Login with Amazon

SDK for JavaScript v1.0

Reference

Login with amazon

Login with Amazon: SDK for JavaScript Reference

Contents

Introduction

Topics

This is the *Login with Amazon SDK for JavaScript Reference*. This documents contains reference information for the Login with Amazon SDK for JavaScript, as well as information about how to load the SDK.

Login with Amazon is a web service that enables Amazon customers to login to your web or mobile app using their Amazon credentials. Once they have logged in, your app can access some information from their Amazon profile. For more information, visit https://login.amazon.com.

Loading the SDK for JavaScript

Before you can call the methods of the SDK you must include the SDK in your website. To include the SDK, add the following code to somewhere inside the `<body>` tag on your website:

```
<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>
```

This code includes the SDK from Amazon's Content Delivery Network. This ensures you have the latest version. The code is included asynchronously so it is safe to use it anywhere within your page.
Once the SDK has loaded, it will invoke the `window.onAmazonLoginReady` method on the global `window` object. In this method, call `amazon.Login.setClientId`, passing your client identifier. If you do not know your client identifier, visit the App Console at login.amazon.com.

The JavaScript SDK requires the `amazon-root` element to be present in the page. Do not apply any CSS styles to the `amazon-root` element. The `amazon-root` element must not be hidden using `display: none` or `visibility: hidden`, or some parts of the SDK may not work properly in Internet Explorer.

amazon.Login Methods

All of the functions in `login.js` are found in the `amazon.Login` namespace. These functions allow you to identify your client application, request an access token, and exchange an access token for customer profile information.

authorize

```
    AuthorizeRequest authorize(options, next);
```

Requests authorization according to `options` then redirects or calls `next`. Depending on the options set, a popup window will open to allow the user to login, or the window will redirect to the login page. You must call `setClientId` prior to calling `authorize`. You must call `authorize` prior to calling `retrieveProfile`.

This method returns an AuthorizeRequest object. Call `onComplete` on that object to register a callback function or redirect URI, similar to the *`next`* parameter. Once the request is complete, the object will contain properties detailing the response (such as an access token or authorization code).

Response caching

When `authorize` receives a valid access token response, it automatically caches the token and associated metadata for reuse. This cache persists across page reloads and browser sessions. If a subsequent `authorize` call can be fulfilled by the cached token response, the SDK will reuse that token instead of requesting a new authorization. Use `options.interactive` (defined below) to override this behavior.

authorize Parameters:

Parameter	Required	Type	Description
options	Yes	Object	Determines whether the login and consent screens appear in a popup window, or on a separate browser page. Parameters described below.
next	No	Function or String	A URI to redirect the browser response, or a JavaScript function to call with the authorization response.

NOTE: If `next` is a URI, once the user logs in the current window will be redirected to the URI and the authorization response will be added to the query string. The URI must use the HTTPS protocol and belong to the same domain as the current window.

options Parameters:

Parameter	Required	Type	Description
interactive	Yes	String	Specifies when to show a login screen to the user. • `auto` will attempt to authorize using a cached token. If the cached token fails or does not exist, initiate a new authorization, showing login and consent screens if necessary. • `always` will ignore cached tokens and always initiate a new authorization. • `never` will used the cached token; if the token does not work, `authorize` will return `invalid_grant`. Defaults to `auto`.
popup	Yes	Boolean	Specifies whether the login and consent screens will open in a popup window. • `true` to use a popup window for login and consent. • `false` to redirect the current browser window to the authorization dialog. Defaults to `true`. If `false`, the `next` parameter in the authorize request **MUST** be a redirect URL. **Note**: popup windows are not supported in iOS WebView-based apps.
response_type	Yes	String	The grant type requested. • `token` to request an Implicit grant. • `code` to request an Authorization Code grant. Defaults to `token`.

scope	Yes	String or Array[String]	The access scope requested. Must be `profile`, `profile:user_id`, `postal_code`, or some combination.
state	No	String	An opaque value used by the client to maintain state between this request and the response. The Login with Amazon authorization service will include this value when redirecting the user back to the client. It is also used to prevent cross-site request forgery. For more information see Cross-site Request Forgery.

For example:

```
options = { scope: 'profile' };
amazon.Login.authorize(options, 'https://example.org/redirect_here')

// on success the current window will redirect to:
// https://example.org/redirect_here?access_token=XYZ&token_type=bearer&…

// on failure the current windowwill redirect to:
// https://example.org/redirect_here?error=access_denied&…
```

If `next` is a callback function, it will be invoked with a response object containing the fields of the authorization response.

```
options = { scope: 'profile' };
amazon.Login.authorize(options, function(response)    {
  if ( response.error ) {
    alert('oauth error ' +    response.error); return;
  }
  alert('success: ' +   response.access_token);
});
```

Returns:

An `AuthorizeRequest` object. `AuthorizeRequest` allows callers to register a callback function or redirect URL to use when the login request is complete. It also allows callers to get the current status of the request. When the request is complete, new properties are added to AuthorizeRequest based on the type of authorization request. If the request fails, error properties are added to the object.

getClientId

```
    getClientId();
```

Gets the client identifier that will be used to request authorization. You must call `setClientId` before calling this function.

Parameters:

None.

Returns:

clientId - (String). The client ID assigned to your application. Maximum size of 100 bytes.

See Also:

- setClientId

logout

```
logout();
```

Logs out the current user after a call to authorize.

Parameters:

None.

Returns:

None.

Examples:

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

See Also:

- authorize

retrieveProfile

```
retrieveProfile(accessToken, callback);
```

Retrieves the customer profile and passes it to a callback function. Uses an access token provided by authorize.

Parameters:

Parameter	Required	Type	Description
accessToken	No	String	An access token. If this parameter is omitted, retrieveProfile will call authorize, requesting the profile scope.

Note: If the `accessToken` is omitted, `retrieveProfile` only requests the profile scope. To get access to the `postal_code` scope without passing an access token, you can call `authorize` yourself:

```
amazon.Login.authorize({ scope: 'postal_code profile' }, function
    () { amazon.Login.retrieveProfile(function (response) {
        // Display profile information.
    });
});
```

Callback (callback):

```
  retrieveProfile(accessToken, callback);
```

Called with the profile data or an error string.

Callback Response Object Parameters:

Parameter	Type	Description
success	Boolean	Indicates whether the request was successful. `True` or `false`.
error	String	Included if the request failed, and contains an error message.
profile	Object	Included if the request was successful, and contains the user's profile information. The values in this object depend on the scope indicated in the authorize request. This object is defined below.

profile Parameters:

Parameter	Type	Description
CustomerId	String	Uniquely identifies the logged-in user for this caller. Only present if the `profile` or `profile:user_id` scopes are requested and granted.
Name	String	The customer's name. Only present if the `profile` scope is requested and granted.
PostalCode	String	The postal code of the customer's primary address. Only present if the `postal_code` scope is requested and granted.
PrimaryEmail	String	The primary email address for the customer. Only present if the `profile` scope is requested and granted.

Examples:

```
<script type="text/javascript">
document.getElementById('LoginWithAmazon').onclick = function() {
      setTimeout(window.doLogin, 1); return false;
};
window.doLogin = function() {
      options = {};
      options.scope = 'profile';
      amazon.Login.authorize(options, function(response) {
            if ( response.error ) {
                  alert('oauth error ' + response.error);
                  return;
            }
      amazon.Login.retrieveProfile(response.access_token, function(response)
      {
            alert('Hello, ' + response.profile.Name);
            alert('Your e-mail address is ' + response.profile.PrimaryEmail);
            alert('Your unique ID is ' + response.profile.CustomerId);
            if ( window.console && window.console.log )
                  window.console.log(response);
            });
      });
};
</script>
```

```
var access_token = 'Atza|EKdsnskdna…';
// obtained from authorization response

amazon.Login.retrieveProfile(access_token, function(response){
  if ( response.success ) {
    alert('Hello, ' +response.profile.Name);
    alert('Your e-mail address is ' + response.profile.PrimaryEmail);
    alert('Your unique ID is ' + response.profile.CustomerId);
  }
  else {
    alert('Oh no! An error happened: ' +  response.error);
  }
});
```

See also:

authorize

setClientId

```
setClientId(clientId);
```

Sets the client identifier that will be used to request authorization. You must call this function before calling `authorize`.

Parameters:

clientId - (String) - **Required**. The client ID assigned to your application.

Returns:

None.

Examples:

```
window.onAmazonLoginReady = function() {
    amazon.Login.setClientId('YOUR-CLIENT-
    ID');
};
```

See Also:

- Loading the SDK for JavaScript

setSandboxMode

```
setSandboxMode(sandboxMode);
```

Determines whether or not Login with Amazon should use the Amazon Pay sandbox for requests. To use the Amazon Pay sandbox, call `setSandboxMode(true)` before calling `authorize`.

Parameters:

sandboxMode - (boolean) - **Required**. `true` to use the Amazon Pay sandbox to process requests, otherwise `false`.

Returns:

None.

See Also:

- authorize
- Testing your integration with the Sandbox environment at docs.developer.amazonservices.com

setSiteDomain

```
setSiteDomain(siteDomain);
```

Sets the domain to use for saving cookies. The domain must match the origin of the current page. Defaults to the full domain for the current page.

For example, if you have two pages using the Login with Amazon SDK for JavaScript, `site1.example.com` and `site2.example.com`, you would set the site domain to

`example.com` in the header of each site. This will ensure that the cookies on both sites have access to the same cached tokens.

Parameters:

`siteDomain` - (`string`) - **Required**. The site to store Login with Amazon cookies. Must share the origin of the current page.

Returns:

None.

See Also:

- setUseCookie

setUseCookie

```
setUseCookie(useCookie);
```

Determines whether or not Login with Amazon should use access tokens written to the `amazon_Login_accessToken` cookie. You can use this value to share an access token with another page. Access tokens will still only grant access to the registered account for whom they were created.

When `true`, the Login with Amazon SDK for JavaScript will check this cookie for cached tokens, and store newly granted tokens in that cookie.

Parameters:

`useCookie` - (`boolean`) - **Required**. `true` to store the access token from `authorize` in a cookie, otherwise `false`.

Returns:

None.

See Also:

- authorize
- setSiteDomain

amazon.Login Classes

AuthorizeRequest

The `AuthorizeRequest` class is used in response to an authorize call. `AuthorizeRequest` allows callers to register a callback function or redirect URL to use when the login request is complete. It also allows callers to get the current status of the request. When the request is complete, `AuthorizeRequest` adds new properties based on the type of authorization request. If the request fails, error properties provide information on the failure.

The following table details which properties are added for each response type:

Response Type	Properties
Authorization Response	code and state.
Access Token Response	access_token, token_type, expires_in, and scope.
Error Response	error, error_description, and error_uri.

onComplete

```
onComplete(next);
```

Registers a callback function or sets a redirect URI to call when the authorization request is complete. If this function is called after the request is complete, the function or redirect will happen immediately. If a callback function is used, the `AuthorizeRequest` will be the first parameter. If a redirect URI is used, the browser will redirect to that URI with the OAuth 2 response parameters included in the query string.

If multiple redirect URLs are set, `AuthorizeRequest` uses the most recent one.

Parameters:

`next` - (`Function` or `String`) A URI to redirect the browser response, or a JavaScript function to call with the authorization response.

access_token

access_token - (`String`) The access token issued by the authorization server.

code

code - (`String`) An authorization code that can be exchanged for an access token.

error

error - (`String`) A short error code indicating why the authorization failed. It can be one of the following:

- access_denied
  The customer or authorization server denied the request.

- invalid_grant
  The authorization server denied the request due to inability to use a cached token.
- invalid_request
  The request is missing a required parameter, has an invalid value, or is otherwise malformed.
- invalid_scope
  One or more of the requested scopes are invalid.
- server_error
  The authorization server encountered an unexpected error. This is analogous to a 500 HTTP status code.
- temporarily_unavailable
  The authorization server is current unavailable due to a temporary condition. This is analogous to a 503 HTTP status code.
- unauthorized_client
  The client is not authorized to perform this request.

error_description

error_description - (`String`) A human-readable description of the error.

error_uri

error_uri - (`String`) A URI for a web page with more information on the error.

expires_in

expires_in - (`Number`) The number of seconds until the access token expires.

scope

scope - (`String`) The scope granted by the authorization server for the access token. Must be `profile`, `profile:user_id`, `postal_code`, or some combination.

state

state - (`String`) The `state` value provided to `authorize` using the `options` object.

status

status - (`String`) The current status of the request. One of `queued`, `in progress`, or complete.

token_type

token_type - (`String`) The type of token issued. Must be bearer.

Cross-site Request Forgery

Cross-site Request Forgery happens when an attacker tricks a user into clicking on a malicious link, where the link goes to a site where the user is currently authenticated. Any commands embedded in that malicious link might be executed automatically because the user is already authenticated on the site, so the user does not see a login screen or any other evidence of malicious activity. In the case of Login with Amazon, Cross-site Request Forgery could be used to mimic a client or an authentication server.

Login with Amazon recommends using the `state` parameter to prevent Cross-site Request Forgery. The client should set the value of the `state` parameter when it initiates an authorization request, and save it to the user's secure session. Unlike the `client id` and `client secret` values, in order for the `state` parameter to be useful in preventing attacks it should be unique, and non-guessable, for each and every authorization request. The authorization server returns the same `state` when communicating with the client to deliver authorization codes and access tokens. To protect users from attacks, the client must ignore communication if the returned `state` parameter doesn't match the value from the initial call.

Calculating the State Parameter

Clients can calculate the `state` parameter value in any way they choose, however, the value should be secure from forgery. Login with Amazon recommends using a securely-generated random string with at least 256 bits of entropy. To calculate a `state` value using this method, use a random number generator suitable for cryptographic operations.

After generating the `state` parameter value, save it to the user's session information, ensuring the information is communicated securely and saved to a secure session. When the `state` is returned by an authorization response, verify the legitimacy of the user by comparing it with the `state` value saved to their session. If the values do not match, you should ignore the authorization response.

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. Access tokens have a maximum size of 2048 bytes. A client must use an access token to retrieve customer profile data.
allowed JavaScript origins	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. When registering a website for Login with Amazon, enter the scheme, domain, and optionally the port, of the webpage which includes the Login with Amazon SDK for JavaScript (for example, http://www.example.com or https://localhost:8080).
allowed return URL	A return URL is an address on a website that uses Login with Amazon. The Login with Amazon authorization service redirects users to this address when they complete login. See also redirect URL.
API key	An identifier that Login with Amazon SDKs use to identify a mobile app to the Login with Amazon 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 each time they use Login with Amazon on your website or mobile app.
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	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. There are two types of authorization grants, an Implicit grant and an Authorization Code grant.
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 grant. The authorization service presents the login screen and the permissions screen to users. It provides access token, refresh token, 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 `com.companyname.appname`.
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 has a maximum size of 100 bytes. The client identifier 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 has a maximum size of 64 bytes. The client secret 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 `com.companyname.appname`.
privacy notice URL	A URL provided by the client when setting up an application. This is displayed on the consent 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.
refresh token	A refresh token is granted by the authorization server 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. Refresh tokens have a maximum size of 2048 bytes.
signature	A signature is a SHA-256 hash value embedded in a mobile app that verifies the identity of the app. They normally take the form of `01:23:45:67:89:ab:cd:ef:01:23:45:67:89:ab:cd:ef:01:23:45:67:89:ab:cd:ef:01:23:45:67:89:ab:cd:ef`.
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.