Pay with Amazon
Express Integration Guide

Pay with Amazon
Express Integration Guide

Contents

Introduction

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

Express Integration offers either a Button Generator Integration to provide simple copy-and-paste HTML code you can use to add a **Pay with Amazon** button to your website, or a Custom Integration where you can generate the **Pay with Amazon** button on your website and make API calls.

This table shows the differences between these features.

	Button Generator Integration	**Custom Integration**
Copy-and-paste integration for fixed price items	✓	
Adjust the payment amount in a button request (as in a shopping cart)		✓
Use the Payment Management Dashboard to manage the capture and authorization of funds	✓	
Use APIs for authorization and capture of funds		✓
Retrieve the buyer's shipping address before capturing the final payment		✓

For more information see Button Generator Integration or Custom Integration.

Getting Started

To use the **Pay with Amazon** button, you need to complete the following tasks:

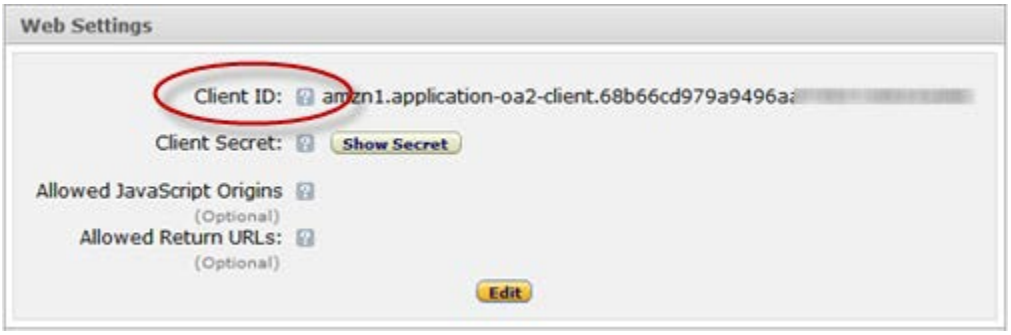
1. Register with Amazon Payments.
   If you've already registered for Amazon Payments you can proceed on to adding a button to your website.
2. Add a **Pay with Amazon** button to your web page with Button Generator Integration or Custom Integration.
3. Test your integration.
4. Collect and manage payments.

Register with Amazon Payments

You need an Amazon Payments account to generate a valid request. If you're using the Button Generator to perform an Express Integration, you can skip the steps marked **Optional**.

1. To register for Amazon Payments, go to https://payments.amazon.com/.

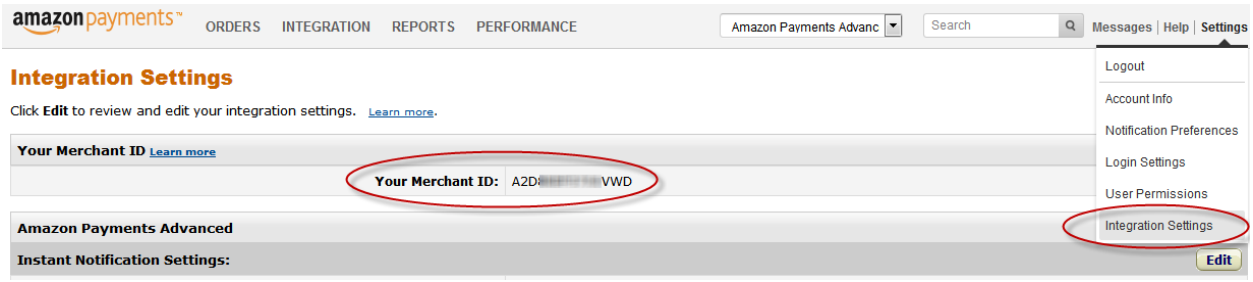
2. Create an application with Login with Amazon. Go to Seller Central to create the necessary Client ID:
   i. From the dropdown list near the top of the page, choose **Login with Amazon**.
   ii. Click **Register new application** and enter information about your application, then click **Save**.
   iii. Click **Web Settings**.
   iv. If you will be doing a custom integration, you need to note and save the Client ID.



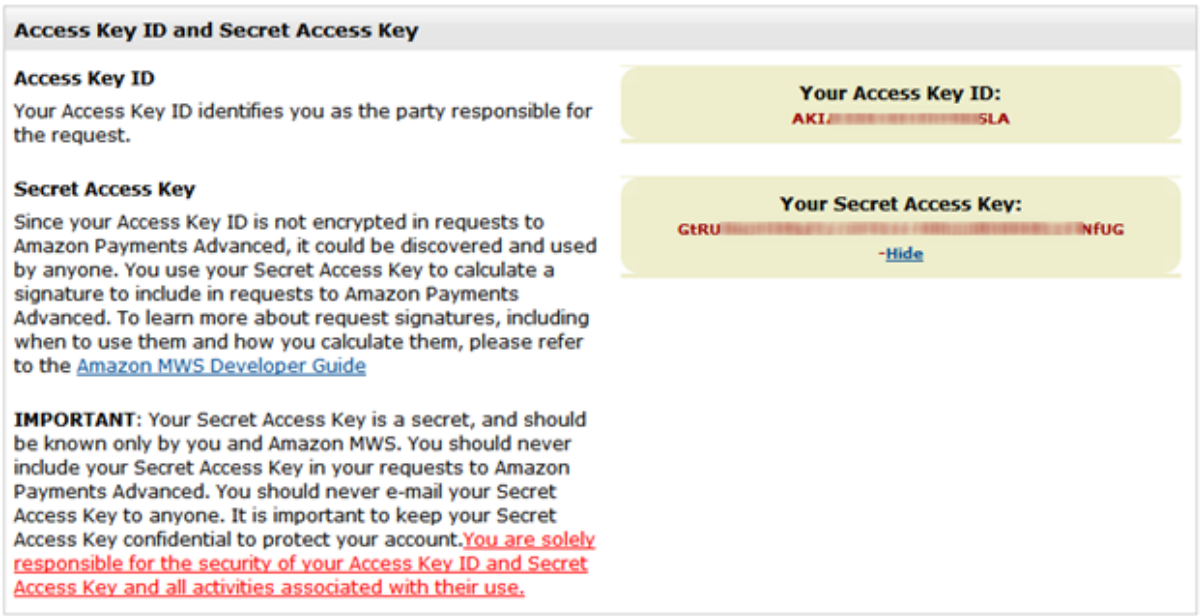
   v. Under **Web Settings**, click **Edit** and enter the **Allowed JavaScript Origin** and **Allowed Return URLs**. Specifying these values is how you authorize interaction between your web site and Amazon. (Web browsers normally block cross-origin communication between scripts unless the script specifically allows it.)
   An *origin* is the combination of protocol, domain name and port, for example, https://www.example.com:8443. Allowed origins must use the HTTPS protocol. If you are using a standard port (port 443) you need only include the domain name, for example, https://www.example.com.
   The *return URL* includes the protocol, domain, path, and query string(s), for

example, https://www.example.com/login.php.
For a detailed explanation of these options and on registering an application with Login with Amazon, see Step 1: Register Your Application in the Login with Amazon Getting Started for Web. (You can ignore the remaining steps in the Login with Amazon Getting Started for Web guide.)

3. **Optional:** For integrations that do not use the Button Generator, you need to note and save your Merchant ID. Your Merchant ID is used to sign your button or widget requests. To get your Merchant ID:
    i. In Seller Central, choose **Amazon Payments Advanced** from the dropdown list near the top of the page.
    ii. Click **Settings** and then click **Integration Settings.** The **Merchant ID** is used as the `sellerId` parameter in the request.



4. **Optional:** For integrations that do not use the Button Generator, you need to create your own signature to use to sign your requests. To create your own signature, you need to get your Amazon Marketplace Web Service (Amazon MWS) Access Key ID and Secret Access Key. Get your keys on the Amazon Payments console by first clicking **Integration** and then clicking **MWS Access Key**.



Button Generator Integration

Button Generator Integration for Login and **Pay with Amazon** offers a Button Generator to provide simple copy-and-paste HTML code you can use to add a full-featured **Pay with Amazon** button to your website.

To use the Button Generator Integration, complete the following tasks:

1. Use the Button Generator to create **Pay with Amazon** Sandbox buttons.
2. Test your **Pay with Amazon** buttons in Sandbox mode.
3. Use the Button Generator to create **Pay with Amazon** Production buttons.
4. Test your **Pay with Amazon** buttons in Production mode.

Step 1. Use the Button Generator to Create Pay with Amazon Sandbox Buttons

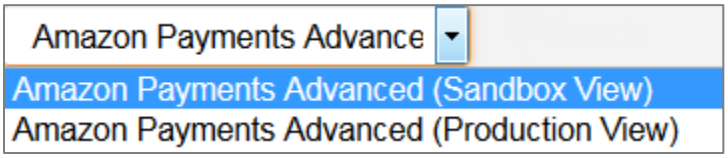
To test your integration and to ensure your buttons work correctly, Amazon Payments provides a Sandbox environment for testing. When you test your **Pay with Amazon** buttons in Sandbox mode, you can simulate the buyers' experience, making as many purchases as needed without incurring any charges.

You need to set up a test account in Sandbox. For instructions on how to do that see Appendix A - Set up an Amazon Payments Sandbox Test Account.

To use the Button Generator to create test buttons:

1. Logon to Seller Central.
2. Use the **Marketplace Switcher** dropdown box and select the **Amazon Payments Advanced (Sandbox View)**.

   The **Marketplace Switcher** appears as a dropdown box in the center of the menu located at the top of the screen:

   If your screen is minimized the Marketplace Switcher dropdown box is replaced with this icon:

   Note: The buttons that you generate in Sandbox mode will be specific to the sandbox test environment and can not be used in a production environment.

3. Go to the **Express Integration Button Generator** by clicking **Integration Central** under the **Integration** tab.
   The Button Generator is located in the **Express Integration Button Generator** section.

4. Click **Create**, enter the values you will be using, and then generate the HTML.

   An advantage to using the Button Generator is that it automatically generates the signature for your request. For parameter values that change, such as with a shopping cart, you would need to regenerate the signature for each request. See Custom Integration for more information.

   The following example shows code produced by the Button Generator.

```
<script async src="https://static-na.payments-
amazon.com/OffAmazonPayments/us/sandbox/js/Widgets.js"></script>
<div
  data-ap-widget-type="expressPaymentButton"
  data-ap-signature="gv5L4ElwMvMWUs1Q2huderKpOF6Fv4ulkL55Jh5B3YE%5D"
  data-ap-seller-id="SELLER_ID"
  data-ap-access-key="ACCESS_KEY"
  data-ap-lwa-client-id="amzn1.application-oa2-client.yourclientID"
  data-ap-return-url="https://www.example.com/order_complete.html"
  data-ap-currency-code="USD"
  data-ap-amount="30.00"
  data-ap-note="Thank you for your order"
  data-ap-shipping-address-required="true"
  data-ap-payment-action="AuthorizeAndCapture"
   >
</div>
```

   The parameters in the sample above are described in the topic Step3. Specify the parameter values under Custom Integration.

Step 2. Test Your Pay with Amazon Buttons in Sandbox Mode

To test your Sandbox buttons:

1. In Seller Central, verify you are still in the Sandbox View by making sure that the **Amazon Payments Advanced (Sandbox View)** option is selected in the **Marketplace Switcher** dropdown box.
2. Add the HTML code generated in step 1 to a web page.
3. Open your web page in a browser and click the **Pay with Amazon** button.
4. Sign on using your ***test*** email address and password you used when setting up your Sandbox Test Account, review the information, and then click **Pay now**.
5. In Seller Central, remaining in the **Sandbox View**, open the **Manage Payments** under the **Orders** tab.
6. Verify the order that you submitted is present and the order amount is correct.

Step 3. Use the Button Generator to Create Pay with Amazon Production Buttons

To generate production buttons:

1. Logon to Seller Central.
2. Use the Marketplace Switcher dropdown box and select the **Amazon Payments Advanced (Production View)**.
3. Select a test button you want to move to production.
4. Go to the **Express Integration Button Generator** by clicking **Integration Central** under the **Integration** tab.
   The Button Generator is located in the **Express Integration Button Generator**.
5. Enter the values you used from the test button, and then generate the HTML.

Step 4. Test your Pay with Amazon Buttons in Production Mode

Important! Orders submitted in a Production environment will be charged and you will be responsible for any fees incurred even if you are testing! You may elect to test only one or two buttons to ensure they are correct.

To test your Production buttons:

1. Add the HTML code generated in step 3 to your production web page.
2. In Seller Central, verify you are in the live production mode by making sure that the **Amazon Payments Advanced (Production View)** option is selected in the **Marketplace Switcher** dropdown box.
3. Open your production web page in a browser and click the **Pay with Amazon** button.
4. In Seller Central open the **Manage Payments Dashboard** and view the order that you submitted.
5. Verify that the order is correct.

Custom Integration

With the Custom Integration you can add your own button code. The button code allows the payment amount to be passed through a parameter and the transaction to be signed prior to making an API call.

To add a button to your web page and, complete the following tasks:

1. Add the Pay with Amazon JavaScript Reference.
2. Build and Create a **Pay with Amazon** Button.
3. Specify the Parameter Values.
4. Generate a Signature for the Payment Request.
5. Integrate Return URL Parameters.
6. Validate the Signature.
7. Retrieve the Shipping Address.
8. Test Your Integration in the Sandbox Environment Then Switch to Production.

Note: You can download code samples, and demos that show how to perform an Express Integration, in various languages (C#, Java, PHP, Ruby) at https://github.com/amzn/pay-with-amazon-express-demo/.

Step 1. Add the Pay with Amazon JavaScript Reference

To render the **Pay with Amazon** button, add the following reference at the end of the <body> section of your web page. Adding the snippet to the end of the <body> section helps ensure that the JavaScript loads completely. The version of the snippet you use depends on whether you're loading the web page into your sandbox environment or the production environment.

Note: A sandbox environment is a testing environment where you can test and modify your code before you push it to the live, production environment. As a best practice, you should always test your code before you release it to production. For information about using the Login and Pay with Amazon Payments Sandbox environment, go to Testing your integration with the Sandbox environment.

For instructions on how to set up a test account see Appendix A - Set up an Amazon Payments Sandbox Test Account.

For your Sandbox integration, use this snippet:

```
<script async type='text/javascript'
    src='https://static-na.payments-
amazon.com/OffAmazonPayments/us/sandbox/js/Widgets.js'>
</script>
```

For your live Production integration, use this snippet:

```
<script async type='text/javascript'
    src='https://static-na.payments-amazon.com/OffAmazonPayments/us/js/Widgets.js'>
</script>
```

Step 2. Build and Create a Pay with Amazon Button

If the payment amount used for your button is variable, such as when customers use a shopping cart to make purchases, then you need to use JavaScript to add the **Pay with Amazon** button with your web site. This topic includes an example of what your code might look like.

When the amount changes on the checkout page and cannot be preloaded, then the signature must be recalculated for each request. You should calculate the signature immediately before passing the payment parameters to `hostedParametersProvider`. The following example shows how you might do this, using a `JQuery.getJSON` call to your backend, and then hooking up a successful response with the `hostedParametersProvider` done function. The done function ends the Amazon button creation steps.

The following example shows making a getJSON call to the backend and hooking in the hostedParametersProvider done function for the Pay with Amazon button.

```
<div id="AmazonPayButton"/>
<script type="text/javascript">
   OffAmazonPayments.Button("AmazonPayButton", "SELLERID_HERE", {
     type: "hostedPayment",

     hostedParametersProvider: function(done){
        args = {
           amount: $("paymentAmount").val(),
           currencyCode: $("amountCurrencyCode").val()
           sellerNote: $("noteToBuyer").val()
        }
        // Call the back end to combine button args with
        // other seller config param values and sign it.
        $.getJSON("/generateRequestSignature",
           args,
           function(data) {
              done (data.params)
           })
        },
        onError: function(errorCode) {
            // your error handling code
        }
    });
</script>
```

The parameters in the sample above are described in the topic Step 3. Specify the parameter values.

Security warning: Do not expose your Amazon Marketplace Web Service (Amazon MWS) secret key in your web page's JavaScript or HTML code. Because your Amazon MWS secret key is used for authentication, it should be kept private in your backend.

Step 3. Specify the Parameter Values

After the **Pay with Amazon** button has been selected you must direct the buyer to the checkout page hosted by Amazon Payments. The URL should contain all data necessary to complete the flow and must be signed with your Amazon MWS secret key to ensure tampered data can be detected and handled appropriately.

Parameter **(Equivalent HTML interface Parameter)**	**Required**	**Need to sign**	**Description**

`sellerId`
 	
`(data-ap-seller-id)`
	Yes	Yes	This is your unique Seller ID. This is the same as your Merchant ID that is displayed in Seller Central. You can view your Merchant ID by going to **Settings**, then **Integration Settings**.
			
			Example: `ADEMO3053M41F7EXAMPLE`

`amount`
 	
`(data-ap-amount)`
	Yes	Yes	The amount of the payment.
			
			Example: `25.50`

`returnURL`
 	
`(data-ap-return-url)`
	Yes	Yes	The URL you want Amazon Payments to return responses to. This must include the scheme name (either HTTP or HTTPS).
			
			Example: `https://www.merchant.com/merchantResponseHandler`

`accessKey`
 	
`(data-ap-access-key)`
	Yes	Yes	Your Amazon MWS public access key.
			
			Example: `ADEMOBRU3PYWWEXAMPLE`

`Signature`
 	
`(data-ap-signature)`
	Yes	No	The signature used to sign your request. The signature ensures that the button parameters have not been tampered with. For more information about how to generate the signature, see Step 4. Generate a signature for the payment request.

`lwaClientId` `(data-ap-lwa-client-id)`	Yes	Yes	The Login with Amazon Client ID of your application. Example: `amzn1.application-oa2-client.demoa234d9024af28f4f6f8078example`
`currencyCode` `(data-ap-currency-code)`	No	Yes *	The currency to use to charge the buyer. Default: current seller region Example: USD
`sellerNote` `(data-ap-note)`	No	Yes *	The message that will appear in the checkout pages. Max length: 1024 characters
`sellerOrderId` `(data-ap-seller-order-id)`	No	Yes *	The seller-specified identifier of this order. This is displayed in buyer emails and in the transaction history on the Amazon Payments website. We recommend that you use the following characters only: lowercase a-z, uppercase A-Z, numbers 0-9, dash (-), or underscore (_). Max length: 50 characters
`shippingAddressRequired` `(data-ap-shipping-address-required)`	No	Yes *	A flag indicating whether the buyer should select shipping address. Default: `true` Valid values: `true` | `false`
`paymentAction` `(data-ap-payment-action)`	No	Yes *	Specifies what happens when someone clicks Pay Now at the end of the checkout flow. Acceptable values: `None | Authorize | AuthorizeAndCapture` ▪ `None`: results in Set and Confirm actions only. (You need to initiate authorize and capture actions separately using API requests.) ▪ `Authorize`: reserves a specified amount against the payment method(s) stored in the order

			reference. ▪ `AuthorizeAndCapture`: reserves a specified amount against the payment method(s) stored in the order reference and transfers funds from an authorized payment instrument. Default: None

*These parameters are required when signing only if you designated a value other than the default.

The following parameters are available in the HTML interface only. These parameters do not require signing.

HTML parameter	Required	Need to sign	Description
`data-ap-widget-type`	Yes	No	The type of widget to include on the page. Valid value: expressPaymentButton

Step 4. Generate a Signature for the Payment Request

If you're using the Pay with Amazon API, you must include a signature in the request parameters so that Amazon can authenticate your payment requests. If you don't include a signature, or if the signature is incorrect, the buyer will be directed to an error page that will redirect to your specified `returnURL` for failure handling.

To generate the signature, complete the following tasks:

1. Construct the string to sign.
2. Sign the string with your Amazon MWS secret access key.
3. Add the signature to the **Pay with Amazon** button parameters.

To create a valid signature, you need to construct the string to sign according to the Amazon MWS V2 signature spec. The string consists of the following elements, with each section separated by a new line:

- The HTTP action. For a Pay with Amazon request it is always POST.
- The request domain. For a Pay with Amazon request it is always a forward slash (/).
- Sorted parameters in query string format, with the URL encoded parameter name and value.

Note: You should include **only the parameters that must be signed** when generating a signature.

The required parameters are:

- `sellerId`
- `amount`
- `returnURL`
- `accessKey`
- `lwaClientId`

You should include these parameters when generating a signature only if you designated a value *other than the default*:

- `currencyCode`
- `sellerNote`
- `sellerOrderId`
- `shippingAddressRequired`
- `paymentAction`

The following example shows what your string to sign might look like. Replace the value of each field according to the parameter values you want to use. (Note that the example includes line breaks so that it is easier to read; a real response would be returned as a single, continuous string.)

```
POST
payments.amazon.com
/
accessKey=ACCESSKEY&amount=1.01
&currencyCode=USD
&lwaClientId=LWACLIENTID
&paymentAction=None
&returnURL=https%3A%2F%2FreturnURL
&sellerId=SELLERID
&sellerNote=SELLERNOTE
&sellerOrderId=SELLERORDERID
&shippingAddressRequired=true
```

Security recommendation: You should use HTTPS when generating the signature. This will help prevent third parties from eavesdropping sensitive payment information.

Step 5. Integrate Return URL Parameters

Whether the request was successful or failed, Login and Pay with Amazon returns an URL containing parameters you can integrate into your order management system. For example, if the request was successful, at the conclusion of the checkout flow, the buyer will be redirected back to the URL you specified in the button request along with parameters describing the successful transaction.

Similarly, if a buyer abandons the checkout process by clicking Cancel on one of the checkout pages, they will be redirected to your website where return URL parameters can provide the reason for abandonment. This topic describes Login and Pay with Amazon return URL parameters.

Important: Before fulfilling orders on a successful response, you should verify the `signature`, `amount`, `sellerOrderId`, and `currencyCode` parameters to ensure that the response was sent by Login and Pay with Amazon and that it has been properly processed. Alternatively, you can make backend calls via the Amazon MWS Off-Amazon Payments API using the `OrderReferenceId`.

Note: Integrating return URL parameters is one way to monitor the status of transactions on your web page, but we recommend you also use Instant Payment Notification (IPN) to monitor transactions. IPN is a HTML POST notification that is sent when a transaction either completes successfully or fails. You can specify the default URL to handle IPN in your Amazon Payments account settings. For more information on IPN, go to Synchronizing your systems with Amazon Payments: Monitoring payment object state transitions in the Login and Pay with Amazon Integration Guide. To view transactions in Seller Central, sign in and go to **Orders**, and then select **Manage Transactions**.

Common parameters

Parameter	Required	Description
`resultCode`	Yes	The response indicating whether checkout was successful. Valid values: `Success` | `Failure`
`sellerId`	Yes	The Seller ID used for the request. If you are a solution provider would use this to identify the merchant you made the call for.

Success parameters

Parameter	Required	Description
`orderReferenceId`	Yes	The ID of the order reference. The order reference is the contract that encapsulates the payment agreement between you and the buyer.
`amount`	Yes	The amount that the buyer has agreed to pay.
`currencyCode`	Yes	The currency to charge the buyer in.
`paymentAction`	Yes	The specified request parameter. Valid values: `None` |`Authorize` | `AuthorizeAndCapture`
`sellerOrderId`	No	The seller-specified ID for this order. This is returned only if it is provided as a button parameter.
`accessKey`	Yes	The Amazon MWS private key's corresponding public key used to create the response signature.
`signature`	Yes	A signature used to ensure that the Amazon Payment response parameters have not been tampered with. If you detect a signature mismatch then ignore the response. For information about how to calculate the signature, see Step 4. Generate a signature for the payment request.

Failure parameters

Parameter	Required	Description
`failureCode`	Yes	The code describing the error. Examples: ▪ `BuyerAbandoned` ▪ `AmazonRejected` ▪ `RequestSignatureFailure` ▪ `InvalidParameterValue` ▪ `MissingParameter` ▪ `InvalidSellerAccountStatus` ▪ `TemporarySystemIssue`

The following example shows what the response might look like if the transaction is successful. (Note that the example includes line breaks so that it is easier to read; a real response would be returned as a single, continuous string.)

```
http://www.courtandcherry.com/AmazonResponseHandler
?resultCode=Success
&orderReferenceId=S01-0912345-1234567
&sellerId=ADEMO3053M41F7TEXAMPLE
&accessKey=ARIWE420982EXAMPLE
&amount=10
&currencyCode=USD
&paymentAction=None
&sellerOrderId=A481WEIFEXAMPLE
&signature=n345ngiwasdfasdJimCJixkEuxqN0021sdf56bDdZ4EXAMPLE
```

The following example shows what a response might look like if the buyer cancels the transaction.

```
http://www.courtandcherry.com/AmazonResponseHandler
?resultCode=Failure
&sellerId=ADEMO3053M41F7EXAMPLE
&failureCode=BuyerAbandoned
```

Step 6. Validate the Signature (optional)

After an order is placed, you should validate the signature in the return URL to ensure that it came from Amazon.

To validate the returned signature, complete the following tasks:

1. Construct the string to sign.
2. Sign the string with your Amazon MWS secret access key.
3. Generate the signature on your server and compare with the returned signature.

To create a valid signature, you need to construct the string to sign according to the Amazon MWS V2 signature spec. The string consists of the following elements, with each section separated by a new line:

- The HTTP action. For generating the signature on your server the request is always GET.
- The domain name of the request URL for your server's return URL file.
- The request domain which is the qualified pathway to your server's return URL file.
- Sorted parameters in query string format, with the URL encoded parameter name and value. Include only the parameters that were in the return URL from Amazon.

The following example shows what your string to sign might look like. Note that if your server's return URL is http://your-domain.com/path/success-file.html the domain URL and request domain would be as shown in the following example.

```
GET
your-domain.com
/path/validatesignature.html
AWSAccessKeyId=AKIAEXAMPLE&SignatureMethod=HmacSHA256&SignatureVersion=
2&amount=99.00&currencyCode=USD&orderReferenceId=S01-999999-
9999999&paymentAction=AuthorizeAndCapture&resultCode=Success&sellerId=A
2MQTZXEXAMPLE&sellerOrderId=1234
```

Step 7. Retrieve the Shipping Address

If your business ships physical goods you can retrieve the buyer's shipping address using the **Manage Transactions** tool, under **Orders**, in Seller Central.

You may also retrieve the shipping address by calling the `GetOrderReferenceDetails` operation. Express Integration handles details such as `SetOrderReferenceDetails` and `ConfirmOrderReferenceDetails` automatically so you can call `GetOrderRefernceDetails` directly using the `orderReferenceId` appended to the return URL you have specified.

The `GetOrderReferenceDetails` operation can be easily tested using the Amazon MWS Scratchpad tool. Open Amazon MWS Scratchpad and make the following selections under the **API Selection**:

- **API Section:** select **Off-Amazon Payments**.
- **Operation:** from under ***OffAmazonPayments-Sandbox*** select **GetOrderReference Details.**

Enter your credentials the **Authentication** parameters and **Submit** the request.

The following example shows a Scratchpad request:

```
POST /OffAmazonPayments/2013-01-01 HTTP/1.1
Content-Type: x-www-form-urlencoded
Host: mws.amazonservices.com
User-Agent: <Your User Agent Header>

AWSAccessKeyId=AKIAJKYFSJU7PEXAMPLE
&Action=GetOrderReferenceDetails
&AddressConsentToken=IQEBLzAtAhUAjagYW4Jrgw84pCaaIDjrKoEhZXsEXAMPLE
&AmazonOrderReferenceId=P01-1234-56EXAMPLE
&MWSAuthToken=amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE
&SellerId=YOUR_SELLER_ID_HERE
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2012-11-05T19%3A01%3A11Z
&Version=2013-01-01
&Signature=CLZOdtJGjAo81IxaLoE7af6HqK0EXAMPLE
```

For more information see GetOrderReferenceDetails in the Off-Amazon Payments API Reference Guide.

The following example shows a Scratchpad response to a `GetOrderReferenceDetails` operation call.

```
<GetOrderReferenceDetailsResponse
   xmlns="http://mws.amazonservices.com/schema/OffAmazonPayments/2013-01-01">
  <GetOrderReferenceDetailsResult>
    <OrderReferenceDetails>
    <AmazonOrderReferenceId>S01-8474066-EXAMPLE</AmazonOrderReferenceId>
    <ExpirationTimestamp>2015-08-31T17:27:35.053Z</ExpirationTimestamp>
    <SellerNote>Thank you for your order.</SellerNote>
    <OrderTotal>
        <Amount>15.00</Amount>
        <CurrencyCode>USD</CurrencyCode>
    </OrderTotal>
    <IdList>
        <member>S01-8474066-2115738-EXAMPLE</member>
    </IdList>
    <OrderReferenceStatus>
        <LastUpdateTimestamp>
          2015-03-04T17:27:43.851Z
        </LastUpdateTimestamp>
        <State>Open</State>
    </OrderReferenceStatus>
    <Destination>
        <DestinationType>Physical</DestinationType>
        <PhysicalDestination>
          <Phone>800-000-0000</Phone>
          <PostalCode>60602</PostalCode>
          <Name>Susie Smith</Name>
          <CountryCode>US</CountryCode>
          <StateOrRegion>IL</StateOrRegion>
          <AddressLine2>Suite 2500</AddressLine2>
          <AddressLine1>10 Ditka Ave</AddressLine1>
          <City>Chicago</City>
        </PhysicalDestination>
    </Destination>
    <ReleaseEnvironment>Sandbox</ReleaseEnvironment>
    <Buyer>
        <Email>address@domain.com</Email>
        <Name>Sandbox_tester</Name>
    </Buyer>
    <SellerOrderAttributes>
        <StoreName>Sandbox_test_app</StoreName>
    </SellerOrderAttributes>
    <CreationTimestamp>2015-03-04T17:27:35.053Z</CreationTimestamp>
    </OrderReferenceDetails>
  </GetOrderReferenceDetailsResult>
  <ResponseMetadata>
        <RequestId>181efc0d-a90b-429f-837e-1209bfe4da0e</RequestId>
  </ResponseMetadata>
</GetOrderReferenceDetailsResponse>
```

Step 8. Test Your Integration in the Sandbox Environment Then Switch to Production

To test your integration and to ensure you have covered all possible edge cases, Amazon Payments provides a Sandbox environment for testing. When you test your implementation in Sandbox mode, you can simulate the buyers' experience as they navigate through the **Pay with Amazon** button on your website.

In Sandbox mode, you can also test your API operation calls to Amazon Payments 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 also lets you simulate various error conditions to help you better manage your buyers' experiences in the event that something goes wrong during the checkout process. For example, you can simulate a payment decline or a browser cookie timeout. For information about using the Login and Pay with Amazon Payments Sandbox environment, go to Testing your integration with the Sandbox environment.

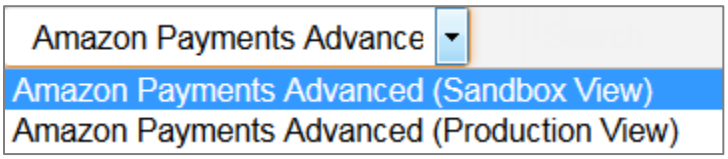
After you have thoroughly tested your integration, you can replace the sandbox endpoints in the JavaScript snippet with the production endpoints and push your integration to your production site. (The production endpoint is shown in the example in Add the JavaScript snippet that renders the button.)

Appendix A - Set up an Amazon Payments Sandbox Test Account

You need to setup a test account in the Amazon Payments Sandbox so that you can test you integration.

1. Logon to Seller Central at https://sellercentral.amazon.com, if you haven't already done so.
2. Select **Amazon Payments Advanced (Sandbox)** from the **Marketplace Switcher** dropdown box found in the center of the menu bar.

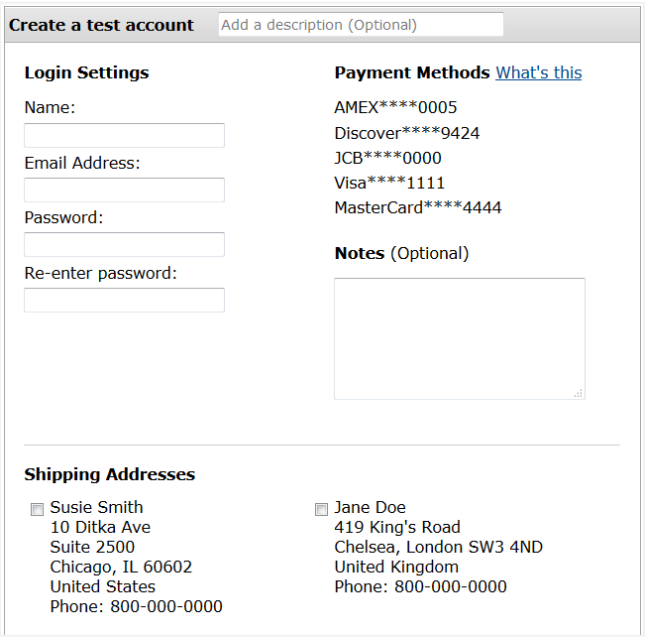
   The **Marketplace Switcher** appears as a dropdown box in the center of the menu located at the top of the screen:

   If your screen is minimized the Marketplace Switcher dropdown box is replaced with this icon:

   
3. Click **Test Accounts** under the **Integration Tab**.
4. Follow the on screen instructions:

- Login Settings - This login is for testing in the Sandbox. It requires a different email address and password than what you use for your production account.
  Email Address - You can use your production email but add "+sandbox" before the "@" sign. For example, if your email address is JayDoe@outlook.com your sandbox email would be JayDoe+sandbox@outlook.com.
  Password - This is a different password than your production password, such as "testSandbox".
- Payments - The credit card numbers under the Payments Methods are preset fictitious charge cards.
- Shipping Addresses - select, or add, a shipping address.



5. Click **Create account** to save your settings.
   Your test account is created.