the order they placed on your web site. For example, the buyer might call in to your customer service center and ask to upgrade their shipping option.

In the event that you need to charge more than the original order amount, Advanced Payment APIs enables you to obtain authorization to charge an amount that is up to 15% or 75 € (or 75 £), whichever is less, above the amount originally specified in the open Order Reference object.

Sample Scenario – Payment Amount Increase

To illustrate this payment increase scenario, assume your buyer has gone through the entire checkout process and confirmed an order of 200 €, including the cost of the items plus taxes and shipping. Before the order is shipped, the buyer calls your customer service center and asks to upgrade their shipping to a more expensive option, which increases the total cost to 230 €.

In this scenario, you call the `Authorize` operation for 230 €, or you call the `Authorize` operation for 30 € if the original amount of 200 € was already authorized. The 230 € amount is allowed because 30 € is a 15% increase over the original Order Reference object amount.

Note: Amazon recommends that you secure an authorization from the buyer before you charge an amount higher than the original order amount to reduce the likelihood of charge backs and A-to-z claims. Amazon also recommends that you send an e-mail to the buyer with the updated order information and that you keep a record of this e-mail.

Changing the shipping address

You do not need to notify Amazon if your buyer contacts you and requests that a shipping address be used for their order that is different from the one provided through the Amazon Payments **AddressBook** widget. You can adjust the address in your own order fulfillment process without passing any information back to Amazon about the change.

Note: If you use a shipping address that was not provided by Amazon through the **AddressBook** widget, you will not be covered by the Amazon Payment Protection Policy. Please refer to the payment protection policy for more information on how shipping address changes will impact your coverage.

Canceling or completing an order

If you need to cancel an order, or you have completed fulfilling an order, you might want to either cancel or close an Order Reference object to support your company's bookkeeping or other internal processes that track order status.

Advanced Payment APIs enables you to use either the `CancelOrderReference` or `CloseOrderReference` operation to change the status of the Order Reference object.

The following list describes the differences between calling the `CloseOrderReference` and `CancelOrderReference` operations:

- `CloseOrderReference` - You call this operation to mark an Order Reference object in the **Open** or **Suspended** state as **Closed**. You do this when you have charged the full amount for the order and you no longer want to request any more authorizations. In the event that you are only able to partially fulfill the order, you should still mark the Order Reference object as **Closed** after charging the appropriate amount. Amazon uses this information to indicate to the buyer that they have completed payment for their order. When the Order Reference object is closed by calling the `CloseOrderReference` operation:
  - You can still perform captures against any open authorizations, but you cannot create any new authorizations on the Order Reference object.
  - You can still execute refunds against the Order Reference object.
- `CancelOrderReference` - You call this operation to mark an Order Reference object in the **Draft**, **Open**, or **Suspended** state as **Canceled** as long as there are no **Pending**, **Completed**, or **Closed** captures against it. You do this when you have canceled the full order and do not want to charge any amount for it. Amazon uses this information to indicate to the buyer that their payment for this order has been canceled. When the Order Reference object is canceled by calling the `CancelOrderReference` operation:
  - You cannot create any new authorizations on the Order Reference object and any existing authorizations that are in the **Open** or **Pending** states are closed.
  - **Note:** You can only cancel an Order Reference object if there are no **Completed**, **Closed**, or **Pending** captures against it.

The `CloseOrderReference` operation

The `CloseOrderReference` operation is useful when you have completed fulfilling the buyer's order, charged the appropriate amount for it, and want to mark the Order Reference object as **Closed** to help synchronize your internal order status with the status of the Order Reference object.

The following example shows how to call the `CloseOrderReference` operation:

```
https://mws-eu.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01
?AWSAccessKeyId=AKIAJKYFSJU7PEXAMPLE
&Action=CloseOrderReference
&AmazonOrderReferenceId=S23-1234567-1234567
&MWSAuthToken=amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE
&SellerId=YOUR_SELLER_ID_HERE
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2013-12-19T19%3A01%3A11Z
&Version=2013-01-01
&Signature=CLZOdtJGjAo81IxaLoE7af6HqK0EXAMPLE
```

For more information about the `CloseOrderReference` operation, including the request parameters and response elements, see "CloseOrderReference" in the Off-Amazon Payments API section reference.

In addition to the response, you will receive a notification from the Instant Payment Notification service. For more information about the Instant Payment Notification service, see Synchronizing your systems with Amazon Payments.

The `CancelOrderReference` operation

The `CancelOrderReference` operation is useful when you have an order that needs to be canceled and you want to ensure that no payment actions are performed against the Order Reference object.

The following example shows how to call the `CancelOrderReference` operation:

```
https://mws-eu.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01
```

```
?AWSAccessKeyId=AKIAJKYFSJU7PEXAMPLE
&Action=CancelOrderReference
&AmazonOrderReferenceId=S23-1234567-1234567
&MWSAuthToken=amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE
&SellerId=YOUR_SELLER_ID_HERE
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2013-12-19T19%3A01%3A11Z
&Version=2013-01-01
&Signature=CLZOdtJGjAo81IxaLoE7af6HqK0EXAMPLE
```

For more information about the `CancelOrderReference` operation, including the request parameters and response elements, see the "CancelOrderReference" in the Off-Amazon Payments API section reference.

In addition to the response, you will receive a notification from the Instant Payment Notification service. You will also receive notifications for any open authorizations against the Order Reference object. For more information about the Instant Payment Notification service, see Synchronizing your systems with Amazon Payments.

Partially fulfilling an order

In the event that you only partially fulfill an order, you can mark the Order Reference as closed by calling the `CloseOrderReference` operation after charging the appropriate amount. In addition, you can also close any outstanding authorizations that you do not intend to use using the `CloseAuthorization` operation. Marking an order reference and authorization as closed will enable you to synchronize your internal order states with Amazon.

The following example shows how to call the `CloseAuthorization` operation:

```
https://mws-eu.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01
?AWSAccessKeyId=AKIAFBM3LG5JEEXAMPLE
&Action=CloseAuthorization
&AmazonAuthorizationId=S23-1234567-1234567-0000001
&ClosureReason=Closing%20the%20auhorization%20transaction
&MWSAuthToken=amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE
&SellerId=YOUR_SELLER_ID_HERE
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2013-12-19T19%3A01%3A11Z
&Version=2013-01-01
&Signature=WlQ708aqyHXMkoUBk69Hjxj8qdh3aDcqpY71hVgEXAMPLE
```

For more information about the `CloseAuthorization` operation, including the request parameters and response elements, see the "CloseAuthorization" in the Off-Amazon Payments API section reference.

Note: Amazon recommends that you always close your order reference after you have processed the entire payment for the order. Otherwise, when buyers review their transactions on the Amazon Payments web site, these order references will be shown as open for 180 days.

Handling payment declines

It is possible that payments will be declined during authorizations, captures, or refunds. Advanced Payment APIs has several business rules to help you address these decline situations and work with buyers to update or change their payment method. The following situations are described:

- Declined authorizations
- Declined captures
- Declined refunds

Declined authorizations

If the `Authorize` operation call is declined, you will see one of the following four reason codes included in the response:

- **InvalidPaymentMethod** – Indicates that there was a problem with the payment method selected. In this situation, the order reference moves to the **Suspended** state and an Instant Notification will be sent to you. Amazon recommends that you do the following:
    - Asynchronous mode

      You will have to handle this scenario with the assumption that the buyer has already left the website.

        - Contact the buyer via an e-mail or phone call.
        - Have your buyer visit the UK Amazon Payments web site, or DE Amazon Payments web site and look up their order. From here, they can update the payment method by following the instructions on the web page.
        - After the buyer updates his or her payment method, the Order Reference object moves from the **Suspended** state to the **Open** state. Advanced Payment APIs will then send you an Instant Payment Notification and the buyer will receive an e-mail informing them that the payment method was updated.
    - Synchronous mode

      Amazon recommends that you handle this decline while the buyer is still completing the check out flow on your website.

        - Within your purchase flow, on your website, display an error message to notify the buyer that the payment method they selected was declined.
        - Redisplay the Amazon Payment method widget on your page. When you do, Amazon will mark the previously selected payment method as **Declined**. You can provide text indicating that the buyer can click the **Declined** link to update their payment information for that payment method or to select an alternate payment method from the widget.
        - After the buyer updates their payment method or selects a new payment method, Amazon will trigger the `onPaymentSelect` callback from the Payment widget.
        - The buyer may click several payment methods in the widget before making the final selection. Once buyer has made the final selection by clicking the **Place Order**, **Update** or similar button as per the UX flow, call the `ConfirmOrderReference` API. This will move the Order Reference from the **Suspended** to the **Open** state. You will receive an Instant Payment Notification indicating the status change or you may call `GetOrderReference` to retrieve the updated order status.
        - Once the order is in the **Open** state, you may attempt a new authorization using the `Authorize` API.
        - The buyer will receive an e-mail informing them that the payment method was updated.
- **AmazonRejected** – Indicates that the `Authorize` operation call was declined due to a determination made by Amazon. You can retry the `Authorize` operation request only if the Order Reference is still in the **Open** state.
- **ProcessingFailure** - Indicates that Amazon could not process the transaction due to an internal processing error. Please retry your request in one to two minutes.
- **TransactionTimedOut** – In **asynchronous mode**, indicates that the `Authorize` operation call was not processed within the default timeout period of 24 hours or within the time period specified by you in the

TransactionTimeout request parameter. In **synchronous mode**, indicates that Amazon could not process your request within 6 - 8 seconds.

If you are observing a high number of declines due to this reason code, try adjusting the timeout value in asynchronous mode or consider using asynchronous mode if you are using synchronous mode. An alternate approach for handling this error in synchronous mode is to retry the transaction in asynchronous mode.

Declined captures

If the `Capture` operation call is declined, you will see one of the following two reason codes included in the response:

- **AmazonRejected** - Indicates that the `Capture` operation call was declined due to a determination made by Amazon. It is likely that the Authorization object and the Order Reference object will also be closed when Amazon rejects a capture. If the authorization is closed, any further capture requests will also be declined. If the Order Reference is still in the **Open** state, request a new authorization, and then request a capture on the authorization.
- **ProcessingFailure** - Indicates that Amazon could not process the transaction due to an internal processing error. Please retry your request in one to two minutes.

In the case that the Order Reference object is still open, Amazon recommends submitting a new `Authorize` operation call, followed by a new `Capture` operation call.

Declined refunds

If the `Refund` operation call is declined, you will see one of the following two reason codes included in the response:

- **AmazonRejected** – Indicates that the `Refund` operation call was declined due to a determination made by Amazon. In this case, you should work with the buyer to issue a refund in an alternate way, for example, to issue a store credit.
- **ProcessingFailure** - Indicates one of the following:
    - Amazon could not process the transaction due to an internal error.
    - The buyer has already received a refund from an A-to-z claim or from a chargeback.

  If you receive a **ProcessingFailure** reason code in the response, you should first check the A-to-z claim and chargeback status for your order in Seller Central, under the **Performance** tab. If no A-to-z claim or chargeback has been issued to the buyer, and if the Capture object is still in the **Completed** state, retry your request in one to two minutes. If the retry attempt does not succeed, contact the buyer and issue the refund in an alternate way.

Synchronizing your systems with Amazon Payments

Payment object state transitions

The Advanced Payment APIs Payments objects (Order Reference object, Authorization object, Capture object, and Refund object) transition to various states as a part of the payment process. These state transitions can take place as a result of certain operations performed by your system or by internal Amazon business rules. For example, an Authorization object requested in asynchronous mode first goes to the **Pending** state while it is being processed. After the processing is complete, it moves to the **Open** state, which indicates that it is ready for capturing funds. After the Authorization object is captured, it moves to the **Closed** state, which indicates that no funds can be captured against that authorization. Alternatively, if you do not capture an authorization within 30 days, Amazon will mark it as **Closed**. You should synchronize your system with the current state of a payment object in Amazon's system to prevent integration errors.

For detailed information about payment object state transitions, see the Off-Amazon Payments API section reference.

Asynchronous nature of payment operations

In asynchronous mode, Amazon does not process `Authorize` and `Refund` requests in real time. The initial response to these calls always returns the state as **Pending**. A capture is processed in real time if it is requested within seven days of authorization. During this period, the synchronous response of the `Capture` request will return the state as **Completed** or **Declined**. If a capture is requested more than seven days after the authorization, the `Capture` operation will return the **Pending** state. These objects remain in the **Pending** state until Amazon processes the request.

The table below shows the initial response to API calls for the `Authorize`, `Capture`, and `Refund` API calls.

API Call	Initial Response to the API Calls
Authorize	**Synchronous Mode:** Either **Open** or **Declined** **Async Mode:** Always **Pending**
Capture	Either **Completed**, **Pending**, or **Declined**
Refund	Always **Pending**

After the processing is complete, Amazon sends an asynchronous Instant Payment Notification to inform you of the processing result. You can then query the status and details of these payment objects using the Off-Amazon Payments API section operations described below.

Monitoring payment object state transitions

You can use Instant Payment Notifications (IPNs) to monitor the state transition of payment objects. IPNs will automatically inform you of the change in state of a payment object. After you have received an Instant Payment Notification, you can poll the service using the `GetOrderReferenceDetails`, `GetAuthorizationDetails`, `GetCaptureDetails`, or `GetRefundDetails` operations to get full details about a payment object.

Instant Notifications example

Amazon sends you a notification when the state of any of the payment objects or the Order Reference object changes. These notifications are always sent without any action required on your part and can be used to update any internal tracking or fulfillment systems you might be using to manage the order.

You can set up your Notification endpoints in Seller Central by accessing the **Integration Settings** page in the **Settings** tab.

For example, you might get a notification from Amazon indicating that an Authorization object has transitioned from the **Pending** state to the **Open** state. This transition indicates that the `Authorize` operation call was successful and that you can now proceed with the order fulfillment process, knowing that the payment was successfully authorized.

With each notification you receive, you should configure your endpoint to send Amazon a "200 OK" response immediately after receipt. If you do not send this response or if your server is down when the SNS message is sent, Amazon SNS will perform retries every hour for 14 days.

For more information about the format of the Instant Notification, see HTTP/HTTPS Notification JSON Format in the Amazon SNS Getting Started Guide, which is available from the Amazon AWS Documentation portal.

The contents of the **NotificationData** member that is returned in the Instant Notification are described by the following publicly available XSD: https://amazonpayments.s3.amazonaws.com/documents/payments_ipn.xsd.

All Instant Notifications received from Amazon are signed. For more information on how to verify the signature, see Verifying the Signatures of Amazon SNS Messages in the Amazon SNS Getting Started Guide, which is available from the Amazon AWS Documentation portal.

The following is a list of notifications available from Amazon:

- **OrderReferenceNotification**
- **AuthorizationNotification**
- **CaptureNotification**
- **RefundNotification**

The following example shows the **OrderReferenceNotification**:

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 432f33bf-9f84-5004-815f-7a6cfEXAMPLE
x-amz-sns-topic-arn: arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic
x-amz-sns-subscription-arn:
    arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic: EXAMPLE
Content-Length: 961
Content-Type: text/plain; charset=UTF-8
Host: ec2-EXAMPLE.compute-1.amazonaws.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent
{
  "Type" : "Notification",
  "MessageId" : "cf5543af-dd65-5f74-8ccf-0a410EXAMPLE",
  "TopicArn" : "arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic",
  "Message" :
    "{\"NotificationReferenceId\":\"32d195c3-a829-4222-b1e2-14ab2EXAMPLE\",
      \"NotificationType\":\"OrderReferenceNotification\",
      \"SellerId\":\"YOUR_SELLER_ID_HERE\",
      \"ReleaseEnvironment\":\"Sandbox\",
      \"Version\":\"2013-01-01\",
      \"NotificationData\":
        \"<?xml version=\\\"1.0\\\" encoding=\\\"UTF-8\\\"?>
          <OrderReferenceNotification
            xmlns=\\\"https://mws-eu.amazonservices.com/
                      ipn/OffAmazonPayments/2013-01-01\\\">\\n
          <OrderReference>\\n
          <AmazonOrderReferenceId>
            S23-1234567-1234567
          <\\/AmazonOrderReferenceId>\\n
          <OrderTotal>\\n
          <Amount>106.00<\\/Amount>\\n
          <CurrencyCode>EUR<\\/CurrencyCode>\\n
          <\\/OrderTotal>\\n
          <OrderReferenceStatus>\\n
          <State>CLOSED<\\/State>\\n
          <ReasonCode>SellerClosed<\\/ReasonCode>\\n
          <LastUpdateTimestamp>
            2013-04-01T10:49:59.532Z
          <\\/LastUpdateTimestamp>\\n
          <\\/OrderReferenceStatus>\\n
          <CreationTimestamp>2013-03-30T09:58:51.234Z<\\/CreationTimestamp>\
\n
          <ExpirationTimestamp>
            2013-04-06T09:58:51.234Z
          <\\/ExpirationTimestamp>\\n
          <\\/OrderReference>\\n
          <\\/OrderReferenceNotification>\",
      \"Timestamp\":\"2013-04-22T06:00:14Z\"}",
  "Timestamp" : "2013-04-22T06:00:15.108Z",
  "SignatureVersion" : "1",
  "Signature" : "deako5R0...CVmPQOI=",
  "SigningCertURL" : "https://sns.EXAMPLE.amazonaws.com/
    SimpleNotificationService-f3ecfb7224c7233fe7bb5f59fEXAMPLE.pem",
  "UnsubscribeURL" : "https://sns.EXAMPLE.amazonaws.com/
    ?Action=Unsubscribe
    &SubscriptionArn=arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic:GUID"
}
```

The following example shows the **AuthorizationNotification**:

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 6f7e123e-49c9-5c9d-a389-5bed0EXAMPLE
x-amz-sns-topic-arn: arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic
x-amz-sns-subscription-arn:
    arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic: EXAMPLE
Content-Length: 961
Content-Type: text/plain; charset=UTF-8
Host: ec2-EXAMPLE.compute-1.amazonaws.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent
{
  "Type" : "Notification",
  "MessageId" : "cf5543af-dd65-5f74-8ccf-0a410EXAMPLE",
  "TopicArn" : "arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic",
  "Message" :
    "{\"NotificationReferenceId\":\"32d195c3-a829-4222-b1e2-14ab28909513\",
      \"NotificationType\":\"PaymentAuthorize\",
      \"SellerId\":\"YOUR_SELLER_ID_HERE\",
      \"ReleaseEnvironment\":\"Sandbox\",
      \"Version\":\"2013-01-01\",
      \"NotificationData\":
      \"<?xml version=\\\"1.0\\\" encoding=\\\"UTF-8\\\"?>
        <AuthorizationNotification
          xmlns=\\\"https://mws-eu.amazonservices.com/
                    ipn/OffAmazonPayments/2013-01-01\\\">\\n
        <AuthorizationDetails>\\n
        <AmazonAuthorizationId>
          S23-1234567-1234567-0000001
        <\\/AmazonAuthorizationId>\\n
        <AuthorizationReferenceId>
          9bbe88cd5ab4435b85d717fd8EXAMPLE
        <\\/AuthorizationReferenceId>\\n
        <AuthorizationAmount>\\n
        <Amount>5.0<\\/Amount>\\n
        <CurrencyCode>EUR<\\/CurrencyCode>\\n
        <\\/AuthorizationAmount>\\n
        <CapturedAmount>\\n
        <Amount>0.0<\\/Amount>\\n
        <CurrencyCode>EUR<\\/CurrencyCode>\\n
        <\\/CapturedAmount>\\n
        <AuthorizationFee>\\n
        <Amount>0.0<\\/Amount>\\n
        <CurrencyCode>EUR<\\/CurrencyCode>\\n
        <\\/AuthorizationFee>\\n
        <IdList/>\\n
        <CreationTimestamp>2013-04-22T05:59:38.186Z<\\/CreationTimestamp>\\n
        <ExpirationTimestamp>
          2013-05-22T05:59:38.186Z
        <\\/ExpirationTimestamp>\\n
        <AuthorizationStatus>\\n
        <State>Open<\\/State>\\n
        <LastUpdateTimestamp>
          2013-04-22T06:00:11.473Z
        <\\/LastUpdateTimestamp>\\n
        <\\/AuthorizationStatus>\\n
        <OrderItemCategories/>\\n
        <CaptureNow>false<\\/CaptureNow>\\n
        <SoftDescriptor/>\\n
        <\\/AuthorizationDetails>\\n
        <\\/AuthorizationNotification>\",
      \"Timestamp\":\"2013-04-22T06:00:14Z\"}",
  "Timestamp" : "2013-04-22T06:00:15.108Z",
  "SignatureVersion" : "1",
  "Signature" : "W/cfaDzC...5glwqJk=",
  "SigningCertURL" : "https://sns.EXAMPLE.amazonaws.com/
    SimpleNotificationService-f3ecfb7224c7233fe7bb5f59fEXAMPLE.pem",
  "UnsubscribeURL" : "https://sns.EXAMPLE.amazonaws.com/
    ?Action=Unsubscribe
    &SubscriptionArn=arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic:GUID"
}
```

The following example shows the **CaptureNotification**:

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 64f5f75c-5799-53e5-b4c3-be8f1EXAMPLE
x-amz-sns-topic-arn: arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic
x-amz-sns-subscription-arn:
    arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic: EXAMPLE
Content-Length: 961
Content-Type: text/plain; charset=UTF-8
Host: ec2-EXAMPLE.compute-1.amazonaws.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent
{
  "Type" : "Notification",
  "MessageId" : "cf5543af-dd65-5f74-8ccf-0a410EXAMPLE",
  "TopicArn" : "arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic",
  "Message" :
    "{\"NotificationReferenceId\":\"32d195c3-a829-4222-b1e2-14ab2EXAMPLE\",
      \"NotificationType\":\"PaymentCapture\",
      \"SellerId\":\"YOUR_SELLER_ID_HERE\",
      \"ReleaseEnvironment\":\"Sandbox\",
      \"Version\":\"2013-01-01\",
      \"NotificationData\":
        \"<?xml version=\\\"1.0\\\" encoding=\\\"UTF-8\\\"?>
          <CaptureNotification
            xmlns=\\\"https://mws-eu.amazonservices.com/
                     ipn/OffAmazonPayments/2013-01-01\\\">\\n
          <CaptureDetails>\\n
          <AmazonCaptureId>S23-1234567-1234567-0000002<\\/AmazonCaptureId>\\n
          <CaptureReferenceId>
            6f4d9dea0c234279a65e77994EXAMPLE
          <\\/CaptureReferenceId>\\n
          <CaptureAmount>\\n
          <Amount>5.0<\\/Amount>\\n
          <CurrencyCode>EUR<\\/CurrencyCode>\\n
          <\\/CaptureAmount>\\n
          <RefundedAmount>\\n
          <Amount>0.0<\\/Amount>\\n
          <CurrencyCode>EUR<\\/CurrencyCode>\\n
          <\\/RefundedAmount>\\n
          <CaptureFee>\\n
          <Amount>0.0<\\/Amount>\\n
          <CurrencyCode>EUR<\\/CurrencyCode>\\n
          <\\/CaptureFee>\\n
          <IdList/>\\n
          <CreationTimestamp>2013-04-22T06:02:22.026Z<\\/CreationTimestamp>\
\n
          <CaptureStatus>\\n
          <State>Completed<\\/State>\\n
          <LastUpdateTimestamp>
            2013-04-22T06:02:25.227Z
          <\\/LastUpdateTimestamp>\\n
          <\\/CaptureStatus>\\n
          <SoftDescriptor>AMZ*softdescriptor<\\/SoftDescriptor>\\n
          <\\/CaptureDetails>\\n
          <\\/CaptureNotification>\",
      \"Timestamp\":\"2013-04-22T06:00:14Z\"}",
  "Timestamp" : "2013-04-22T06:00:15.108Z",
  "SignatureVersion" : "1",
  "Signature" : "dUWd9lrs...iNGKnR4=",
  "SigningCertURL" : "https://sns.EXAMPLE.amazonaws.com/
    SimpleNotificationService-f3ecfb7224c7233fe7bb5f59fEXAMPLE.pem",
  "UnsubscribeURL" : "https://sns.EXAMPLE.amazonaws.com/
    ?Action=Unsubscribe
    &SubscriptionArn=arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic:GUID"
}
```

The following example shows the **RefundNotification**:

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 5f43584c-1f96-5880-9c98-119f5EXAMPLE
x-amz-sns-topic-arn: arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic
x-amz-sns-subscription-arn:
    arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic: EXAMPLE
Content-Length: 961
Content-Type: text/plain; charset=UTF-8
Host: ec2-EXAMPLE.compute-1.amazonaws.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent
{
  "Type" : "Notification",
  "MessageId" : "cf5543af-dd65-5f74-8ccf-0a410EXAMPLE",
  "TopicArn" : "arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic",
  "Message" :
    "{\"NotificationReferenceId\":\"32d195c3-a829-4222-b1e2-14ab2EXAMPLE\",
    \"NotificationType\":\"PaymentRefund\",
    \"SellerId\":\"YOUR_SELLER_ID_HERE\",
    \"ReleaseEnvironment\":\"Sandbox\",
    \"Version\":\"2013-01-01\",
    \"NotificationData\":
      \"<?xml version=\\\"1.0\\\" encoding=\\\"UTF-8\\\"?>
```

```
         <RefundNotification
           xmlns=\\\"https://mws-eu.amazonservices.com/
                     ipn/OffAmazonPayments/2013-01-01\\\">\\n
         <RefundDetails>\\n
         <AmazonRefundId>S23-1234567-1234567-0000003<\\/AmazonRefundId>\\n
         <RefundReferenceId>
           07fff0c4e05046958db7e47607e7db17
         <\\/RefundReferenceId>\\n
         <RefundType>SellerInitiated<\\/RefundType>\\n
         <RefundAmount>\\n
         <Amount>5.0<\\/Amount>\\n
         <CurrencyCode>EUR<\\/CurrencyCode>\\n
         <\\/RefundAmount>\\n
         <FeeRefunded>\\n
         <Amount>0.0<\\/Amount>\\n
         <CurrencyCode>EUR<\\/CurrencyCode>\\n
         <\\/FeeRefunded>\\n
         <CreationTimestamp>2013-04-22T06:07:34.617Z<\\/CreationTimestamp>\\n
         <RefundStatus>\\n
         <State>Completed<\\/State>\\n
         <LastUpdateTimestamp>
           2013-04-22T06:09:20.178Z
         <\\/LastUpdateTimestamp>\\n
         <\\/RefundStatus>\\n
         <SoftDescriptor>AMZ*softDescriptor<\\/SoftDescriptor>\\n
         <\\/RefundDetails>\\n
         <\\/RefundNotification>\",
     \"Timestamp\":\"2013-04-22T06:00:14Z\"}",
   "Timestamp" : "2013-04-22T06:00:15.108Z",
   "SignatureVersion" : "1",
   "Signature" : "kjac14DH...oQT6FbA=",
   "SigningCertURL" : "https://sns.EXAMPLE.amazonaws.com/
     SimpleNotificationService-f3ecfb7224c7233fe7bb5f59fEXAMPLE.pem",
   "UnsubscribeURL" : "https://sns.EXAMPLE.amazonaws.com/
     ?Action=Unsubscribe
     &SubscriptionArn=arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic:GUID"
}
```

Polling API example

The following example shows how to call the `GetAuthorizationDetails` operation to get the status of an Authorization object:

```
https://mws-eu.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01
?AWSAccessKeyId=AKIAFBM3LG5JEEXAMPLE
&Action=GetAuthorizationDetails
&AmazonAuthorizationId=S23-1234567-1234567-0000001
&MWSAuthToken=amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE
&SellerId=YOUR_SELLER_ID_HERE
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2013-12-19T19%3A01%3A11Z
&Version=2013-01-01
&Signature=WlQ708aqyHXMkoUBk69Hjxj8qdh3aDcqpY71hVgEXAMPLE
```

For more information about the `GetAuthorizationDetails` operation, including the request parameters and response elements, see "GetAuthorizationDetails" in the Off-Amazon Payments API section reference.

Best practices for handling IPNs

Amazon sends you a notification when the state of any of the payment objects or the Order Reference object changes. These notifications are always sent automatically after they have been configured once during the integration phase and can be used to update any internal tracking or fulfillment systems you might be using to manage the order.

After you receive an IPN, a best practice is to always perform a get operation for the respective object for which you have received the notification to ensure the IPN message date and time are more current that the existing object's date and time.

Buyer-facing e-mail content that you can provide

Amazon sends an e-mail to buyers in the following situations:

- When an Order Reference object is confirmed.
- Any time that funds are captured.
- Any time that a refund is issued.
- Any time that a payment method is changed.

Amazon supplies most of the content for the e-mails, but there are several elements within the body of the message that you can provide. The following table lists the buyer-facing e-mail content that you can define based on account settings and API operation parameters:

E-mail Field	Source
Seller customer service e-mail address	Specified when you set up your Amazon seller account or from the Seller Central **Settings** tab under **Account Info**.
Seller customer service phone number	Specified when you set up your Amazon seller account or from the Seller Central **Settings** tab under **Account Info**.
Seller store name	Specified when you set up your Amazon seller account or from the Seller Central **Settings** tab under **Account Info**. Can be overridden in the `SetOrderReferenceDetails` operation.
Seller order identifier	Specified in the `SetOrderReferenceDetails` operation.
Total order amount	Specified in the `SetOrderReferenceDetails` operation.
Order date	The time stamp when the Order Reference object is confirmed and moves to the **Open** state.
Seller note	Specified in the `SetOrderReferenceDetails` operation, `Authorize` operation, `Capture` operation, or `Refund` operation.

Testing your integration with the Sandbox environment

The Advanced Payment APIs Sandbox environment enables you to thoroughly test your Pay with Amazon integration before going live with the Pay with Amazon payment option. When you are testing your implementation in Sandbox mode, you will be able to simulate the buyer experience as they navigate through the Amazon Payments widgets on your web site.

In Sandbox mode, you can also test your API operation calls to Amazon to ensure that the calls are configured correctly and that the responses include all the payment parameters that you need to track the entire order. The Sandbox environment even lets you simulate various error conditions to help you better manage your buyers' experiences in the event that something goes wrong during the checkout experience. For example, you can simulate a payment decline or a browser cookie timeout.

Test buyer accounts

When you first access the Sandbox environment through Seller Central, Amazon recommends configuring some test buyer accounts to help with your integration testing. These test accounts can be modified to suit most use cases, including the addition of shipping addresses that might violate a business rule your company has about where items can be shipped. You can configure these accounts from Seller Central by going to the **Integration** tab and selecting **Test Accounts**.

With the customization available through Seller Central, Amazon encourages you to test as many scenarios as you see fit through multiple test buyer accounts.

Shipping Address and Payment Instrument

There are sample shipping addresses and payment instruments that have been added in the Sandbox environment to enable the development and debugging of the **AddressBook** and **Wallet** widget.

When you create a test buyer account, the account will be pre-loaded with test payment instruments and a set of shipping addresses. You can edit the test buyer account properties, except for the payment instruments, to simulate various purchase scenarios. The shipping address and payment instrument are rendered in the AddressBook and Wallet widgets accordingly.

Billing Address

A single default billing address has been added to the Sandbox environment to enable testing of addresses for your locale. The sample billing address cannot be changed at this time.

Sandbox Simulations of Constraints

The **PaymentMethodNotAllowed** constraint can be simulated in the Sandbox environment so that you can test your code and ensure you handle this constraint properly. The following example shows how you can simulate the **PaymentMethodNotAllowed** constraint response to the `SetOrderReferenceDetails` operation.

In sandbox you need to do two things to simulate this constraint:

1. Set the **SellerNote** field of **OrderReferenceAttributes** with this simulation string: `{"SandboxSimulation": {"Constraint":"PaymentMethodNotAllowed"}}`
2. Select the direct debit method from the sandbox Wallet Widget.

In sandbox, switching to and from direct debit will enable or disable this constraint as long as the applicable simulation string is set. This is helpful for simulating real world scenarios, in the production environment, where the constraint may appear or disappear depending upon the payment method chosen by the buyer. To reduce the likelihood of this constraint triggering in production, we recommend you set the OrderTotal before you render the Wallet Widget.

Here is a fully encoded example:

```
https://mws-eu.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01
?AWSAccessKeyId=0GS7553JW74RRM612K02EXAMPLE
&Action=SetOrderReferenceDetails
&AmazonOrderReferenceId=S23-1234567-1234567
&MWSAuthToken=amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE
&OrderReferenceAttributes.OrderTotal.Amount=106
&OrderReferenceAttributes.OrderTotal.CurrencyCode=EUR
&OrderReferenceAttributes.SellerNote=%7B%22SandboxSimulation%22%3A%7B
%22Constraint%22%3A%22PaymentMethodNotAllowed%22%7D%7D
&OrderReferenceAttributes.SellerOrderAttributes.SellerOrderId=5678-23
&SellerId=YOUR_SELLER_ID_HERE
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2013-11-16T19%3A01%3A11Z
&Version=2013-01-01
&Signature=2RPzkOgQmDybUjk0dA54maCEXAMPLE
```

For more information about the SetOrderReferenceDetails operation, including the request parameters and response elements, see the Off-Amazon Payments API section .

Direct Debit

Currently Direct Debit is only available as a buyer payment method for those sellers registered with Amazon Payments in Germany.

Differences between the Sandbox and Production environments

The various objects used in Advanced Payment APIs are all managed through a set of business rules such as the amount of time until an object expires after being created. To help facilitate testing in the Advanced Payment APIs Sandbox, the following business rule has been adjusted:

Object	Production business rule	Sandbox business rule
An Authorization object in the **Open** state	Moves to the **Closed** state in 30 days	Moves to the **Closed** state in two days

Sandbox simulations

The Advanced Payment APIs Sandbox can be used to simulate specific payment scenarios. These simulations allow you to generate responses that can be used to test your own business processes. The following tables outline how certain responses and state transitions can be simulated. For more information about the states and reason codes, see the Off-Amazon Payments API section reference.

Note: For states and reason codes that can be used with a simulation string, you must specify the desired **State** and **ReasonCode**. Invalid combinations will result in an **InvalidSandboxSimulationSpecified** error.

To simulate various object states and reason codes:

Order Reference object			
State	**Reason Code**	**Simulation String**	**How to simulate in Sandbox**
Draft		N/A	Click the **Pay with Amazon** Sandbox button and sign in using your test buyer account credentials. This creates an Order

Order Reference object			
State	**Reason Code**	**Simulation String**	**How to simulate in Sandbox**
			Reference object in the **Draft** state.
Open		N/A	Confirm a **Draft** Order Reference object by using the `ConfirmOrderReference` operation.
Suspended	**InvalidPaymentMethod**	N/A	Simulate the Authorization object **Declined** state with reason code **InvalidPaymentMethod**. The Order Reference object then moves to the **Suspended** state.
Canceled	**SellerCanceled**	N/A	Cancel an **Open** or **Suspended** Order Reference object with the `CancelOrderReference` operation.
	Stale	N/A	Do not confirm a **Draft** Order Reference object within three hours of its creation. The Order Reference object then moves to the **Stale** state.
	AmazonCanceled	N/A	Cannot be simulated.
Closed	**Expired**	N/A	Do not close or cancel an Order Reference object. An **Open** or **Suspended** Order Reference object is closed by Amazon with this reason code after 180 days of its creation.
	MaxAmountCharged	N/A	Capture: • 15% or 75£, for the UK • 15% or 75€, for Germany whichever is less, above the amount originally specified in the open Order Reference object.
	MaxAuthorizationsCaptured	N/A	Fully or partially capture 10 authorizations against an **Open** Order Reference object.
	AmazonClosed	`{"SandboxSimulation": {"State":"Closed", "ReasonCode":"AmazonClosed"}}`	Specify this value in the **ClosureReason** request parameter of the `CloseOrderReference` operation for an **Open** Order Reference object.
	SellerClosed	N/A	Close an **Open** Order Reference object with the `CloseOrderReference` operation without specifying the simulation string for the **AmazonClosed** reason code.

Authorization object			
State	**Reason Code**	**Simulation String**	**How to simulate in Sandbox**
Pending		N/A	Request an authorization by calling the `Authorize` operation in asynchronous mode. All Authorization objects are in the **Pending** state for 30 seconds after you submit the `Authorize`

Authorization object			
State	**Reason Code**	**Simulation String**	**How to simulate in Sandbox**
			request. This cannot be simulated in synchronous mode.
Open		N/A	Request an authorization by calling the `Authorize` operation. In asynchronous mode, the Authorization object goes to the **Open** state after remaining in the **Pending** state for 30 seconds. In synchronous mode, the `Authorize` object immediately moves to the **Open** state.
Declined	**InvalidPaymentMethod**	`{"SandboxSimulation": {"State":"Declined", "ReasonCode":"InvalidPaymentMethod", "PaymentMethodUpdateTimeInMins":5}}`	Specify this value in the **SellerAuthorizationNote** request parameter of the `Authorize` operation. The order reference then moves from the **Open** state to the **Suspended** state. You can use the `PaymentMethod UpdateTimeInMins` parameter, to specify the time (between 1-240 minutes) after which the Order Reference object should move from the **Suspended** state back to the **Open** state. This simulates the buyer updating an invalid payment method and the Order Reference object moving back to the **Open** state.
	AmazonRejected	`{"SandboxSimulation": {"State":"Declined", "ReasonCode":"AmazonRejected"}}`	Specify this value in the **SellerAuthorizationNote** request parameter of the `Authorize` operation.
	ProcessingFailure	N/A	Cannot be simulated.
	TransactionTimedOut	`{"SandboxSimulation": {"State":"Declined", "ReasonCode":"TransactionTimedOut"}}`	Specify this value in the **SellerAuthorizationNote** request parameter of the `Authorize` operation.
Closed	**ExpiredUnused**	`{"SandboxSimulation": {"State":"Closed", "ReasonCode":"ExpiredUnused", "ExpirationTimeInMins":1}}`	Specify this value in the **SellerAuthorizationNote** request parameter of the `Authorize` operation. You can use the **ExpirationTimeInMins** parameter to specify the time after which the Authorization object should be closed (between 1 – 60 minutes).
	MaxCapturesProcessed	N/A	Capture an **Open** Authorization object by using the `Capture` operation.
	AmazonClosed	`{"SandboxSimulation": {"State":"Closed", "ReasonCode":"AmazonClosed"}}`	Specify this value in the **SellerAuthorizationNote** request parameter of the `Authorize` operation.
	OrderReferenceCanceled	N/A	Cancel the Order Reference object associated with the Authorization object by using the `CancelOrderReference` operation.

Authorization object			
State	**Reason Code**	**Simulation String**	**How to simulate in Sandbox**
	SellerClosed	N/A	Close an **Open** Authorization object by using the `CloseAuthorization` operation.

Capture object			
State	**Reason Code**	**Simulation String**	**How to simulate in Sandbox**
Pending		`{"SandboxSimulation": {"State":"Pending"}}`	Specify this value in the **SellerCaptureNote** request parameter of the `Capture` operation. The Capture object then remains in the **Pending** state for 30 seconds.
Declined	**AmazonRejected**	`{"SandboxSimulation": {"State":"Declined"}, {"ReasonCode":"AmazonRejected"}}`	Specify this value in the **SellerCaptureNote** request parameter of the `Capture` operation.
	ProcessingFailure	N/A	Cannot be simulated.
Completed		N/A	Capture funds against an **Open** Authorization object by calling the `Capture` operation.
Closed	**MaxAmountRefunded**	N/A	The maximum amount that you can refund is the lesser of 15% or • 75£ above the captured amount in the UK • 75€ above the captured amount in Germany No refunds are allowed against the capture after it is moved to the **Closed** state. The refund limit applies against the capture object, and not the sum of captures.
	MaxRefundsProcessed	N/A	Issue 10 refunds by calling the `Refund` operation against the corresponding **Completed** Capture object.
	AmazonClosed	`{"SandboxSimulation": {"State":"Closed"}, {"ReasonCode":"AmazonClosed"}}`	Specify this value in the **SellerCaptureNote** request parameter of the `Capture` operation.

Refund object			
State	**Reason Code**	**Simulation String**	**How to simulate in Sandbox**
Pending		N/A	Request a Refund by calling the `Refund` operation. All Refund objects will be in the **Pending** state for 30 seconds after you submit the `Refund` request.
Declined	**AmazonRejected**	`{"SandboxSimulation": {"State":"Declined", "ReasonCode":"AmazonRejected"}}`	Specify this value in the **SellerRefundNote** request parameter of the `Refund` operation.
	ProcessingFailure	N/A	Cannot be simulated.
Completed		N/A	Refund funds against a **Completed** Capture object by calling the `Refund` operation.

The following example demonstrates how you can simulate the Authorization **Declined** state with reason code **InvalidPaymentMethod**, using the **SellerAuthorizationNote** request parameter:

```
https://mws-eu.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01
?AWSAccessKeyId=AKIAFBM3LG5JEEXAMPLE
&Action=Authorize
&AmazonOrderReferenceId=S23-1234567-1234567
&AuthorizationAmount.Amount=50
&AuthorizationAmount.CurrencyCode=EUR
&AuthorizationReferenceId=test_authorize_1
&MWSAuthToken=amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE
&SellerAuthorizationNote=
    %7B%22SandboxSimulation%22%3A%7B%22State%22%3A%22Declined%22%2C
    %22ReasonCode%22%3A%22InvalidPaymentMethod%22%7D%7D
&SellerId=YOUR_SELLER_ID_HERE
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2013-10-03T19%3A01%3A11Z
&TransactionTimeout=60
&Version=2013-01-01
&Signature=WlQ708aqyHXMkoUBk69Hjxj8qdh3aDcqpY71hVgEXAMPLE
```

Handling errors

Both the Amazon Payments widgets and the Off-Amazon Payments API operations can throw errors. In order to provide the best possible buyer experience on your website, you should catch and handle both kinds of errors.

Handling errors from Off-Amazon Payments API section operations

When you get an error from the Off-Amazon Payments API section operations, you may be able to retry that operation depending on the nature of the error. Properly formed operation requests are idempotent—that is, if you call the same operation on an already successful request, you will not create a duplicate transaction. For order reference requests, the idempotency key is the **AmazonOrderReferenceId**. For authorize, capture, and refund requests, the idempotency key is the **AuthorizationReferenceId**, **CaptureReferenceId**, and **RefundReferenceId**, respectively. For example, you cannot refund against the same capture object more than once if you provide the same **RefundReferenceId**. This functionality allows you to safely retry any operation call if the error meets the conditions described in the table below.

If you want to retry an operation call after you receive an error, you can immediately retry after the first error response. If you want to retry multiple times, Amazon recommends that you implement an "exponential backoff" approach up to some limit. Then, log the error and proceed with a manual follow-up and investigation. For example, you can time your retries in the following time spacing: 1s, 2s, 4s, 10s, 30s. The actual backoff times and limit will depend upon your business processes.

The following table describes re-triable errors:

HTTP code	HTTP status	Description and corrective action
500	InternalServerError	The server encountered an unexpected condition which prevented it from fulfilling the request. This error is safe to retry.
502	Bad Gateway	A server between the client and the destination server was acting as a gateway or proxy and rejected the request. In many cases this is an intermittent issue and is safe to retry. If the issue persists please contact your network administrator.
503	ServiceUnavailable RequestThrottled	There was an unexpected problem on the server side or the server has throttled you for exceeding your transaction quota. For a complete explanation of throttling, see "Throttling: Limits to how often you can submit requests" in the Amazon MWS Developer Guide. The client should retry the request or reduce the frequency of requests.
504	Gateway Timeout	A server between the client and the destination server was acting as a gateway or proxy and timed out on the request. In many cases this is an intermittent issue and is safe to retry.

For more information, see "Error codes" in the Off-Amazon Payments API section reference.

Handling errors from Advanced Payment APIs widgets

As discussed in Step 1 - Add the Button widget for buyer authentication, you can configure the **Button**, **AddressBook**, and **Wallet** widgets to notify you of error conditions. These widgets will all send error notifications if certain integration errors are made. The following code example shows how you can read the **errorCode** and **errorMessage** associated with the error and display it in a text field. These error codes will help you debug your integration more quickly and if logged can notify you of potential issues on your production site.

```
<label>Debug Error Code     :</label>
<div id="errorCode"></div>
<br>
<label>Debug Error Message  :</label>
<div id="errorMessage"></div>

<script>
var amazonOrderReferenceId;
new OffAmazonPayments.Widgets.Button ({
sellerId: 'YOUR_SELLER_ID_HERE',
onSignIn: function(orderReference) {
amazonOrderReferenceId = orderReference.getAmazonOrderReferenceId();
window.location = 'https://yoursite.com/redirect_page?session='
+ amazonOrderReferenceId;
},
onError: function(error) {
   document.getElementById("errorCode").innerHTML = error.getErrorCode() ;
   document.getElementById("errorMessage").innerHTML =
 error.getErrorMessage() ;
}
}).bind("payWithAmazonDiv");
</script>
```

The following table lists the various error codes that are returned from the Advanced Payment APIs widgets and the corresponding error message that the buyer will see within the widget body:

Error Code	Description	Error Message Displayed to the Buyer
AddressBookWidgetNotApplicable	The Amazon **AddressBook** widget cannot be used because the `useAmazonAddressBook` parameter was set to *false* for this order reference.	We're sorry, but there's a problem processing your payment from this web site. Please contact the seller.
AddressNotModifiable	You cannot modify the shipping address when the order reference is in the given state.	You cannot change the shipping address for this order. Please contact the seller for assistance.
BuyerNotAssociated	The buyer is not associated with the given order reference. The buyer must sign in before you render the widget.	This session is not valid. Please re-start the checkout process by clicking the Pay with Amazon button.
InvalidAccountStatus	Your seller account is not in an appropriate state to execute this request. For example, it has been suspended or you have not completed registration.	If the error happened when displaying the **Button** widget, the widget will not appear. There is no error message displayed to the buyer. If the error happened when displaying the **AddressBook** or **Wallet** widgets, the buyer will see this message: We're sorry, but there's a problem processing your payment from this web site. Please contact seller for assistance.
InvalidOrderReferenceId	The specified order reference identifier is invalid.	We're sorry, but there's a problem processing your payment from this web site. Please contact the seller.
InvalidParameterValue	The value assigned to the specified parameter is not valid.	If the error happened when displaying the **Button** widget, the widget will not appear.

Error Code	Description	Error Message Displayed to the Buyer
		There is no error message displayed to the buyer. If the error happened when displaying the **AddressBook** or **Wallet** widgets, the buyer will see this message: We're sorry, but there's a problem processing your payment from this web site. Please contact seller for assistance.
InvalidSellerId	The seller identifier that you have provided is invalid. Specify a valid **SellerId**.	If the error happened when displaying the **Button** widget, the widget will not appear. There is no error message displayed to the buyer. If the error happened when displaying the **AddressBook** or **Wallet** widgets, the buyer will see this message: We're sorry, but there's a problem processing your payment from this web site. Please contact seller for assistance.
MissingParameter	The specified parameter is missing and must be provided.	If the error happened when displaying the **Button** widget, the widget will not appear. There is no error message displayed to the buyer. If the error happened when displaying the **AddressBook** or **Wallet** widgets, the buyer will see this message: We're sorry, but there's a problem processing your payment from this web site. Please contact seller for assistance.
PaymentMethodNotModifiable	You cannot modify the payment method when the order reference is in the given state.	You cannot change the payment method for this order. Please contact the seller for assistance.
ReleaseEnvironmentMismatch	You have attempted to render a widget in a release environment that does not match the release environment of the Order Reference object. The release environment of the widget and the Order Reference object must match.	We're sorry, but there's a problem processing your payment from this website. Please contact seller for assistance.
StaleOrderReference	The specified order reference was not confirmed in the allowed time and is now canceled. You cannot associate a payment method and an address with a canceled order reference.	Your session has expired. Please sign in again by clicking on the Pay with Amazon button.
UnknownError	There was an unknown error in the service.	We're sorry, but there's a problem processing your payment from this web site. Please contact the seller.

VAT Registered Sellers

If you need the buyer's address for sending VAT invoices or performing VAT calculations, it is important that your Amazon Seller registration contains your VAT information. In addition, setting VAT information in your Amazon Seller registration also enables monthly VAT reports for VAT-registered sellers.

VAT Information Setup

There are two ways to set your VAT information in your Amazon Seller registration:

1. During registration, you can provide your VAT information.
    1. Under the Business and Contact Information section select **I accept and acknowledge all the conditions set out in the VAT Agreement.** check box.
    2. Click the **VAT Agreement** link for more information.
2. Set your VAT information in Seller Central.
    - **a.** Logon to your Seller Central account.
    - **b.** Click Settings, in the upper corner of the screen, and then click Account Info.
    - **c.** Click Edit under the Business and Contact Information section.

      You can also enter additional VAT numbers.

VAT Reports

Setting your VAT information in your Amazon Seller registration also enables monthly VAT report functionality for your VAT-registered sellers.

Each VAT report lists the VAT invoice for seller fees paid for sales via Amazon Payments; the supplier and seller names, the VAT exclusive price, VAT percent, VAT charged, and the total for the sale.

Appendix

Sample Addresses and Payment Instruments

Sample UK and DE addresses and payment methods have been added in the Sandbox environment to enable development and debugging of AddressBook and Wallet widget information that is specific to these locales.

Sample default shipping addreses

When you create a test buyer account, you can associate default shipping addresses to that account at the time it is created. The available defaults are shown in the table below. The shipping address is rendered in the **AddressBook** widget.

Sample DE shipping addressed	Sample UK shipping addresses
Max Mustermann 123 Schutzstrasse 81543 Muenchen Deutschland +491731112222	Jane Doe 419 Kings Row Manchester M13 9PL Great Britain +447774443333
Moritz Harz 432 Baumstrasse 60596 Frankfurt Deutschland +491724444321	John Canterbury 111 Embassy Way Birmingham B15 TT Great Britain +447741112222
Wolfgang Kampf 82 Schoenstrasse 33100 Paderborn Deutschland +491625552222	Elisabeth Harrison 4973 Primrose Lane London SE1 2BY Great Britain +44774999888
Elise Schmidt 8675 Edelweiss Weg 79117 Freiburg Deutschland +491721111111	Charles Reed 1 Applewood Ave Spruce Tree Cottage Warwickshire CV32 4EL Great Britain +447741111222
Karin Roggen 1 Altstadtring	Abe Smith 23 Prime Way

Sample DE shipping addressed	Sample UK shipping addresses
21129 Hamburg Deutschland +491722223333	Churchill House London W2 4RJ Great Britain +447747474747

Sample default payment methods

Sample default payment methods are shown in the following table. The payment method is rendered in the **Wallet** widget.

Sample DE payment methods	Sample UK payment methods
InstrumentTypeName:"Visa", InstrumentNumber:”1111”, InstrumentType:”CC”	InstrumentTypeName:"Visa", InstrumentNumber:”1111”, InstrumentType:”CC”
InstrumentTypeName:"AMEX", InstrumentNumber:”0005”, InstrumentType:”CC”	InstrumentTypeName:"AMEX", InstrumentNumber:”0005”, InstrumentType:”CC”
InstrumentTypeName:"MasterCard", InstrumentNumber:”4444”, InstrumentType:”CC”	InstrumentTypeName:"MasterCard", InstrumentNumber:”4444”, InstrumentType:”CC”
InstrumentTypeName:"Bankeinzug", InstrumentNumber:”9424”, InstrumentType:”DD”	InstrumentTypeName:"MasterCard Debit (SoloMaestro)", InstrumentNumber:”9424”, InstrumentType:”DC”

Sample default billing address

A single fixed billing address has been added to the Sandbox environment to enable testing of addresses that may have unique features in these locales. These sample billing addresses cannot be changed at this time. The sample fixed billing addresses for Sandbox Simulation are as follows:

Sandbox fixed DE billing address	Sandbox fixed UK billing address
• Name arguments full_name :Liam Barker • Mailing address arguments address_line_1 : address_line_2 : Meininger Strasse 58 city : Neunkirchen postal_code : 66538 country_code : DE	• Name arguments --full_name : Amber Kelly • Mailing address arguments address_line_1 : 87 Terrick Rd city : Eilean Darach postal_code : IV23 2TW country_code : GB