
Login with Amazon

Getting Started Guide for iOS

Version 1.2

Login with Amazon: Getting Started Guide for iOS

Copyright © 2014 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 iOS..... | 1 |
| Install Xcode..... | 2 |
| Install the Login with Amazon SDK for iOS..... | 3 |
| Run the Sample App..... | 4 |
| Register with Login with Amazon..... | 5 |
| iOS Bundle ID and API Keys..... | 8 |
| Create a Login with Amazon Project..... | 10 |
| Create a New Login with Amazon Project..... | 11 |
| Install the Login with Amazon Library..... | 11 |
| Add Your API Key to Your App Property List..... | 13 |
| Add a URL Scheme to Your App Property List..... | 14 |
| Add a Login with Amazon Button to Your App..... | 15 |
| Using the SDK for iOS APIs..... | 17 |
| Handling the Login Button and Getting Profile Data..... | 18 |
| Checking for User Login at Startup..... | 21 |
| Clearing Authorization State and Logging Out a User..... | 23 |

Getting Started for iOS

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

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 6.0 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](#).

Once Xcode is installed, you can [Install the Login with Amazon SDK for iOS](#) (p. 3).

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 login and view their profile data.

If you have not installed Xcode, see [Install Xcode](#) (p. 2).

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](#) (p. 11) for instructions on how to add the library to an iOS project.

Once the Login with Amazon SDK for iOS is installed, you can [Create a New Login with Amazon Project](#) (p. 11) after you [Register with Login with Amazon](#) (p. 5).

Run the Sample App

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

1. Download [SampleLoginWithAmazonAppForiOS.zip](#) and copy the `SampleLWAAppiOS` 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 `SampleLWAAppiOS/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

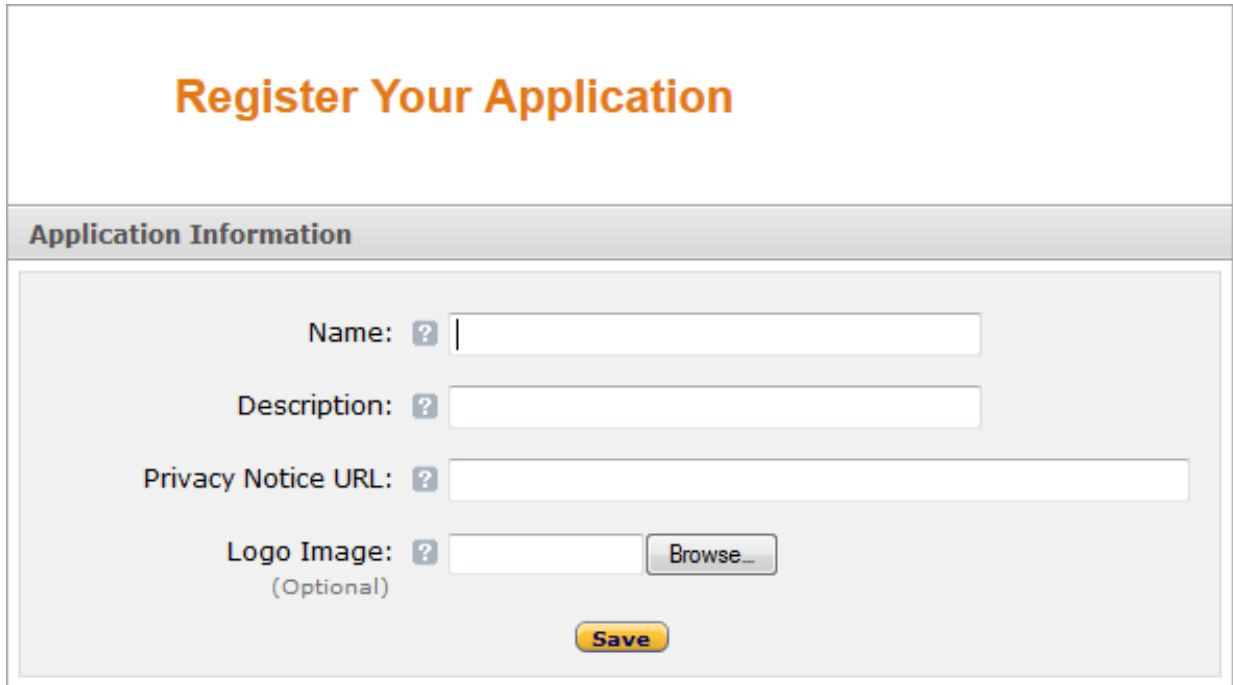
Topics

- [Register Your Login with Amazon Application](#)
- [Adding an iOS App to a Security Profile](#)
- [iOS Bundle ID and API Keys](#)

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 when they first login to one of your apps and are asked to share information with you. Users will see the name of your application, your logo, and a link to your privacy policy. These steps demonstrate how to register a Login with Amazon application and add an iOS app to that account.

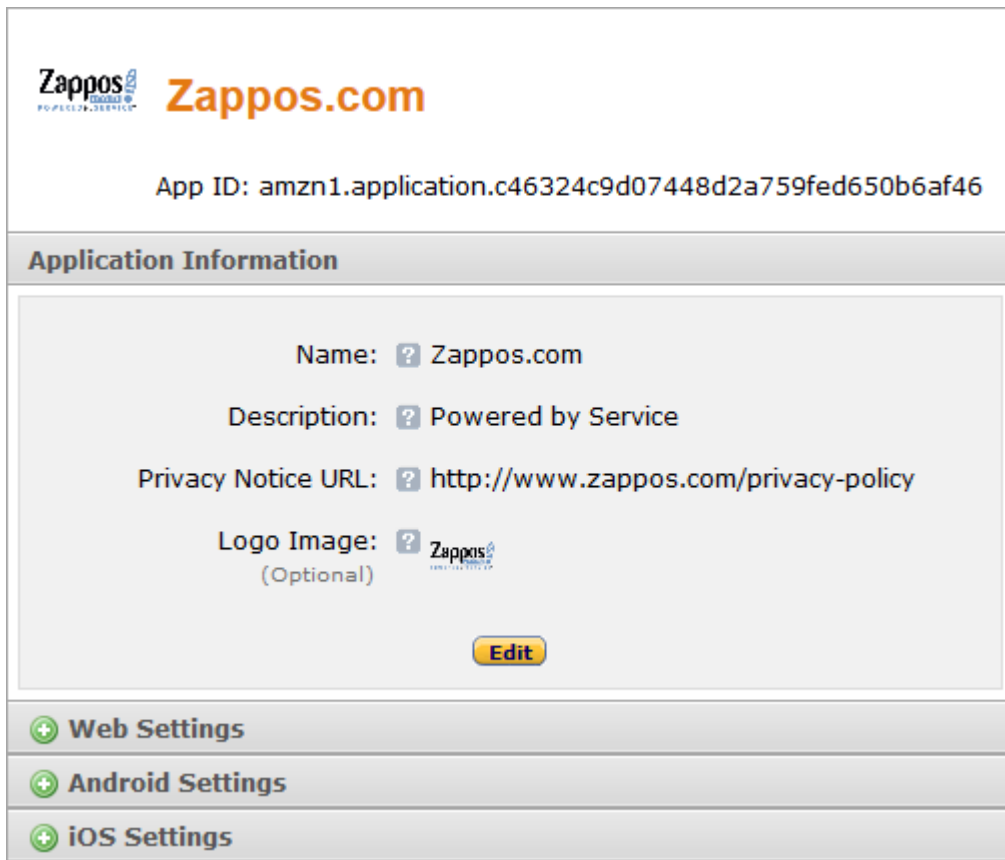
Register Your Login with Amazon Application

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:



The screenshot shows the 'Register Your Application' form. At the top, the title 'Register Your Application' is displayed in orange. Below it is a section header 'Application Information' in a grey bar. The form contains four input fields, each with a question mark icon: 'Name', 'Description', 'Privacy Notice URL', and 'Logo Image'. The 'Logo Image' field is labeled '(Optional)' and includes a 'Browse...' button. A yellow 'Save' button is located at the bottom center of the form.

- 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:



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

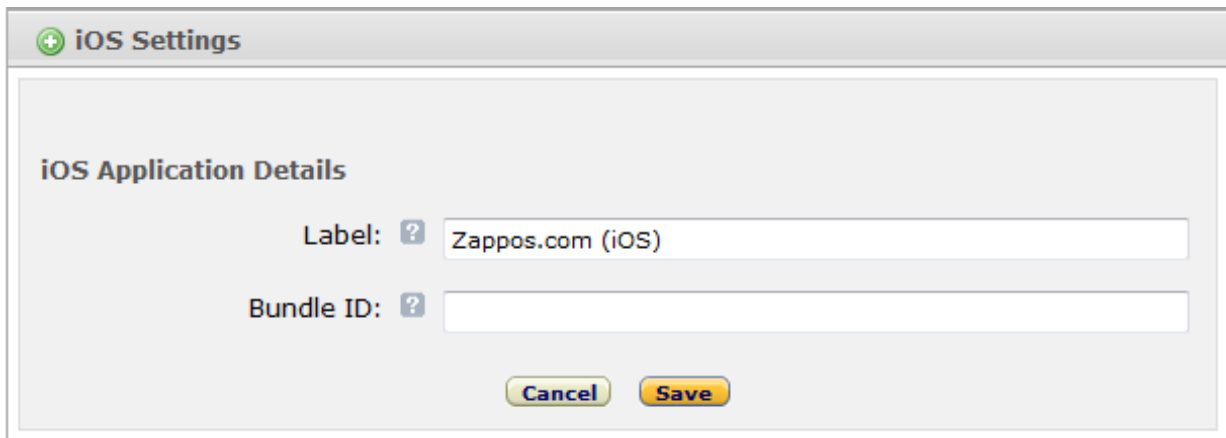
After your basic application settings are saved, you can add settings for specific websites and mobile apps that will use this Login with Amazon account.

Adding an iOS App to a Security Profile

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.

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**.

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.

Determining 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.

Retrieving an iOS API Key

Once 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. Visit <https://login.amazon.com>.
2. Click **App Console**.
3. On the **Apps** box, click on your application.
4. Find your iOS app under the **iOS Settings** section.
If you have not registered an iOS app, see [Adding an iOS App to a Security Profile](#) (p. 7).
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.

6. See [Add Your API Key to Your App Property List](#) (p. 13) for instructions on adding the API key to your iOS app.

Create a Login with Amazon Project

Topics

- [Create a New Login with Amazon Project](#)
- [Install the Login with Amazon Library](#)
- [Add Your API Key to Your App Property List](#)
- [Add a URL Scheme to Your App Property List](#)
- [Add a Login with Amazon Button to Your App](#)

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, you should create one now. If you have an existing app, skip to [Install the Login with Amazon Library](#) (p. 11).

1. Launch **Xcode**.
2. If you are presented with a Welcome to Xcode dialog, select **Create a New Xcode Project**. Otherwise, in 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 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

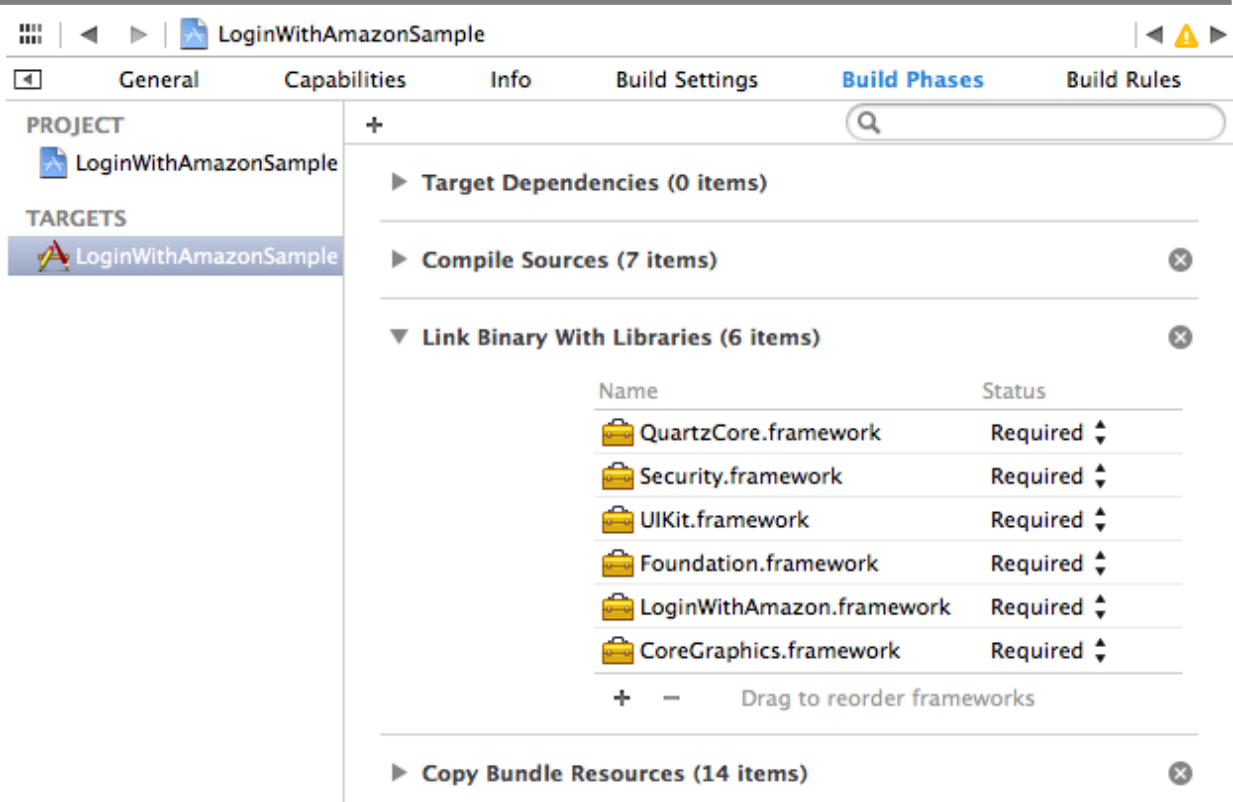
If you have not yet downloaded the Login with Amazon SDK for iOS, see [Install the Login with Amazon SDK for iOS](#) (p. 3).

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. With your project open in Xcode, select the `Frameworks` folder, then click **File** from the main menu and select **Add Files to "project"**.
2. 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. Click **Edit** from the main menu and select **Delete**.
3. Select the name of your project in the **Project Navigator**.
The **Project Editor** will appear in the editor area of the Xcode workspace.
4. Click your project name under **Targets**, and select **Build Phases**. Expand **Link Binary with Libraries** and click the plus sign to add a library.
5. In the search box, type `Security.framework`. Select **Security.framework** and click **Add**.

Login with Amazon: Getting Started Guide for iOS

Install the Login with Amazon Library

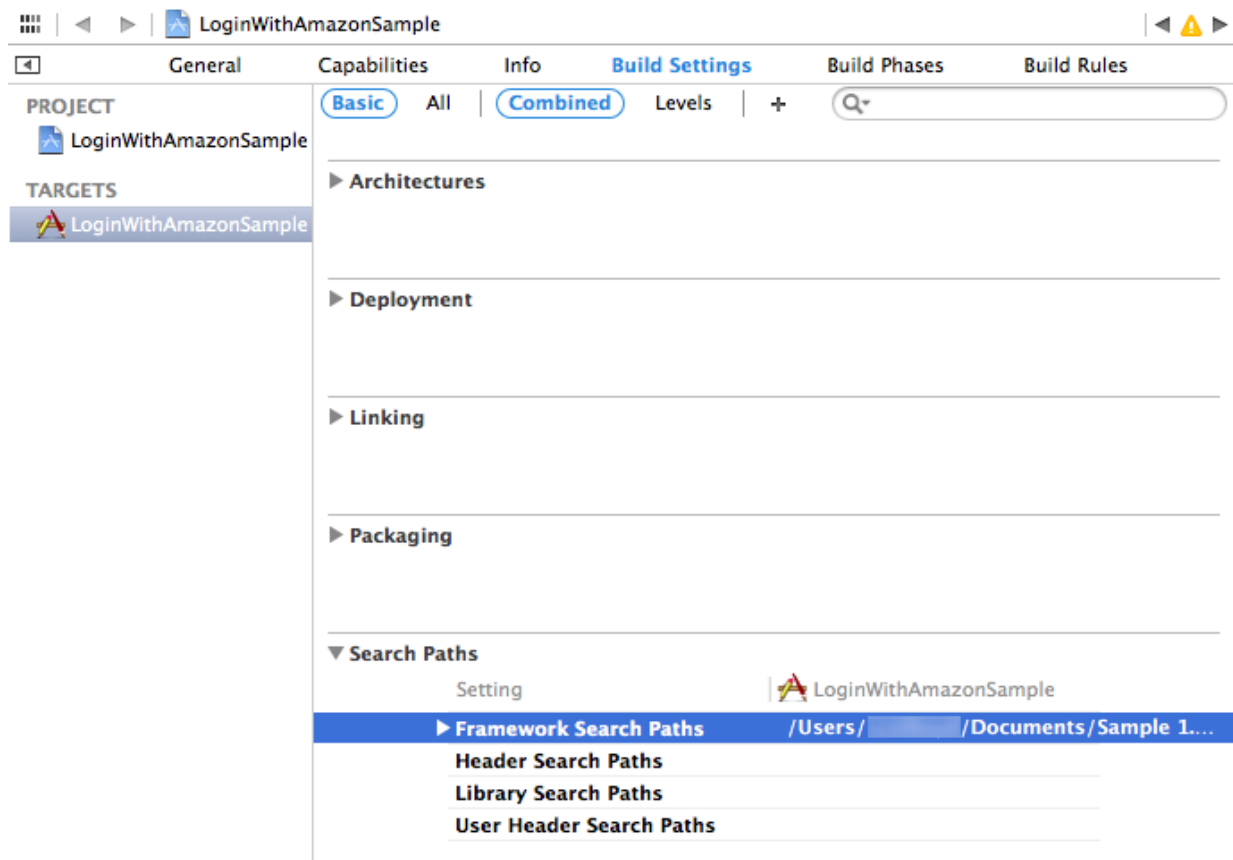


6. Select **Build Settings**. Click **All** to view all settings.
7. Under **Search Paths**, ensure that the `LoginWithAmazon.framework` directory is in the **Framework Search Paths**.

Login with Amazon: Getting Started Guide for iOS

Add Your API Key to Your App Property List

For example:



If you used the Login with Amazon 1.0 library, you can remove any references to the 1.0 library path in the **Header Search Paths** or **Library Search Paths**.

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

Before building your project, if you used the Login with Amazon 1.0 library, replace `#import "AIMobileLib.h"`, `#import "AIAuthenticationDelegate.h"`, or `#import "ALError.h"` in your source files with `#import <LoginWithAmazon/LoginWithAmazon.h>`. `LoginWithAmazon.h` includes all the Login with Amazon headers at once.

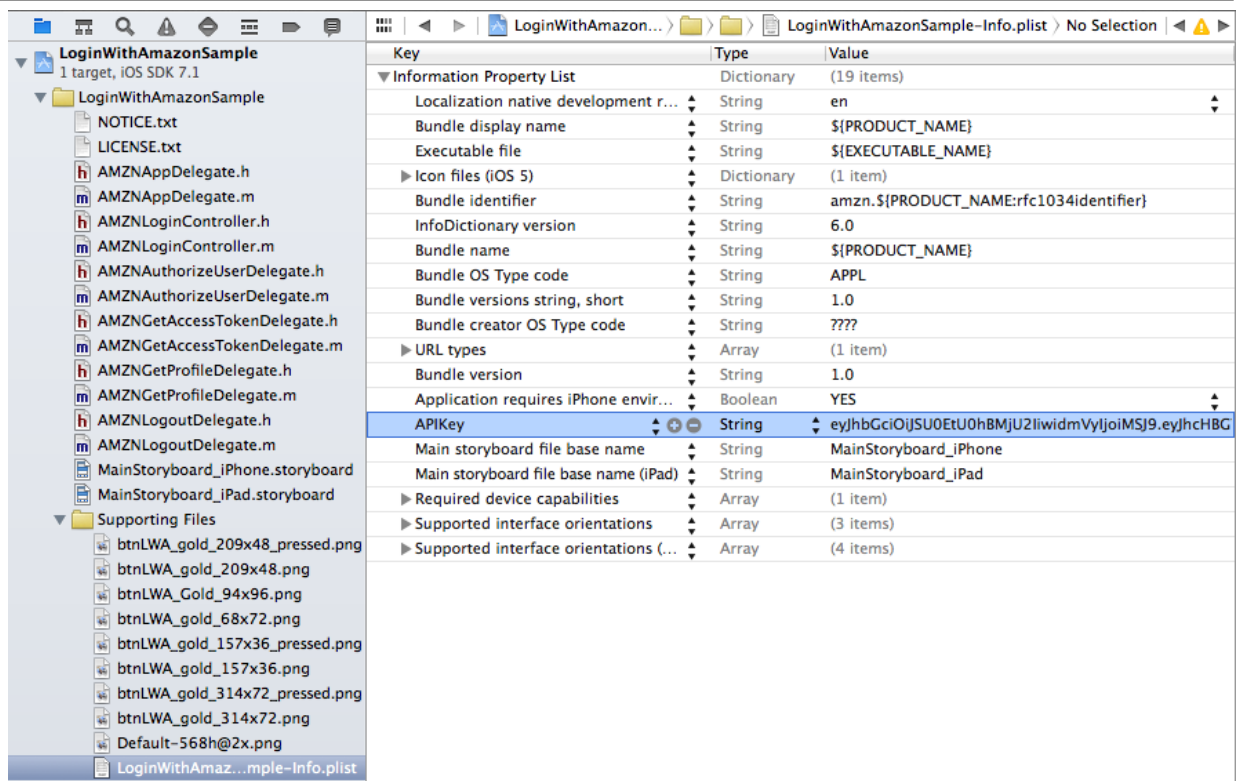
Add Your API Key to Your App Property List

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:

Login with Amazon: Getting Started Guide for iOS

Add a URL Scheme to Your App Property List



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

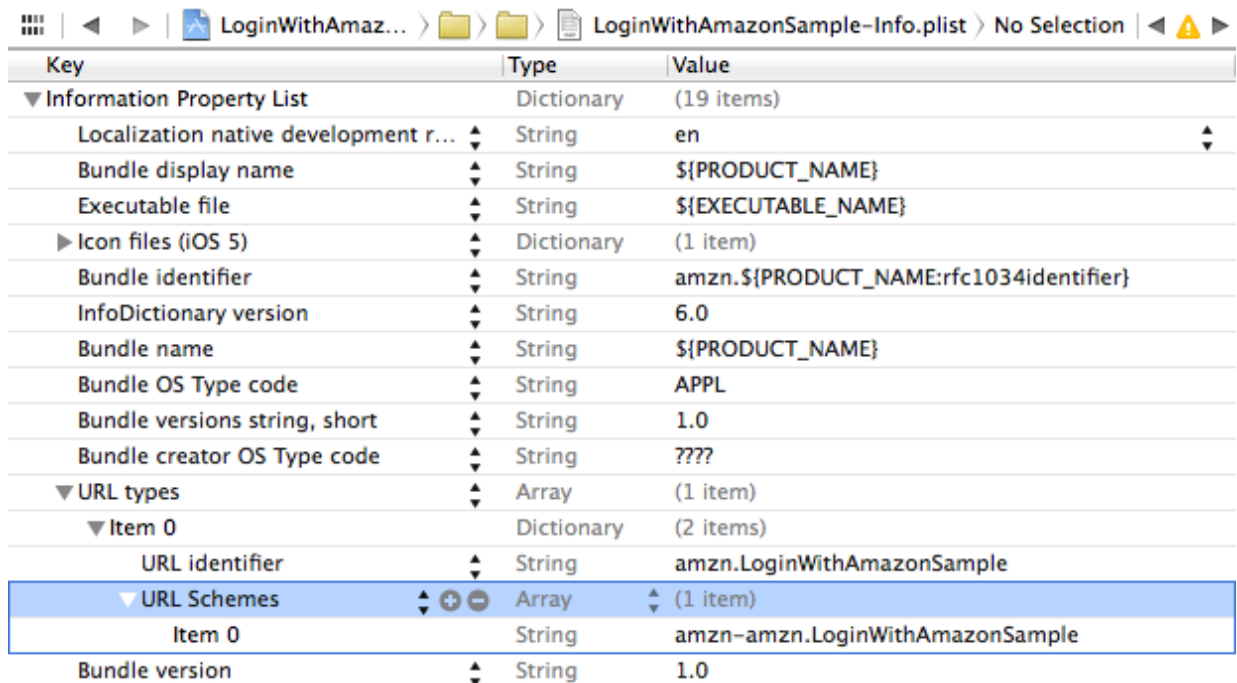
Add a URL Scheme to Your App Property List

When the user logs in, they will be presented with a login page using Safari. In order for your app to receive confirmation of their login, you must add a URL scheme so 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 [Implementing Custom URL Schemes](#) 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:

Login with Amazon: Getting Started Guide for iOS

Add a Login with Amazon Button to Your App



| Key | Type | Value |
|--|------------|--|
| ▼ Information Property List | Dictionary | (19 items) |
| Localization native development region | String | en |
| Bundle display name | String | #{PRODUCT_NAME} |
| Executable file | String | #{EXECUTABLE_NAME} |
| ▶ Icon files (iOS 5) | Dictionary | (1 item) |
| Bundle identifier | String | amzn.#{PRODUCT_NAME:rfc1034identifier} |
| InfoDictionary version | String | 6.0 |
| Bundle name | String | #{PRODUCT_NAME} |
| Bundle OS Type code | String | APPL |
| Bundle versions string, short | String | 1.0 |
| Bundle creator OS Type code | String | ???? |
| ▼ URL types | Array | (1 item) |
| ▼ Item 0 | Dictionary | (2 items) |
| URL identifier | String | amzn.LoginWithAmazonSample |
| ▼ URL Schemes | Array | (1 item) |
| Item 0 | String | amzn-amzn.LoginWithAmazonSample |
| Bundle version | String | 1.0 |

2. Make sure none of the entries are selected. Then, in the main menu, click **Editor**, and **Add Item**. Type 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**. Type 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**. Type 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 a Login with Amazon Button to Your App

Login with Amazon provides several standard buttons you can use to prompt users to login 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 a tutorial on how to add a button to an app, see [Configuring the View in Your First iOS App](#) 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 [Configuring the View in Your First iOS App](#) on [developer.apple.com](#) tutorial includes steps on adding a button event.
3. Choose a button image.
Consult our [Login with Amazon Style Guidelines](#) for a list of buttons 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.

Login with Amazon: Getting Started Guide for iOS

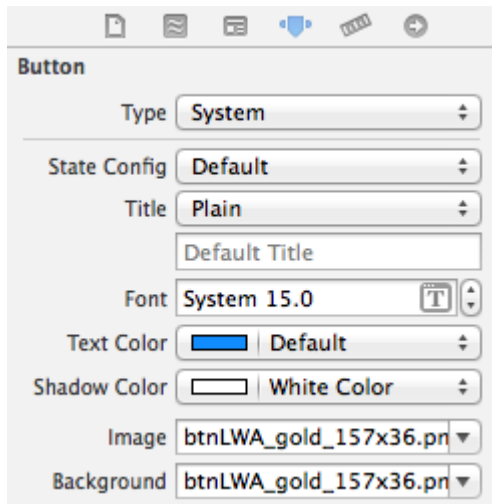
Add a Login with Amazon Button to Your App

- 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) 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 on 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.

Using the SDK for iOS APIs

Topics

- [Handling the Login Button and Getting Profile Data](#)
- [Checking for User Login at Startup](#)
- [Clearing Authorization State and Logging Out a User](#)

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

Handling the Login Button and Getting Profile Data

This section explains how to call the `authorizeUserForScopes:delegate:` and `getProfile:` APIs to login a user and retrieve their profile data. 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](#) (p. 11).
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. Create the `AMZNAuthorizeUserDelegate` class to implement `AIAuthenticationDelegate`.

When `authorizeUserForScopes:delegate:` completes, it will call the `requestDidSucceed:` or `requestDidFail:` method on an object that implements the `AIAuthenticationDelegate` protocol.

```
@interface AMZNAuthorizeUserDelegate : NSObject<AIAuthenticationDelegate>
@end
```

For more information, see [Working with Protocols](#) on developer.apple.com.

4. Call `authorizeUserForScopes:delegate:` in `onLoginButtonClicked`.

If you followed the steps in [Add a Login with Amazon Button to Your App](#) (p. 15), you should have an `onLoginButtonClicked:` method linked to a Login with Amazon button. In that method, call `authorizeUserForScopes:delegate:` to prompt the user to login and authorize your application.

When your application is authorized, it is authorized for one or more data sets known as *scopes*. The first parameter is an array of scopes that encompass 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 three scopes: `profile`, which contains the user's name, email address, and Amazon account id; `profile:user_id`, which contains only the Amazon account id; and `postal_code`, which contains the user's zip/postal code.

The second parameter to `authorizeUserForScopes:delegate:` is an object that implements the `AIAuthenticationDelegate` protocol, in this case an instance of the `AMZNAuthorizeUserDelegate` class.

```
- (IBAction)onLogInButtonClicked:(id)sender {
    // Make authorize call to SDK to get a secure access token
    // for the user.
    // While making the first call you can specify the minimum basic
    // scopes needed.

    // Requesting both scopes for the current user.
    NSArray *requestScopes =
        [NSArray arrayWithObjects:@"profile", @"postal_code", nil];

    AMZNAuthorizeUserDelegate* delegate =
        [[AMZNAuthorizeUserDelegate alloc] initWithParentController:self];

    [AIMobileLib authorizeUserForScopes:requestScopes delegate:delegate];
}
```

Add your delegate implementation header to the class calling `authorizeUserForScopes:`. For example:

```
#import "AMZNAuthorizeUserDelegate.h"
```

5. Create an `AMZNGetProfileDelegate`.

`AMZNGetProfileDelegate` is our name for a class that implements the `AIAuthenticationDelegate` protocol, and will process the result of the `getProfile:` call. Like `authorizeUserForScopes:delegate:`, `getProfile:` supports the `requestDidSucceed:` and `requestDidFail:` protocol methods. `requestDidSucceed:` receives an `APIResult` object with profile data in the `result` property. `requestDidFail:` receives an `AIError` object with information on the error in the `error` property.

To create a delegate class from a normal class declaration, import `AIAuthenticationDelegate.h` and add the protocol to the declaration in your class header file:

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

@interface AMZNGetProfileDelegate : NSObject<AIAuthenticationDelegate>

@end
```

6. Implement `requestDidSucceed:` for your `AMZNAuthorizeUserDelegate`.

In `requestDidSucceed:`, call `getProfile:` to retrieve the customer profile. `getProfile:`, like `authorizeUserForScopes:delegate:`, uses the `AIAuthenticationDelegate` protocol.

```
- (void)requestDidSucceed:(APIResult *)apiResult {
    // Your code after the user authorizes application for
    // requested scopes.

    // Load new view controller with user identifying information
    // as the user is now successfully logged in.

    AMZNGetProfileDelegate* delegate =
        [[[AMZNGetProfileDelegate alloc]
         initWithParentController:parentViewController] autorelease];
    [AIMobileLib getProfile:delegate];
}
```

Add your delegate implementation header to the class calling `getProfile:`. For example:

```
#import "AMZNGetProfileDelegate.h"
```

7. Implement `requestDidSucceed:` for your `AMZNGetProfileDelegate`.

`requestDidSucceed:` has two main tasks; retrieve the profile data from the `APIResult`, and pass the data to the UI.

To retrieve the profile data from the `APIResult`, access the `result` property. For a `getProfile:` response, that property will contain a dictionary of property values for the user profile properties. The profile properties are `name`, `email`, and `user_id` for the profile scope and `postal_code` for the `postal_code` scope.

```
- (void)requestDidSucceed:(APIResult *)apiResult {
    // Get profile request succeeded. Unpack the profile information
    // and pass it to the parent view controller

    NSString* name = [(NSDictionary*)apiResult.result
        objectForKey:@"name"];
    NSString* email = [(NSDictionary*)apiResult.result
        objectForKey:@"email"];
    NSString* user_id = [(NSDictionary*)apiResult.result
        objectForKey:@"user_id"];
    NSString* postal_code = [(NSDictionary*)apiResult.result
        objectForKey:@"postal_code"];

    // Pass data to view controller
}
```

8. Implement requestDidFail: for your AMZNGetProfileDelegate.

requestDidFail: includes an **APIError** object containing details about the error. **showLoginPage** is a hypothetical method that would reset the main view controller to show the Login with Amazon button.

```
- (void)requestDidFail:(APIError *)errorResponse {
    // Get Profile request failed for profile scope.

    // If error code = kIAApplicationNotAuthorized,
    // allow user to log in again.
    if(errorResponse.error.code == kIAApplicationNotAuthorized) {
        // Show authorize user button.
        [parentViewController showLoginPage];
    }
    else {
        // Handle other errors
        [[[UIAlertView alloc] initWithTitle:@" " message:[NSString
            stringWithFormat:@"Error occurred with message: %@",
            errorResponse.error.message] delegate:nil
            cancelButtonTitle:@"OK"otherButtonTitles:nil] autorelease]
            show];
    }
}
```

9. Implement requestDidFail: for your AMZNAuthorizeUserDelegate.

```
- (void)requestDidFail:(APIError *)errorResponse {
    NSString *message = errorResponse.error.message;
    // Your code when the authorization fails.

    [[[UIAlertView alloc] initWithTitle:@" " message:[NSString
        stringWithFormat:@"User authorization failed with message: %@",
        errorResponse.error.message] delegate:nil
        cancelButtonTitle:@"OK"otherButtonTitles:nil] autorelease] show];
}
```

10. Implement application:openURL:sourceApplication:annotation: in the class in your project that handles the UIApplicationDelegate protocol. (By default this will be the AppDelegate class in your project). When the app presents the Amazon login page using the Safari browser, if the user completes logs in the browser will redirect to the app using the URL Scheme the app registered earlier. That redirect is passed to **application:openURL:sourceApplication:annotation:..**

application:openURL:sourceApplication:annotation: returns **YES** if the URL was successfully handled. **handleOpenURL:sourceApplication:** is an SDK library function that will

handle Login with Amazon redirect URLs for you. If `handleOpenURL:sourceApplication:` returns YES, then the URL was handled.

```
- (BOOL)application:(UIApplication *)application
    openURL:(NSURL *)url
    sourceApplication:(NSString *)sourceApplication
    annotation:(id)annotation
{
    // Pass on the url to the SDK to parse authorization code
    // from the url.
    BOOL isValidRedirectSignInURL =
        [AIMobileLib handleOpenURL:url sourceApplication:sourceApplication];

    if(!isValidRedirectSignInURL)
        return NO;

    // App may also want to handle url
    return YES;
}
```

For more information on `application:openURL:sourceApplication:annotation:`, see [UIApplicationDelegate Protocol Reference](#) on developer.apple.com.

Checking 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 `getAccessTokenForScopes:withOverrideParams:delegate:` to see if the app is still authorized.

1. Create an `AMZNGetAccessTokenDelegate` class.

`AMZNGetAccessTokenDelegate` implements the `AIAuthenticationDelegate` protocol, and will process the result of the `getAccessTokenForScopes:withOverrideParams:delegate:` call. `AIAuthenticationDelegate` contains two methods, `requestDidSucceed:` and `requestDidFail:`. `requestDidSucceed:` receives an `APIResult` object with token data, while `requestDidFail:` receives an `APIError` object with information on the error.

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

@interface AMZNGetAccessTokenDelegate : NSObject<AIAuthenticationDelegate>

@end
```

Add your delegate implementation header to the class calling `getAccessTokenForScopes:withOverrideParams:delegate:`. For example:

```
#import "AMZNGetAccessTokenDelegate.h"
```

2. On app startup, call `getAccessTokenForScopes:withOverrideParams:delegate:` to see if the application is still authorized.

`getAccessTokenForScopes:withOverrideParams:delegate:` retrieves the raw access token that Login with Amazon uses to access a customer profile. If the method succeeds, the app is still authorized and a call to `getProfile:` should succeed.

`getAccessTokenForScopes:withOverrideParams:delegate:` uses the `AIAuthenticationDelegate` protocol in the same manner as

Login with Amazon: Getting Started Guide for iOS

Checking for User Login at Startup

`authorizeUserForScopes:delegate:..` Pass the object implementing the protocol as the `delegate` parameter.

```
- (void)checkIsUserSignedIn {
    AMZNGetAccessTokenDelegate* delegate =
        [[[AMZNGetAccessTokenDelegate alloc]
         initWithParentController:self] autorelease];
    NSArray *requestScopes =
        [NSArray arrayWithObjects:@"profile", @"postal_code", nil];
    [AIMobileLib getAccessTokenForScopes:requestScopes
     withOverrideParams:nil delegate:delegate];
}
```

3. Implement `requestDidSucceed:` on your `AMZNGetAccessTokenDelegate`.

`requestDidSucceed:` has one task: to call `getProfile:`. This example calls `getProfile:` using the same listener you declared in the previous section (see steps 6-8).

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

- (void)requestDidSucceed:(APIResult *)apiResult {
    // Your code to use access token goes here.

    // Since the application has authorization for our scopes, we can
    // get the user profile.
    AMZNGetProfileDelegate* delegate = [[[AMZNGetProfileDelegate alloc]
     initWithParentController:parentViewController] autorelease];

    [AIMobileLib getProfile:delegate];
}
```

4. Implement `requestDidFail:` on your `AMZNGetAccessTokenDelegate`.

`requestDidFail:` includes an `APIError` object containing details about the error. If you receive an error, you can reset the main view controller to show the Login with Amazon button.

```
- (void)requestDidFail:(APIError *)errorResponse {
    // Your code to handle failed retrieval of access token.

    // If error code = kIAApplicationNotAuthorized, allow user
    // to log in again.
    if(errorResponse.error.code == kIAApplicationNotAuthorized) {
        // Show Login with Amazon button.
    }
    else {
        // Handle other errors
        [[[[UIAlertView alloc] initWithTitle:@"" message:[NSString
         stringWithFormat:@"Error occured with message: %@",
         errorResponse.error.message] delegate:nil
         cancelButtonTitle:@"OK" otherButtonTitles:nil]
         autorelease] show];
    }
}
```

Clearing Authorization State and Logging Out a User

The `clearAuthorizationState:` method will clear the user's authorization data from the `AIMobileLib` local data store. A user will have to log in 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. Declare an `AMZNLogoutDelegate`. This is a class that implements the `AIAuthenticationDelegate` protocol. For our purposes, we can inherit the class from `NSObject`:

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

@interface AMZNLogoutDelegate : NSObject<AIAuthenticationDelegate>

@end
```

Add your delegate implementation header to the class calling `clearAuthorizationState:`. For example:

```
#import "AMZNLogoutDelegate.h"
```

2. Call `clearAuthorizationState:`.

Once a user has successfully logged in, you may provide a logout mechanism so they can clear their authorization data. Your mechanism might be a hyperlink, or a menu item, but for this scenario the example will create a `logoutButtonClicked` method for a logout button.

```
- (IBAction)logoutButtonClicked:(id)sender {
    AMZNLogoutDelegate* delegate = [[[AMZNLogoutDelegate alloc]
        initWithParentController:self] autorelease];

    [AIMobileLib clearAuthorizationState:delegate];
}
```

The only parameter to `clearAuthorizationState` is a an `AIAuthenticationDelegate` that implements `requestDidSucceed:` and `requestDidFail:`.

3. Implement `requestDidSucceed:`. This method will be called when the user's information is cleared. You should then show them as logged out.

```
- (void)requestDidSucceed:(APIResult *)apiResult {
    // Your additional logic after the user authorization
    // state is cleared.
    [[[UIAlertView alloc] initWithTitle:@"" message:@"User Logged out."
        delegate:nil cancelButtonTitle:@"OK" otherButtonTitles:nil]
        show];
}
```

4. Implement `requestDidFail:`. This method will be called if for some reason the user's information cannot be cleared from the cache. In that case, you should not show them as logged out.

```
- (void)requestDidFail:(APIError *)errorResponse {
    // Your additional logic after the SDK failed to clear
    // the authorization state.

    [[[UIAlertView alloc] initWithTitle:@"" message:[NSString
stringWithFormat:@"User Logout failed with message: %@",
errorResponse.error.message] delegate:nil
cancelButtonTitle:@"OK" otherButtonTitles:nil]
autorelease] show];
}
```