Login with Amazon

Getting Started Guide

for iOS apps

Login with amazon

Login with Amazon: Getting Started Guide for iOS

Copyright © 2017 Amazon.com, Inc., 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

Introduction

Topics

In this guide we will show you how to add Login with Amazon to your iOS app, using the Login with Amazon SDK for iOS v3.0+.

After completing this guide you should have a working Login with Amazon button in your app that allows users to login with their Amazon credentials. To learn more about the login flow your customers will experience when they use Login with Amazon within your app, please see our Customer Experience Overview for iOS apps.

Install iOS Developer Tools

Install XCode

The Login with Amazon SDK for iOS is provided by Amazon to help you add Login with Amazon to your iOS application. The SDK is intended to be used with the Xcode development environment. The SDK supports apps running on iOS 7.0 and later using ARMv7, ARMv7s, ARM64, i386, and x86_64.

You can install Xcode from the Mac App Store. For more information, see Xcode: What's New on developer.apple.com.

After Xcode is installed, you can Install the Login with Amazon SDK for iOS and Run the Sample App, as described below.

Install the Login with Amazon SDK for iOS

The Login with Amazon SDK for iOS comes in two packages. The first contains the iOS library and supporting documentation. The second contains a sample application that allows a user to log in and view their profile data.

If you have not yet installed Xcode, see the instructions in the Install Xcode section above.

1. Download LoginWithAmazonSDKForiOS.zip and extract the files to a directory on your hard drive.
   You should see a **LoginWithAmazon.framework** directory. This contains the Login with Amazon library.
   At the top level of the zip is a **LoginWithAmazon.docset** directory. This contains the API documentation.
2. See Install the Login with Amazon Library for instructions on how to add the library to an iOS project.

When the Login with Amazon SDK for iOS is installed, you can Create a New Login with Amazon Project after you Register with Login with Amazon.

Run the Sample App

To run the sample application, open the sample in Xcode.

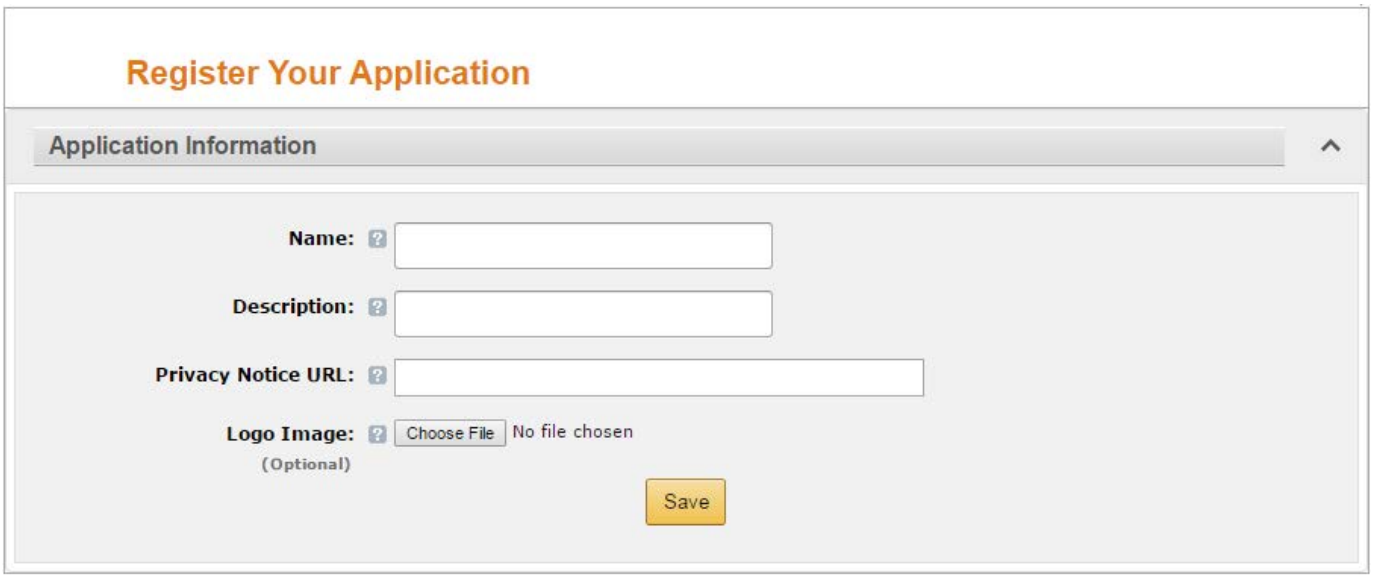
1. Download SampleLoginWithAmazonAppForiOS.zip and copy the **SampleLoginWithAmazonAppForiOS** directory to your **Documents** folder.
2. Start Xcode. If the Welcome to Xcode dialog pops up, click **Open Other**. Otherwise, from the main menu, click **File** and select **Open**.
3. Select the **Documents** folder, and select **SampleLoginWithAmazonAppForiOS/LoginWithAmazonSample/LoginWithAmazonSample.xcodeproj**. Click **Open**.
4. The sample project should now load. When it is finished, choose **Product** from the main menu, and select **Run**.

Register with Login with Amazon

Before you can use Login with Amazon on a website or in a mobile app, you must register an application with Login with Amazon. Your Login with Amazon application is the registration that contains basic information about your business, and information about each website or mobile app you create that supports Login with Amazon. This business information is displayed to users each time they use Login with Amazon on your website or mobile app. Users will see the name of your application, your logo, and a link to your privacy policy. These steps demonstrate how to register your iOS app for use with Login with Amazon.

Register Your Login with Amazon Application

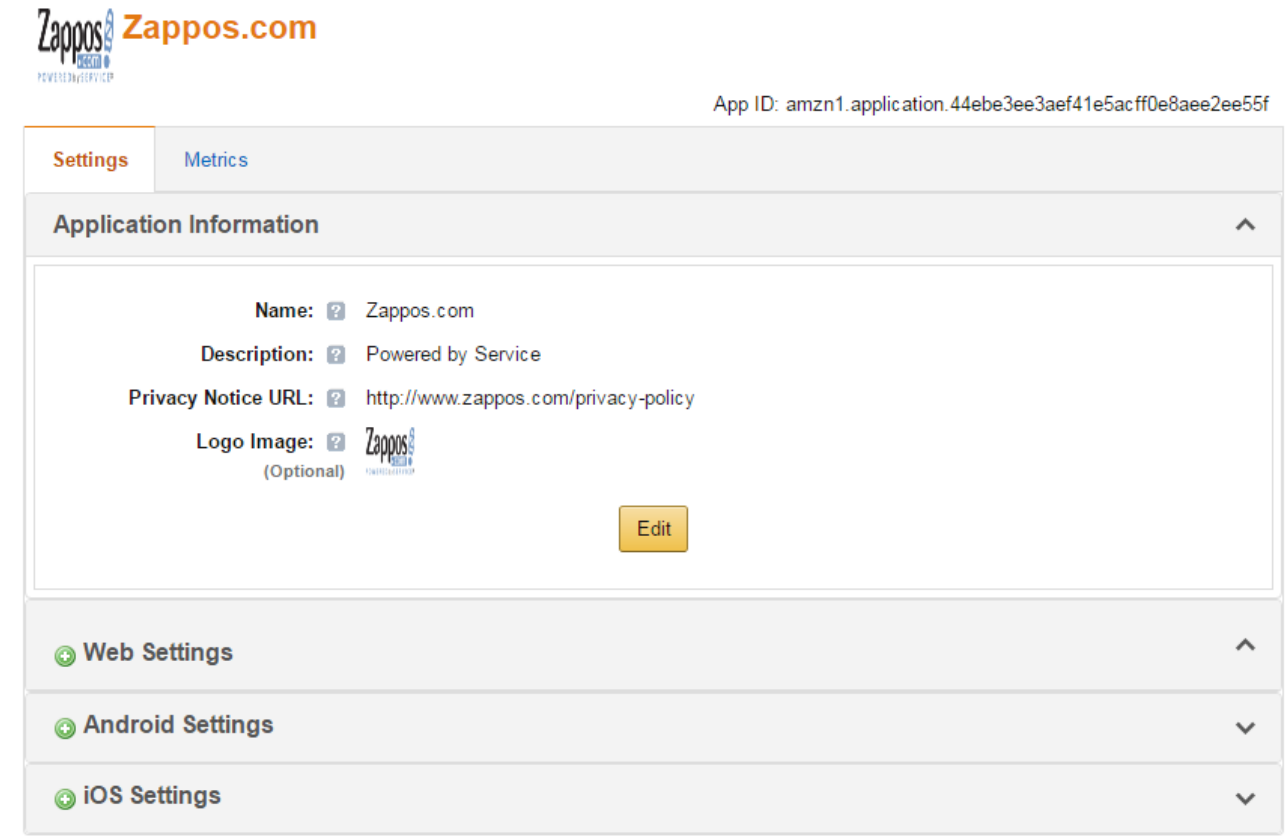
1. Go to https://login.amazon.com.
2. If you have signed up for Login with Amazon before, click **App Console**. Otherwise, click **Sign Up**. You will be redirected to Seller Central, which handles application registration for Login with Amazon. If this is your first time using Seller Central, you will 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, 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. The **Description** helps you differentiate each of your Login with Amazon applications and is not displayed to users.

b. Enter a **Privacy Notice URL** for your application.
   The **Privacy Notice URL** is the location of your company's 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 **Choose File** and locate the applicable image.
   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:



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.

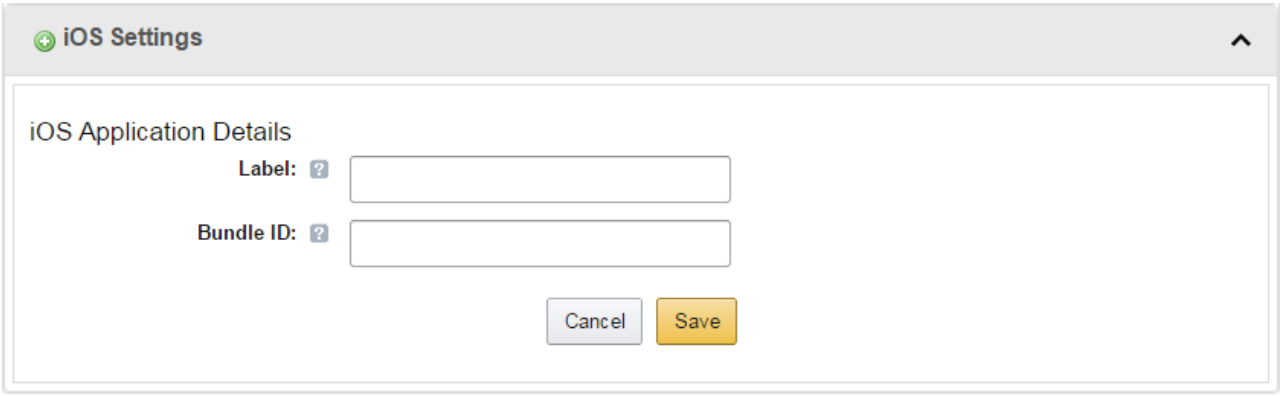
If different versions of your app have different bundle IDs, such as for one or more testing versions and a production version, each version requires its own API Key. From the **iOS Settings** of your app, click the **Add API Key** button to create additional keys for your app (one per version).

Add iOS Settings to your Application

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

To register an iOS App, you have to specify the Bundle identifier for the app project. Login with Amazon will use the bundle ID to generate an API key. The API key will grant your app access to the Login with Amazon authorization service. Follow these steps to add an iOS app to your account:

1. From the **Application** screen, click **iOS Settings**. If you already have an iOS app registered, look for the **Add API Key** button in the **iOS Settings** section.
   The **iOS Application Details** form will appear:



2. Enter the **Label** of your iOS App.
   This does not have to be the official name of your app. It simply identifies this particular iOS app among the apps and websites registered to your Login with Amazon application.
3. Enter your **Bundle ID**. This must match the bundle identifier of your iOS project.
   To determine your bundle identifier, open the project in Xcode. Open the properties list for the project (**<project>-Info.plist**) in the **Project Navigator**. The **Bundle identifier** is one of the properties in the list.
4. Click **Save**.

If different versions of your app have different bundle IDs, such as for one or more testing versions and a production version, each version requires its own API Key. From the **iOS Settings** of your app, click the **Add API Key** button to create additional keys for your app (one per version).

iOS Bundle ID and API Keys

The Bundle identifier is unique to every iOS app. Login with Amazon uses the Bundle ID to construct your API Key. The API Key enables the Login with Amazon authorization service to recognize your app.

Determine a Bundle Identifier for an iOS App

1. Open your app project in Xcode.
2. Open the **Information Property List** for the project (**<project>-Info.plist**) in the **Project Navigator**.
3. Find **Bundle identifier** in the list of properties.

Retrieve an iOS API Key

After you have registered an iOS version and provided a Bundle ID, you can retrieve the API key from the registration page for your Login with Amazon application. You will need to place that API key into your project's property list. Until you do, the app will not be authorized to communicate with the Login with Amazon authorization service.

1. Go to https://login.amazon.com.

2. Click **App Console**.
3. In the **Apps** box, click your application.
4. Find your iOS app under the **iOS Settings** section.
   If you have not already registered an iOS app, see Add iOS Settings to an Application.
5. Click **Generate API Key Value**. A popup window will display your API key. To copy the key, click **Select All** to select the entire key.

   Note: The API Key Value is based, in part, on the time it is generated. Thus, subsequent API Key Value(s) you generate may differ from the original. You can use any of these API Key Values in your app as they are all valid.
6. See Add Your API Key to Your App Property List for instructions on adding the API key to your iOS app.

Create a Login with Amazon Project

In this section, you will learn how to create a new Xcode project for Login with Amazon and configure the project.

Create a New Login with Amazon Project

If you do not yet have an app project for using Login with Amazon, follow the instructions below to create one. If you have an existing app, skip to the Install the Login with Amazon Library section below.

1. Launch **Xcode**.
2. If you are presented with a **Welcome to Xcode** dialog, select **Create a New Xcode Project**. Otherwise, from the **File** menu, select **New** and **Project**.
3. Select the type of project you wish to create and click **Next**.
4. Enter a **Product Name** and a **Company Identifier**. Note your **Bundle Identifier**, and click **Next**.
5. Select a location in which to store your project and click **Create**.

You will now have a new project that you can use to call Login with Amazon.

Install the Login with Amazon Library

If you have not yet downloaded the Login with Amazon SDK for iOS, see Install the Login with Amazon SDK for iOS.

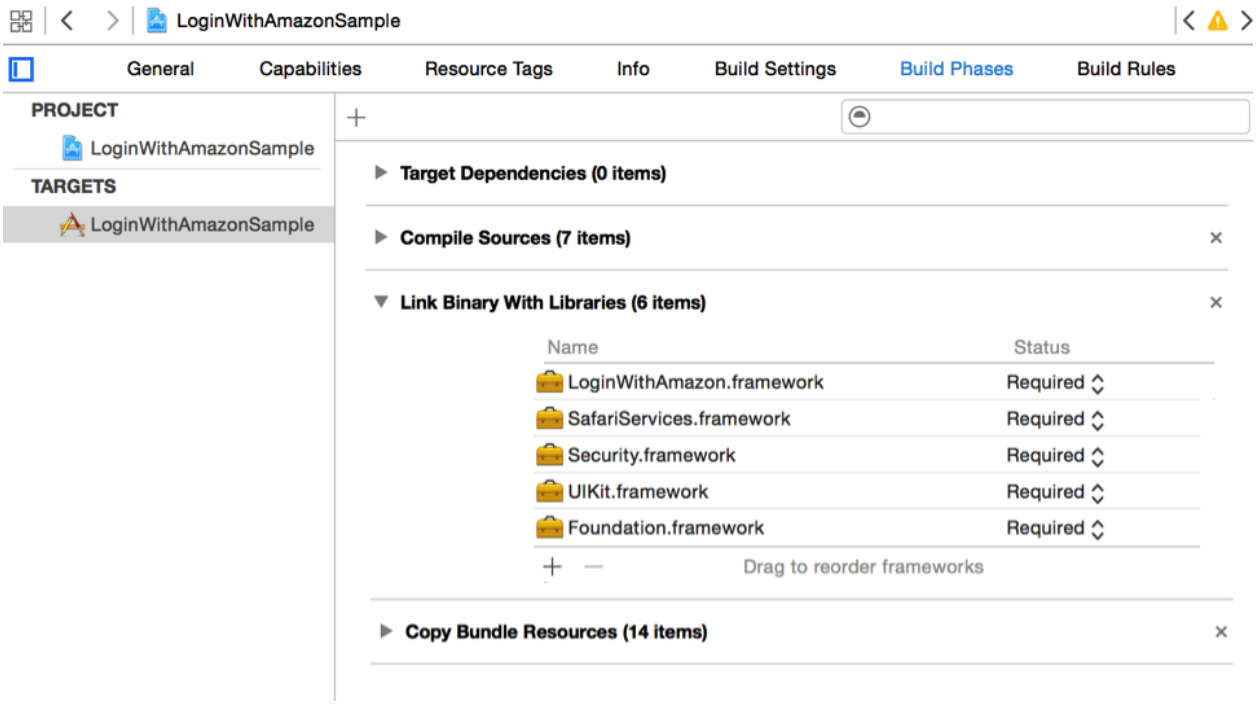
A Login with Amazon project must link the **LoginWithAmazon.framework** and **Security.framework** libraries. You will also need to configure the framework search path to find the Login with Amazon headers.

1. If your project doesn't have a Frameworks folder, right-click the project name in the Navigator pane in Xcode, then click **New Group**.
2. Name the new group **Frameworks**.
3. Select the **Frameworks** folder and click **File** from the Main menu.
4. Select **Add Files to Project**.
5. In the dialog, select **LoginWithAmazon.framework** and click **Add**.
   If you used the Login with Amazon 1.0 library, delete the **login-with-amazon-sdk** directory and

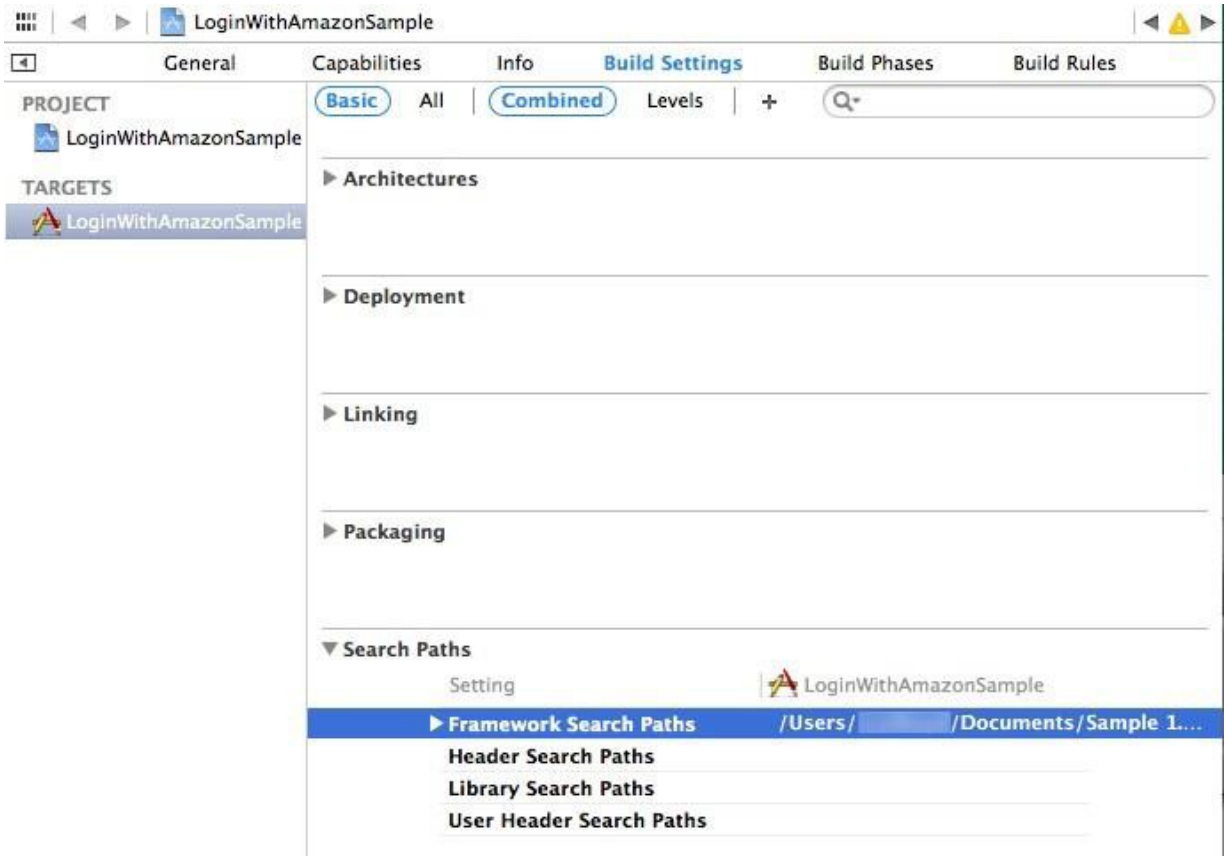
login-with-amazon-sdk.a from the Frameworks folder. Click **Edit** from the main menu and select **Delete**.

6. Select the name of your project in the **Project Navigator**.
   The **Project Editor** will appear in the editor area of the Xcode workspace.

7. Click your project name under **Targets**, and select **Build Phases**. Expand **Link Binary with Libraries** and click the plus sign to add a library.
8. In the search box, enter **Security.framework**. Select **Security.framework** and click **Add**.
9. In the search box, enter **SafariServices.framework**. Select **SafariServices.framework** and click **Add**.



10. Select **Build Settings**. Click **All** to view all settings.
11. Under **Search Paths**, ensure that the **LoginWithAmazon.framework** directory is in the **Framework Search Paths**.
    For example:



12. Before building your project, if you used the Login with Amazon 1.0 library, replace `#import "AIMobileLib.h"`, `#import "AIAuthenticationDelegate.h"`, and `#import "AIError.h"` in your source files with a single `#import <LoginWithAmazon/LoginWithAmazon.h>`. The `LoginWithAmazon.h` line includes all of the Login with Amazon headers at once.

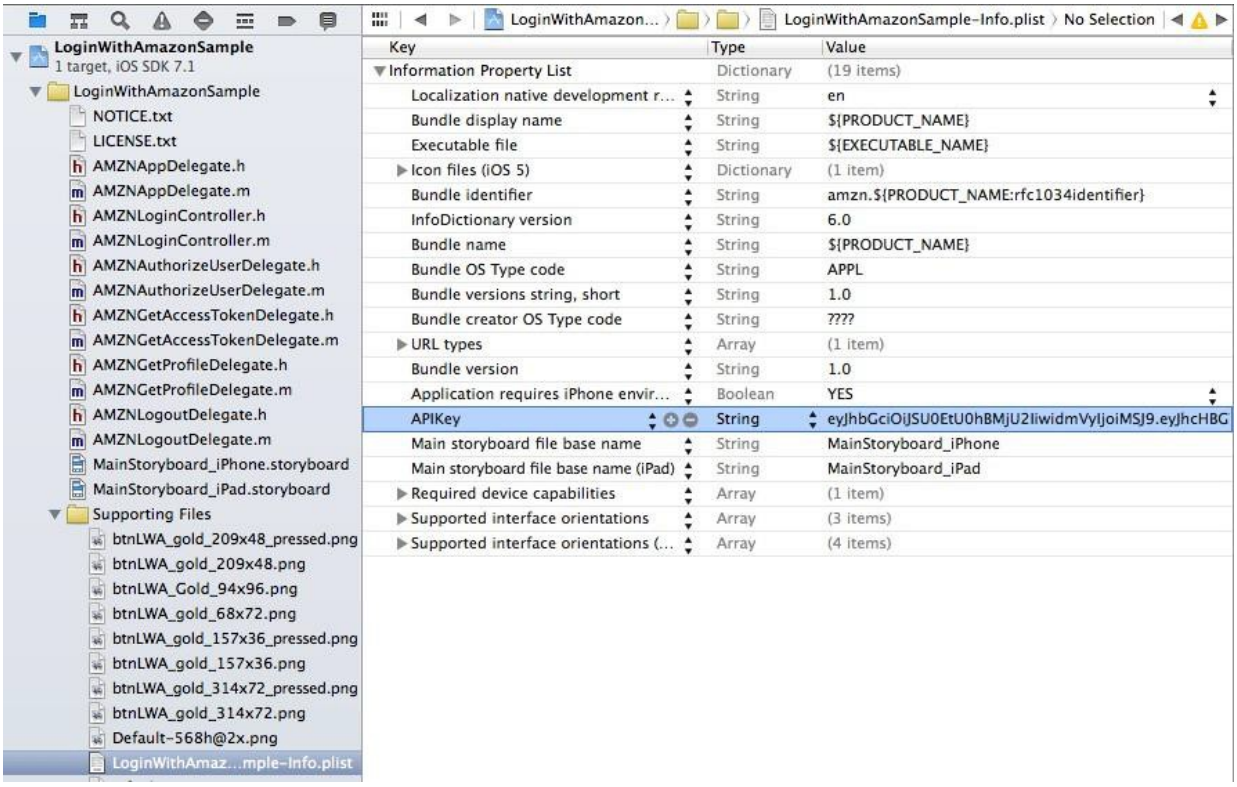
    Additionally, you can remove any references to the 1.0 library path in the **Header Search Paths** or **Library Search Paths**.

13. From the main menu, click **Product** and select **Build**. The build should complete successfully.

Add Your API Key to Your App Property List

When you register your iOS application with Login with Amazon, you are assigned an API key. This is an identifier that the Amazon Mobile Library will use to identify your application to the Login with Amazon authorization service. The Amazon Mobile Library loads this value at runtime from the `APIKey` property value in your application's Information Property List.

1. With your project open, select the **Supporting Files** folder, then select the **<project>-Info.plist** file (where **<project>** is the name of your project). This should open the property list for editing:

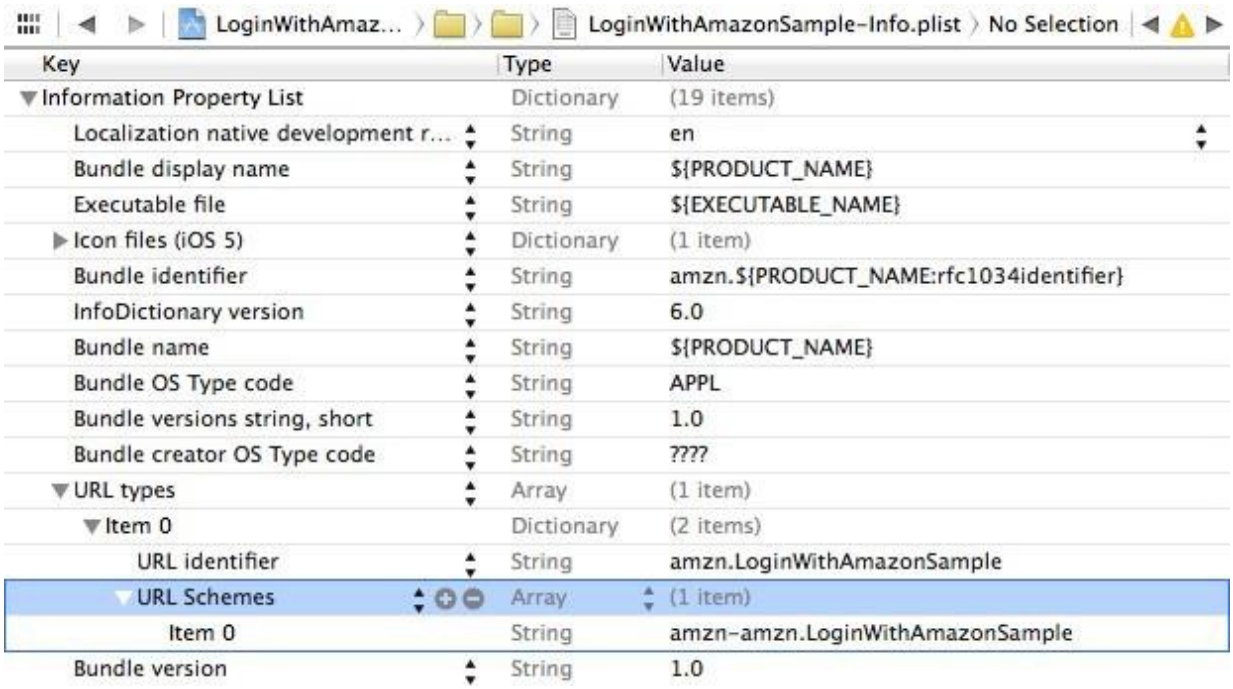


2. Make sure that none of the entries are selected. Then, from the main menu, click **Editor**, and **Add Item**. Enter **APIKey** and press **Enter**.
3. Double-click under the **Value** column to add a value. Paste your API Key as the value.

Add a URL Scheme to Your App Property List

When the user logs in, they will be presented with an Amazon login page. In order for your app to receive confirmation of their login, you must add a URL scheme so that the web page can redirect back to your app. The URL scheme must be declared as **amzn-<bundleID>** (for example, **amzn-com.example.app**). For more information, see Using URL Schemes to Communicate with Apps on developer.apple.com.

1. With your project open, select the **Supporting Files** folder, then select the **<project>-Info.plist** file (where **<project>** is the name of your project). This should open the property list for editing:



2. Make sure that none of the entries are selected. Then, from the main menu, click **Editor**, and **Add Item**. Enter or select **URL types** and press **Enter**.
3. Expand **URL types** to reveal **Item 0**. Select **Item 0** and, from the main menu, click **Editor** and **Add Item**. Enter or select **URL Identifier** and press **Enter**.
4. Select **Item 0** under **URL Identifier** and double-click under the **Value** column to add a value. The value is your bundle ID. You can find your bundle ID listed as **Bundle identifier** in the property list.
5. Select **Item 0** under **URL types** and, from the main menu, click **Editor** and **Add Item**. Enter or select **URL Schemes** and press **Enter**.
6. Select **Item 0** under **URL Schemes** and double-click under the **Value** column to add a value. The value is your bundle ID with **amzn-** prepended (for example, **amzn-com.example.app**). You can find your bundle ID listed as **Bundle identifier** in the property list.

Add Login with Amazon Buttons to your App

Login with Amazon provides several standard buttons you can use to prompt users to log in from your app. This section gives steps for downloading an official Login with Amazon image and pairing it with an iOS `UIButton`.

1. Add a standard `UIButton` to your app.
   For tutorials and information on how to add a button to an app, see Creating and Configuring View Objects and Start Developing iOS Apps Today on developer.apple.com.
2. Add the **Touch Up Inside** event for the button to a method named `onLoginButtonClicked`. Leave the implementation blank for now. The Creating and Configuring View Objects and Start Developing iOS Apps Today documents on

developer.apple.com include steps on adding a button event.

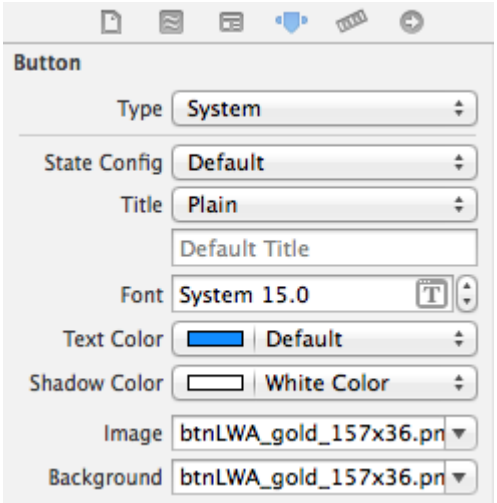
3. Choose a button image.

   Consult our Login with Amazon Style Guidelines for a list of buttons that you can use in your app. Download a copy of the **LWA_for_iOS.zip** file. Find your preferred button in both the 1x and 2x directories and extract them from the zip. Extract the **_Pressed** version of your button if you want to show the button in a Selected state.

4. Add the images to your project.
   a. In Xcode, with your project loaded, click **File** from the main menu and select **Add Files to "project"**.
   b. In the dialog, select the button image file(s) that you downloaded and click **Add**.
   c. The buttons should now be in the project under your project directory. Move them to the **Supporting Files** folder.

5. Add the image to your button.

   To enable the image for your button, you can modify the button attribute or use the `setImage:forState` method on the `UIButton` object. Follow these steps to modify the image attribute for your button:
   a. Open the storyboard for your app.
   b. Select the button in your storyboard by clicking it or selecting it from the **View Controller Scene** tree.
   c. In the **Utilities** window, open the **Attributes Inspector**.



   d. At the top of the **Attribute Inspector**, set the **Type** of button to **System**.
   e. In the second group of settings, select **Default** for **State Config**.
   f. In the second group of settings, drop down the **Image** setting.
   g. Select the Login with Amazon button graphic you added to the project. Do not select the 2x version: it will be loaded automatically on high density display (Retina) devices.
   h. Set the same image for the **Background** setting.
   i. If you want to specify a pressed version of the button, select **Selected** for **State Config**, and set the **Image** to the **_Pressed** version of your button.
   j. On the storyboard, adjust the size of your button to accommodate the image, if necessary.

Use the SDK for iOS APIs

In this section, you will add code to your project to sign in a user with Login with Amazon.

Connect the AppDelegate

Implement `application:openURL:options:` in the class in your project that handles the `UIApplicationDelegate` protocol. By default, this will be the `AppDelegate` class. When a user successfully logs into your app using Login with Amazon, they will be redirected from the Amazon login screen back to your app based on the URL Scheme you added to your App Property List earlier. In order to handle this redirect, you must implement the `application:openURL:options:` method, which returns YES if the URL is successfully handled.

The Login with Amazon SDK for iOS provides a library function, `handleOpenURL:sourceApplication:` which handles any redirect URL sent from Amazon pages. It returns YES if the URL is successfully handled by the SDK. Call this method within the `application:openURL:options:` method.

To invoke this method, you will need to import `<LoginWithAmazon/LoginWithAmazon.h>`.

```
import <LoginWithAmazon/LoginWithAmazon.h>

@implementation AppDelegate
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)
  url options:(NSDictionary<UIApplicationOpenURLOptionsKey,id> *)options {

   return [AMZNAuthorizationManager handleOpenURL:url
   sourceApplication:options[UIApplicationOpenURLOptionsSourceApplicationKey]];
  }
@end
```

Handle the Login Button and Get Profile Data

This section explains how to call the `authorize:withHandler:` API to login a user. This includes creating an `onLoginButtonClicked:` listener for your Login with Amazon button.

1. Add Login with Amazon to your iOS project. See Install the Login with Amazon Library.

2. Import the Login with Amazon API to your source file.
   To import the Login with Amazon API, add the following `#import` statements to your source file:

   ```
   #import <LoginWithAmazon/LoginWithAmazon.h>
   ```

3. Call `authorize:withHandler:` in `onLoginButtonClicked`.
   If you followed the steps in Add a Login with Amazon Button to Your App, you should have an `onLoginButtonClicked:` method linked to a Login with Amazon button. In that method, call `authorize:withHandler:` to prompt the user to login and authorize your application.

This method will enable the user to sign in and consent to the requested information in one of the following ways:

1. Switches to web view in a secure context (if the Amazon Shopping app is installed to the device)
2. Switches to Safari View Controller (on iOS 9 and later)
3. Switches to the system browser (on iOS 8 and earlier)

The secure context for the first option is available when the Amazon Shopping app is installed to the device. If the user is already signed in to the Amazon Shopping app, this API will skip the sign in page, leading to a **Single Sign-On (SSO)** experience. See Customer Experience in iOS apps to learn more.

The first parameter to `authorize:withHandler:` is an `AMZNAuthorizeRequest` object that indicates what **scope** your application is requesting authorization for. A scope encompasses the user data you are requesting from Login with Amazon. The first time a user logs in to your app, they will be presented with a list of the data you are requesting and asked for approval.

Login with Amazon currently supports the following scopes:

Scope name	Description
profile	Gives access to the user's name, email address, and Amazon account ID.
profile:user_id	Gives access to the user's Amazon account ID only.
postal_code	Gives access to the user's zip/postal code on file for their Amazon account.

Use the methods defined in `AMZNProfileScope` to get a scope object and add it to your `AMZNAuthorizeRequest` object. See the sample code below for details.

The second parameter to `authorize:withHandler:` is `AMZNAuthorizationRequestHandler`, described in the next step.

4. Create an `AMZNAuthorizationRequestHandler` block object. `AMZNAuthorizationRequestHandler` processes the result of the `authorize:withHandler:` call. To learn more about objective-c blocks, see Working with Blocks on developer.apple.com.

   The first parameter of `AMZNAuthorizationRequestHandler` is an `AMZNAuthorizeResult` object. After a user is authorized successfully, `AMZNAuthorizeResult` will contain an access token which can be used to access a user's profile data, and an `AMZNUser` object, which contains the user's profile data.

   The second parameter of `AMZNAuthorizationRequestHandler` is a Boolean called `userDidcancel`. This parameter will be set to true if the user:

   1. Closes the Safari View Controller during login and authorization (on iOS 9 and later)
   2. Closes the web view in the Amazon Shopping app
   3. Cancels the login or rejects authorization

   The third parameter of `AMZNAuthorizationRequestHandler` is an `NSError` object which

contains error details if the login and authorization fails due to the SDK or authorization server.

```
- (IBAction)onLogInButtonClicked:(id)sender {
 // Build an authorize request.
 AMZNAuthorizeRequest *request = [[AMZNAuthorizeRequest alloc] init];
   request.scopes = [NSArray arrayWithObjects:
   // [AMZNProfileScope userID],
   [AMZNProfileScope profile],
   [AMZNProfileScope postalCode]];

 // Make an Authorize call to the Login with Amazon SDK.
 [[AMZNAuthorizationManager sharedManager] authorize:request
  withHandler:^(AMZNAuthorizeResult *result, BOOL
 userDidCancel, NSError *error) {
     if (error) {
         // Handle errors from the SDK or authorization server.
     } else if (userDidCancel) {
         // Handle errors caused when user cancels login.
     } else {
         // Authentication was successful.
         // Obtain the access token and user profile data.
         NSString *accessToken = result.token;
         AMZNUser *user = result.user;
         NSString *userID = user.userID;
    }
 }];
}
```

Fetch User Profile Data

As long as a user is logged in and authorized to your app, you can fetch their user profile data at any time. This section explains how to use the `fetch:` method of the `AMZNUser` class to retrieve the most up-to-date user profile data for users who are currently authorized. The profile data you can retrieve is based on the scope indicated in the `authorize` call.

1. Call `AMZNUser fetch:`.
   This method will fetch profile data via an `AMZNUserFetchRequestHandler` block object. The first parameter to `AMZNUserRequestHandler` is an `AMZNUser` object. The `AMZNUser` object can include a `userID`, `name`, `email`, and `postalCode`, depending on the requested scope.

```
[AMZNUser fetch:^(AMZNUser *user, NSError *error) {
  if (error) {
    // Error from the SDK, or no user has authorized to the app.
  } else if (user) {
    NSString *userID = user.userID;
    //NSString *name = user.name;
    //NSString *email = user.email;
    //NSString *postalCode = user.postalCode;
  }
}];
```

Check for User Login at Startup

If a user logs into your app, closes the app, and restarts the app later, the app is still authorized to retrieve data. The user is not logged out automatically. At startup, you can show the user as logged in if your app is still authorized. This section explains how to use `authorize:withHandler:` to see if the app is still authorized.

1. Create an `AMZNAuthorizeRequest` object and specify scopes that indicate the user data your application is requesting authorization for. For more information on scopes, see Handle the Login Button and Get Profile Data.
2. Set `AMZNAuthorizeRequest.interactiveStrategy` to `AMZNInteractiveStrategyNever`. `AMZNAuthorizeRequest` supports multiple strategies for prompting user login:
    - `AMZNInteractiveStrategyAuto` (default): The SDK looks for a locally stored authorization grant from previous `authorize:withHandler:` responses. If one is available, valid, and contains all requested scopes, the SDK will return a successful response via `AMZNAuthorizationRequestHandler`, and will not prompt the user to login. Otherwise, the user will be prompted to login.
    - `AMZNInteractiveStrategyAlways`: The SDK will always prompt the user to login regardless of whether they have previously been authorized to use the app. When the user is prompted, the SDK will remove all locally cached authorization grants for the app.
    - `AMZNInteractiveStrategyNever`: The SDK looks for a locally stored authorization grant from previous `authorize:withHandler` responses. If one is available, valid, and contains all requested scopes, the SDK will return an `AMZNAuthorizeResult` object that contains an access token and user profile data. Otherwise, it will return an `NSError` object via `AMZNAuthorizationRequestHandler`.

```
// Build an authorize request.
   AMZNAuthorizeRequest *request = [[AMZNAuthorizeRequest alloc] init];
   request.scopes = [NSArray arrayWithObjects:
   // [AMZNProfileScope userID],
    [AMZNProfileScope profile],
    [AMZNProfileScope postalCode]];

    request.interactiveStrategy = AMZNInteractiveStrategyNever;

    [[AMZNAuthorizationManager sharedManager] authorize:request
    withHandler:^(AMZNAuthorizeResult *result, BOOL
     userDidCancel, NSError *error) {
       if (error) {
          // Error from the SDK, indicating the user was not previously
authorized to your app for the requested scopes.
       } else {
           // The user was previously authorized to your app.
           // Obtain the access token and user profile data.
           NSString *accessToken = result.token;
           AMZNUser *user = result.user;
           NSString *userID = user.userID;
         }
       }];
```

Clear Authorization Data and Log Out a User

This section explains how to use the `signOut` method to clear the user's authorization data from both the `AIMobileLib` local data store, and the authorization server. The user will have to login again in order for the app to retrieve profile data. Use this method to log out a user, or to troubleshoot login problems in the app.

1. Implement a logout mechanism.
   When a user has successfully logged in, you should provide a logout mechanism so they can clear their profile data and previously authorized scopes. Your mechanism might be a hyperlink, button, or a menu item.

2. Call `signOut:`.
   Call `signOut:` in your logout handler to remove a user's authorization data (access tokens, profile) from the local store, and their authentication state from the server. The input parameter to `signOut` is an `AMZNAuthorizationRequestHandler` block object. The block should detect and handle `NSError` objects, which are returned when `signOut:` fails.

```
[[AMZNAuthorizationManager sharedManager] signOut:^(NSError * _Nullable
error) {
  if (!error) {
    // error from the SDK or Login with Amazon authorization server.
  }
}];
```

Test your Integration

Launch your app in an iOS device or simulator and confirm you can log in with your Amazon.com credentials.

Note: When testing on iOS10 simulators, you may see the error message **APIKey for the Application is invalid** for an `authorizeUserForScopes` request, or **Unknown Error Code** for a `clearAuthorizationState` request. This is a known bug with Apple which occurs when the SDK tries to access the keychain. Until Apple resolves the bug, you can work around it by enabling **Keychain Sharing** for your app under the **Capabilities** tab of your app's target. This bug only impacts simulators. You can test on actual iOS10 devices without using any workaround.