# Developer Guide for UID2 Mobile SDKs

*SDK version:*

*iOS V 0.2.0*

*Android V 0.4.0*

## Features

The UID2 Mobile SDKs are designed to manage UID2 Identity on behalf of iOS and Android apps. It enables UID2 Identity to be persisted across app lifecycles by securely storing it on a device via platform-native encryption tools.

By default, the SDK automatically refreshes UID2 Identity based on expiration dates. However, you can disable this to allow implementing apps to manage the UID2 Identity lifecycle manually.

## Open Source

The following open source GitHub repositories are available for these SDKs:

* <https://github.com/IABTechLab/uid2-ios-sdk>
* <https://github.com/IABTechLab/uid2-android-sdk>

## Installation

Follow the applicable installation instructions:

* iOS
* Android

### IOS

Install the iOS SDK via Swift Package Manager (SPM). There are two installation options:

* Package.swift
* Xcode

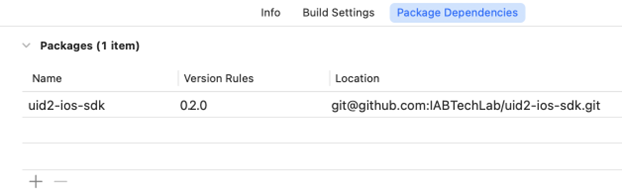
#### Package.swift

Add the following dependency to Package.swift:

dependencies: [   
.package(url: "https://github.com/IABTechLab/uid2-ios-sdk.git", from: "0.2.0"),   
],

#### Xcode

Add the following dependency via Xcode:



### Android

There are two options for installing the Android UID2 SDK:

* Gradle
* Maven

#### Gradle

To install with Gradle, add the SDK as a dependency in the build.gradle file:

implementation 'com.uid2.uid2-android-sdk:0.4.0'

#### Maven

To install with Maven, , add the SDK as a dependency in the pom.xml file:

<dependency>

<groupId>com.uid2</groupId>

<artifactId>uid2-android-sdk</artifactId>

<version>0.4.0</version>

</dependency>

## How to Use the SDK

The initial UID2 Identity *must* be generated by the implementing application and then passed into the UID2 SDK. Because of security requirements, the UID2 Mobile SDKs cannot create new UID2 identities.

The UID2 Mobile SDKs can perform refreshes of UID2 identities, after an Identity is established. This is because the refresh functionality relies on the refresh tokens that are part of the UID2 Identity.

The **UID2Manager** singleton is the primary developer API for the UID2 SDK. It is responsible for storing, refreshing, and retrieving UID2 Identity.

The behavior is a little different depending on the SDK that’s in use:

* iOS: The UID2Manager is initialized automatically the first time it is accessed and can be configured to support automatic or manual refresh capabilities.
* Android: The UID2Manager must be initialized manually before it can be used (see Android Initialization content below).

### UID2Manager API

This section includes the functions and variables that are part of the UID2Manager API.

#### Functions

The following functions are available as part of the UID2Manager API.

##### setIdentity()

Sets the UID2 Identity to be managed by the SDK.

##### resetIdentity()

Resets or removes the UID2 Identity currently being managed by the SDK.

##### refreshIdentity()

Manually refreshes the UID2 Identity being managed by the SDK.

##### getAdvertisingToken()

If the current UID2 Identity is valid, this function returns the advertising token.

##### setAutomaticRefreshEnabled()

Toggle for automatic refresh functionality.

#### Variables

The following variables are available as part of the UID2Manager API.

##### identity

The Identity variable stores and returns the current UID2Identity data object being managed by the SDK.

##### identityStatus

The identityStatus variable stores and returns the status of the current UID2 Identity being managed by the SDK

### Android Initialization

The Android implementation expects the singleton to be initialized before use. This does two things:

* It allows for easier access later.
* It allows the consuming application to potentially provide its own network instance, responsible for making requests.

The initialization can be done during the creation of the APPLICATION instance, as shown in the following example:

class MyApplication : Application() {

override fun onCreate() {

super.onCreate()

// Initialize the UID2Manager class. Use DefaultNetworkSession rather than providing our own

// custom implementation. This can be done to allow wrapping something like OkHttp.

UID2Manager.init(this.applicationContext)

## Code Samples

The following code samples provide examples of performing specific activities relating to managing UID2.

Set the initial UID2 Identity (iOS):

UID2Manager.shared.setIdentity(\_ identity: UID2Identity)

Set the Initial UID2 Identity (Android):

UID2Manager.getInstance().setIdentity(identity: UID2Identity)

Get the Advertising Token to pass to the Advertising SDK (iOS):

UID2Manager.shared.getAdvertisingToken()

Get the Advertising Token to pass to the Advertising SDK (Android):

UID2Manager.getInstance().getAdvertisingToken()

-⧫-