
Login with Amazon

Getting Started Guide for Websites

Version 1.0

Login with Amazon: Getting Started Guide for Websites

Copyright © 2013 Amazon Services, LLC or its affiliates. All rights reserved.

Amazon and the Amazon logo are trademarks of Amazon.com, Inc. or its affiliates. All other trademarks not owned by Amazon are the property of their respective owners.

Contents

Getting Started for Web.....	1
Register Your Application.....	2
Add the Login with Amazon Button.....	4
Add the SDK for JavaScript.....	5
Obtain Profile Information.....	6
Finish Integration with Your Website.....	9
Glossary.....	10

Getting Started for Web

Topics

- [Register Your Application](#)
- [Add the Login with Amazon Button](#)
- [Add the SDK for JavaScript](#)
- [Obtain Profile Information](#)
- [Finish Integration with Your Website](#)

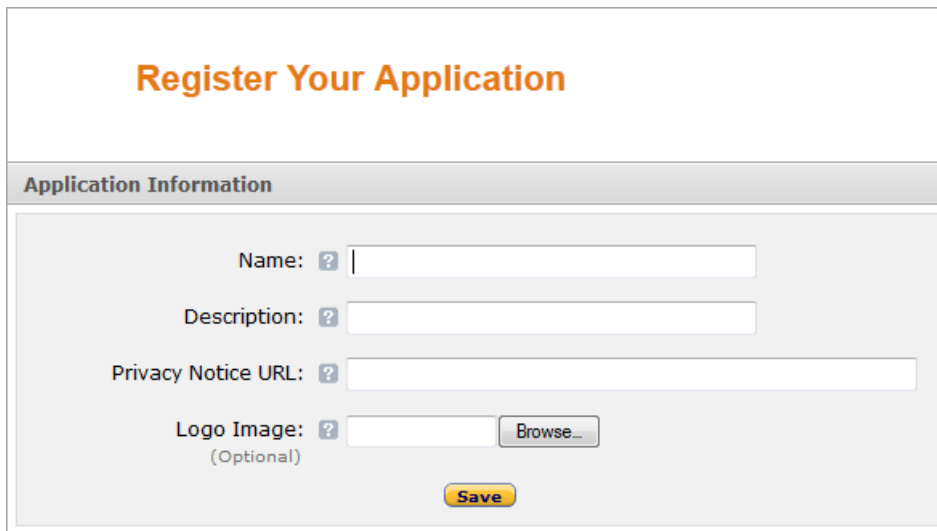
In this guide we will show you how to add Login with Amazon to your website or web service. After completing this guide you should have a working Login with Amazon button on your website to allow users to log in with their Amazon credentials.

Register Your Application

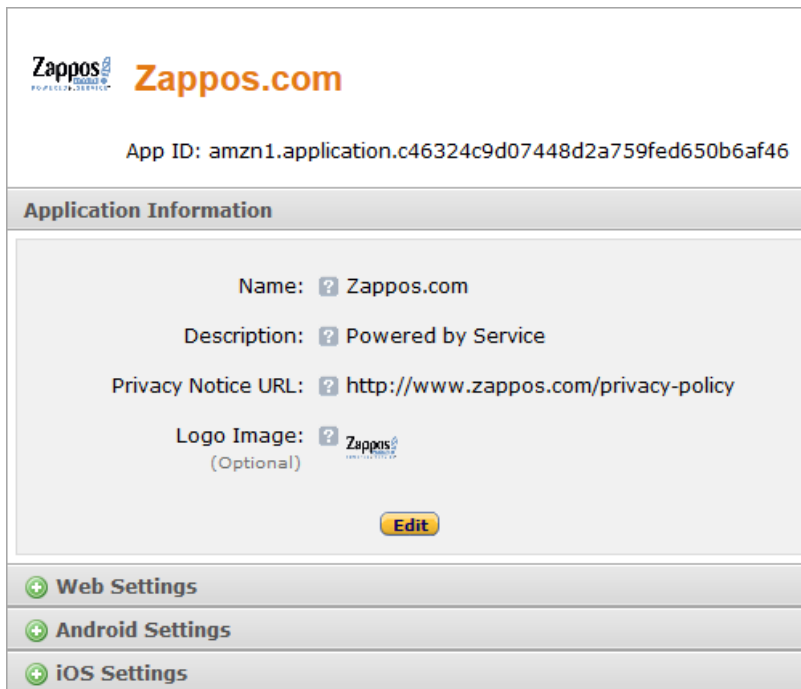
First, you will need to register your website as an [Application](#) on the [App Console](#) at login.amazon.com. Then you can add settings for specific websites and mobile apps.

Register Your Login with Amazon Application

1. Visit <https://login.amazon.com>.
2. If you have signed up for Login with Amazon before, click **App Console**. Otherwise, click **Sign Up**. You will be redirected to Seller Central, which handles application registration for Login with Amazon. If this is your first time using Seller Central, you will be asked to set up a Seller Central account.
3. Click **Register new application**. The **Register Your Application** form will appear:



- a. In the **Register Your Application** form, you must enter a **Name** and a **Description** for your application.
The **Name** is the name displayed on the consent screen when users agree to share information with your application. This name applies to Android, iOS, and website versions of your application.
 - b. You must enter a **Privacy URL** for your application now.
The Privacy Notice URL is the location of your company or application's privacy policy (for example, <http://www.example.com/privacy.html>). This link is displayed to users on the consent screen.
 - c. If you want to add a **Logo Image** for your application, click **Browse**.
This logo is displayed on the sign-in and consent screen to represent your business or website. The logo will be shrunk to 50 pixels in height if it is taller than 50 pixels; there is no limitation on the width of the logo.
4. Click **Save**. Your sample registration should look similar to this:



The screenshot shows the 'Application Information' section for an application named 'Zappos.com'. The App ID is 'amzn1.application.c46324c9d07448d2a759fed650b6af46'. Below the application information, there are three expandable sections: 'Web Settings', 'Android Settings', and 'iOS Settings', each with a green plus icon.

Zappos.com


App ID: amzn1.application.c46324c9d07448d2a759fed650b6af46

Application Information

Name: ? Zappos.com

Description: ? Powered by Service

Privacy Notice URL: ? http://www.zappos.com/privacy-policy

Logo Image: ?  (Optional)

[Edit](#)

[+ Web Settings](#)

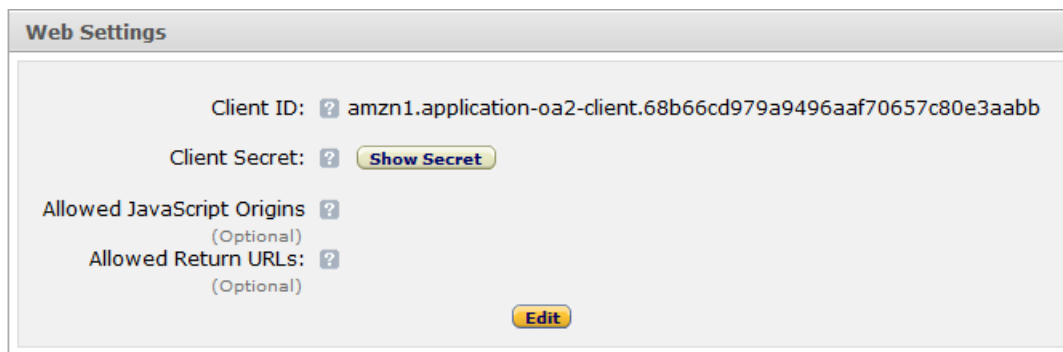
[+ Android Settings](#)

[+ iOS Settings](#)

After your basic application settings are saved, you can add settings for specific websites and mobile apps that will use this Login with Amazon account.

Adding a Website to Your Application

1. From the Application screen, click **Web Settings**. You will automatically be assigned values for **Client ID** and **Client Secret**. The client ID identifies your website, and the client secret is used in some circumstances to verify your website is authentic. The client secret, like a password, is confidential. To view the client secret, click **Show Secret**.



The screenshot shows the 'Web Settings' section. It displays the Client ID as 'amzn1.application-oa2-client.68b66cd979a9496aaf70657c80e3aabb'. The Client Secret is shown with a 'Show Secret' button. Below, there are fields for 'Allowed JavaScript Origins' and 'Allowed Return URLs', both marked as optional. An 'Edit' button is at the bottom.

Web Settings

Client ID: ? amzn1.application-oa2-client.68b66cd979a9496aaf70657c80e3aabb

Client Secret: ? [Show Secret](#)

Allowed JavaScript Origins ?
(Optional)

Allowed Return URLs ?
(Optional)

[Edit](#)

2. To add **Allowed Javascript Origins** or **Allowed Return URLs** to your application, click **Edit**.

Note

To use Login with Amazon with a website, you must specify either an allowed JavaScript origin (for the Implicit grant) or an allowed return URL (for the Authorization Code grant). If you are using Pay with Amazon, you must specify an allowed JavaScript origin.

Login with Amazon: Getting Started Guide for Websites

Add the Login with Amazon Button

Web Settings

Client ID: ? amzn1.application-oa2-client.68b66cd979a9496aaf70657c80e3aabb

Client Secret: ? [Show Secret](#)

Allowed JavaScript Origins ?
(Optional) [Add Another](#)

Allowed Return URLs ?
(Optional) [Add Another](#)

[Cancel](#) [Save](#)

- a. If your website will use the Login with Amazon SDK for JavaScript, add your website origin to **Allowed JavaScript Origins**.

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 80 or port 443) you need only include the domain name (for example, `https://www.example.com`).

Adding your domain here allows the SDK for JavaScript to communicate with your website directly during the login process. Web browsers normally block cross-origin communication between scripts unless the script specifically allows it.

To add more than one origin, click **Add another**.

- b. If your website will be making HTTPS calls to the Login with Amazon authorization service and specifying a `redirect_uri` for replies, add those redirect URIs to Allowed Return URLs.

The return URL includes the protocol, domain, path, and query string(s) (for example, `https://www.example.com/login.php`).

To add more than one return URL, click **Add another**.

3. Click **Save**.

Add the Login with Amazon Button

Next, add the Login with Amazon button to your website. You can pick from a variety of buttons and choose the image that best fits your website. See the [Login with Amazon Style Guidelines](#) for best practices and a list of images to choose from.

1. Add the following code to your website where you would like the button to appear. For the purposes of this guide, this must be an HTTPS website:

```
<a href id="LoginWithAmazon">
  
</a>
```

2. **Optional.** Add the following link to your website where you would like a "Logout" prompt to appear:

```
<a id="Logout">Logout</a>
```

3. Refresh the page to confirm that the button now appears on your website.

Add the SDK for JavaScript

The Login with Amazon SDK for JavaScript will handle all of the difficult parts of integrating Login with Amazon into your website.

1. Add the following code after the opening `<body>` in your page to load the JavaScript into your page:

```
<div id="amazon-root"></div>
<script type="text/javascript">
  window.onAmazonLoginReady = function() {
    amazon.Login.setClientId('YOUR-CLIENT-ID');
  };
  (function(d) {
    var a = d.createElement('script'); a.type = 'text/javascript';
    a.async = true; a.id = 'amazon-login-sdk';
    a.src = 'https://api-cdn.amazon.com/sdk/login1.js';
    d.getElementById('amazon-root').appendChild(a);
  })(document);
</script>
```

2. Replace YOUR-CLIENT-ID with your Client ID from the *Register Your Application* section above.
3. Add the following JavaScript after the Login with Amazon button on your site.

```
<script type="text/javascript">

  document.getElementById('LoginWithAmazon').onclick = function() {
    options = { scope : 'profile' };
    amazon.Login.authorize(options,
      'https://www.example.com/handle_login.php');
    return false;
  };

</script>
```

4. Replace `www.example.com` with the domain of your website.

Note

Once the user has logged in and consented to share the specified data, the current window will be redirected to the given URI and the authorization response will be added to the query string. The URI must use the https protocol and be on the same domain as the current window.

5. **Optional.** After users are authorized, you should add access to a Logout hyperlink or button on your site so they can logout.

Add the following JavaScript to enable users to logout:

```
<script type="text/javascript">
  document.getElementById('Logout').onclick = function() {
    amazon.Login.logout();
  };
</script>
```

You will be handling the response from Amazon with `/handle_login.php` on your website in the next section. You can change this path to one of your choosing at a later time.

Obtain Profile Information

Next, obtain the user's profile information from Amazon using the [Access Token](#) returned by the SDK.

1. In your server-side application, handle the request made to `/handle_login.php`, and obtain profile information using the access token and the Profile REST API.

Example PHP

```
// Verify that the access token belongs to us
// The token must be url-encoded when passed to tokeninfo
$c = curl_init('https://api.amazon.com/auth/o2/tokeninfo?access_token='
    . urlencode($_REQUEST['access_token']));
curl_setopt($c, CURLOPT_RETURNTRANSFER, true);

$r = curl_exec($c);
curl_close($c);
$d = json_decode($r);

if ($d->aud != 'YOUR-CLIENT-ID') {
    // the access token does not belong to us
    header('HTTP/1.1 404 Not Found');
    echo 'Page not found';
    exit;
}

// Exchange the access token for user profile
// The token must NOT be url-encoded when passed to profile
$c = curl_init('https://api.amazon.com/user/profile');
curl_setopt($c, CURLOPT_HTTPHEADER, array('Authorization: bearer '
    . $_REQUEST['access_token']));
curl_setopt($c, CURLOPT_RETURNTRANSFER, true);

$r = curl_exec($c);
curl_close($c);
$d = json_decode($r);

echo sprintf('%s %s %s', $d->name, $d->email, $d->user_id);
```

Example Python

You must download the [pycurl](#) library to use this sample code.

```
import pycurl
import urllib
import json
import StringIO

...

b = StringIO.StringIO()

# Verify that the access token belongs to us
# The token must be url-encoded when passed to tokeninfo
c = pycurl.Curl()
c.setopt(pycurl.URL,
         "https://api.amazon.com/auth/o2/tokeninfo?access_token="
         + urllib.quote_plus(access_token))
c.setopt(pycurl.SSL_VERIFYPEER, 1)
c.setopt(pycurl.WRITEFUNCTION, b.write)

c.perform()
d = json.loads(b.getvalue())

if d['aud'] != 'YOUR-CLIENT-ID' :
    # the access token does not belong to us
    raise BaseException("Invalid Token")

# Exchange the access token for user profile
# The token must NOT be url-encoded when passed to profile
b = StringIO.StringIO()

c = pycurl.Curl()
c.setopt(pycurl.URL, "https://api.amazon.com/user/profile")
c.setopt(pycurl.HTTPHEADER, ["Authorization: bearer " + access_token])
c.setopt(pycurl.SSL_VERIFYPEER, 1)
c.setopt(pycurl.WRITEFUNCTION, b.write)

c.perform()
d = json.loads(b.getvalue())

print "%s %s %s"%(d['name'], d['email'], d['user_id'])
```

Example Java

You must download the [Jackson](#) and [HttpComponents](#) libraries to use this sample code.

```
import com.fasterxml.jackson.core.type.TypeReference;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.apache.http.client.fluent.Content;
import org.apache.http.client.fluent.Request;

import java.net.URLEncoder;
import java.util.Map;

...

// Verify that the access token belongs to us
// The token must be url-encoded when passed to tokeninfo
Content c =
    Request.Get("https://api.amazon.com/auth/o2/tokeninfo?access_token="
        + URLEncoder.encode(access_token, "UTF-8"))
        .execute()
        .returnContent();

Map m = new ObjectMapper().readValue(c.toString(), new TypeReference<>(){});

if (!"YOUR-CLIENT-ID".equals(m.get("aud"))) {
    // the access token does not belong to us
    throw new RuntimeException("Invalid token");
}

// Exchange the access token for user profile
// The token must NOT be url-encoded when passed to profile
c = Request.Get("https://api.amazon.com/user/profile")
    .addHeader("Authorization", "bearer " + access_token)
    .execute()
    .returnContent();

m = new ObjectMapper().readValue(c.toString(), new TypeReference<>(){});

System.out.println(String.format("%s %s %s", m.get("name"),
    m.get("email"), m.get("user_id")));
```

Example Ruby

```
require "rubygems"
require "net/https"
require "json"
require "uri"

...

# Verify that the access token belongs to us
# The token must be url-encoded when passed to tokeninfo
uri = URI.parse("https://api.amazon.com/auth/o2/tokeninfo?access_token="
  + URI.encode(access_token))
req = Net::HTTP::Get.new(uri.request_uri)
http = Net::HTTP.new(uri.host, uri.port)
http.use_ssl = true
http.verify_mode = OpenSSL::SSL::VERIFY_PEER

response = http.request(req)
decode = JSON.parse(response.body)

if decode['aud'] != 'YOUR-CLIENT-ID'
  # the access token does not belong to us
  raise "Invalid token"
end

# Exchange the access token for user profile
# The token must NOT be url-encoded when passed to profile
uri = URI.parse("https://api.amazon.com/user/profile")
req = Net::HTTP::Get.new(uri.request_uri)
req['Authorization'] = "bearer " + access_token
http = Net::HTTP.new(uri.host, uri.port)
http.use_ssl = true
http.verify_mode = OpenSSL::SSL::VERIFY_PEER

response = http.request(req)
decode = JSON.parse(response.body)

puts sprintf "%s %s %s", decode['name'], decode['email'],
  decode['user_id']
```

2. Launch your website and confirm you can log in with your Amazon.com credentials.

Finish Integration with Your Website

Now you know how to add Login with Amazon to your website. The next steps are to integrate Amazon user accounts into your account management system and use these to personalize your website for Amazon customers. For more information, see:

- The Login with Amazon [Developer's Guide for Websites](#)
- <https://login.amazon.com/documentation/combining-user-accounts>

Need more help? Check out our [forums](#).

Glossary

access scope	An access scope defines the type of user profile data the client is requesting. The first time a user logs in, they see a list of the items in the access scope and must agree to provide the data to the client in order to proceed.
access token	An access token is granted by the authorization server when a user logs in to a site. An access token is specific to a client, a user, and an access scope . A client must use an access token to retrieve customer profile data.
allowed JavaScript origin	A JavaScript origin is the combination of protocol, domain, and port where a JavaScript call originates. By default, web browsers block JavaScript calls from one origin that try to call script on another origin. The Login with Amazon SDK for JavaScript allows calls from other origins if they are specified as part of an application .
allowed Return URL	<p>A return URL is an address on a website that uses Login with Amazon. The authorization service redirects users to this address when they complete login.</p> <p>See also redirect URL (p. 11)</p>
API key	This is an identifier that Login with Amazon SDKs use to identify a mobile app to the authorization service . API keys are generated when you register a mobile app.
application	An application is the registration that contains information the authorization service needs to verify a client before that client can access customer profiles . It also contains basic information about your business that is displayed to users when they first login to one of your apps and are asked to share information with you.
AppStore ID	An AppStore ID uniquely identifies a mobile app in the Amazon AppStore.
authorization code	An authorization code is a value used by the Authorization Code grant to allow a website to request an access token .
Authorization Code grant	An Authorization Code grant is an authorization grant that uses server-based processing to request an access token . Using the authorization code grant, the server receives an authorization code as a query parameter after the user logs in. The server exchanges the authorization code, client identifier , and client secret for an access token and a refresh token .
authorization grant	<p>An authorization grant is the process where the authorization service verifies a client website's request for access to a customer profile. An authorization grant requires a client identifier and an access scope, and may require a client secret. If the process succeeds, the website is granted an access token.</p> <p>There are two types of authorization grants, an Implicit grant and an Authorization Code grant.</p>
authorization service	The Login with Amazon authorization service is the collection of endpoints provided by Amazon that allows a client to login a user through authorization grants . The authorization service presents the login screen

	and the permissions screen to users. It provides access tokens , refresh tokens , and customer profile data to Login with Amazon clients.
bundle identifier	The bundle identifier is a unique identifier for an iOS app. They normally take the form of <code>com.companyname.appname</code> .
client	A client is a website or mobile app that uses Login with Amazon.
client identifier	The client identifier is a value assigned to the client when they register with Login with Amazon. It is used in conjunction with the client secret to verify the identity of the client when they request an authorization grant from the authorization service . The client identifier is not secret.
client secret	The client secret, like the client identifier , is a value assigned to the client when they register with Login with Amazon. It is used in conjunction with the client identifier to verify the identity of the client when they request an authorization grant from the authorization service . The client secret must be kept confidential.
consent screen	When a user logs into a website or mobile app for the first time, they are presented with a consent screen if the app requests profile data. The consent screen shows the name, logo image file , and privacy notice URL associated with app, along with the access scope the app is requesting.
customer profile	A customer profile contains information about the Login with Amazon customer, including their name, email address, postal code, and a unique identifier. A website must obtain an access token before they can obtain a customer profile. The kind of profile data returned is determined by the access scope .
Implicit grant	An Implicit Grant is an authorization grant that can be completed using only the user's web browser. Using the implicit grant, the browser receives an access token as a URI fragment. An implicit grant requires a client identifier and an access scope . The implicit grant does not return a refresh token .
login screen	The login screen is an HTML page presented to users when they try to login to a website or mobile app using Login with Amazon. Users can enter an existing Amazon account or create a new one from this page.
logo image file	A PNG file provided by the client when setting up an application . This is displayed on the permissions screen if the user has not granted access to the client website. The logo represents the client website.
package name	A package name is a unique identifier for an Android app. They normally take the form of <code>com.companyname.appname</code> .
privacy notice URL	A URL provided by the client when setting up an application . This is displayed on the permissions screen if the user has not granted access to the client website. The URL should direct users to the privacy policy for the client website.
redirect URL	A URL provided by the client to the authorization service . After the user logs in, the service will redirect the user's browser to this address. See also allowed Return URL (p. 10)
refresh token	A refresh token is granted by the authorization service when the client uses the Authorization Code grant . A client can use a refresh token to request a new access token when the current access token expires.

signature	A signature is an MD5 hash value embedded in a mobile app that verifies the identity of the app. They normally take the form of 01:02:03:04:05:06:07:08:09:0a:0b:0c:0d:0e:0f.
user	A user is a person who visits a client website and tries to log in using Login with Amazon.
version	A version is a particular type of Login with Amazon client registered to an application . A Login with Amazon application can have multiple versions, each supporting either Android, iOS, or web.