Oplytic Attribution

V 2.3
January 9, 2019

Oplytic provides attribution for app-to-app and mobile-web-to-app mobile marketing. Oplytic leverages the tracking provided by Universal Links (iOS) and App Links (Android) to deliver the precise attribution and re-attribution needed for mobile marketers.

Two-Step Set-Up

Step 1: Set-Up with Oplytic

Use the API or contact customer service to provide us with the following:

General	iOS	Android
• App Name (friendly one-word name)	• Bundle ID • Team ID • iTunes App-Store Link	• Package Name • SHA256 Cert Fingerprints

After Oplytic receives the above information we will send you your app tracking link. For example, with an App Name of "yourapp", your tracking link would be:

- https://yourapp.oplct.com/

Append any parameters you want to be stored for attribution. If you are tracking campaign and affiliate, for example, your link might look like:

- https://yourapp.oplct.com/?cid={Campaign ID}&affid={Affiliate ID}

Step 2: Add Oplytic to Your iOS and Android Apps

1) Enable App Links: In XCode or Android Studio.

2) Include the Oplytic Library: Add the project / source files, or reference the binaries (SDK).

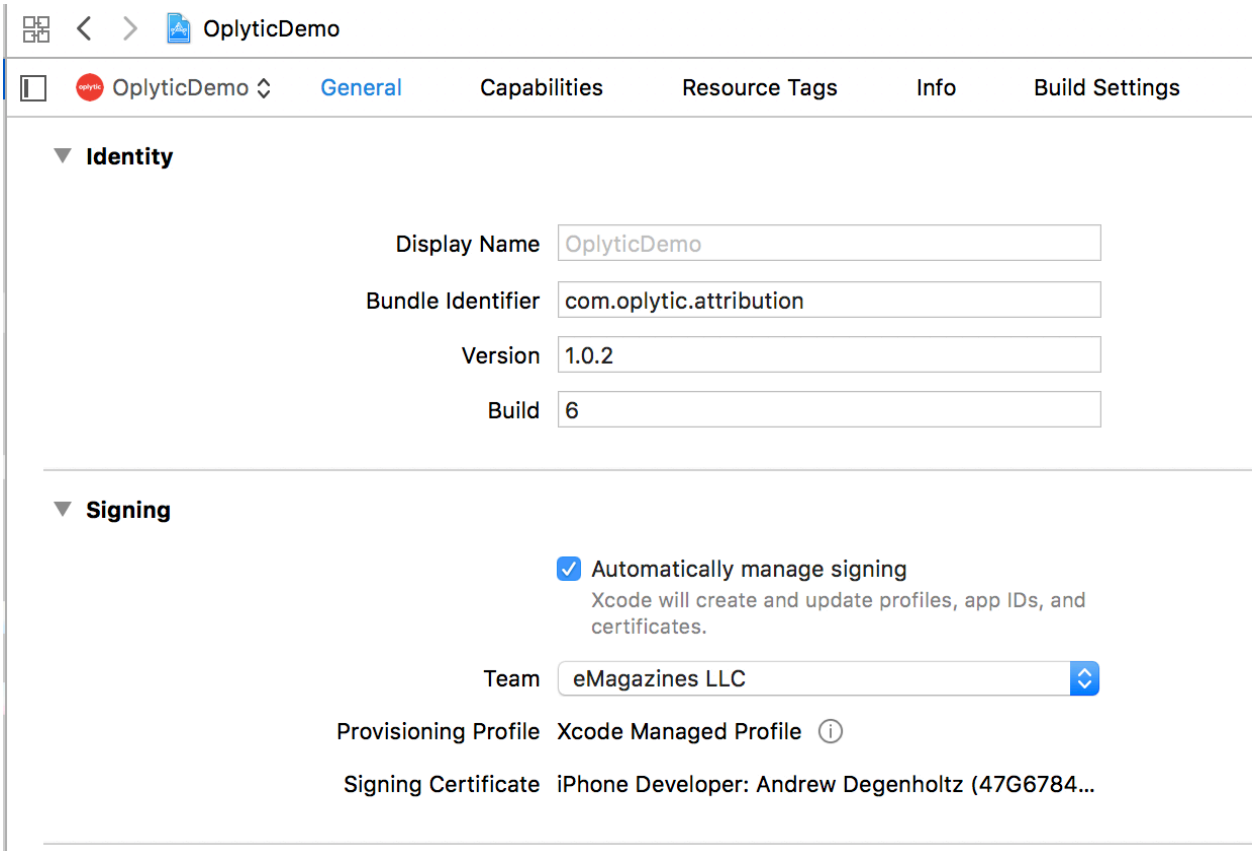
3) Use the Oplytic Library: Attribute app activity by adding just a few lines of code to your app.

iOS Walk-Through

Set-Up with Oplytic

Register your app with the Oplytic API or customer service. Simply provide a friendly one-word app name, the Bundle ID and the Team ID.

Grab the Bundle ID and Team ID from the XCode development environment. Look for these settings in the “General” tab, “Identity” and “Signing” sections.



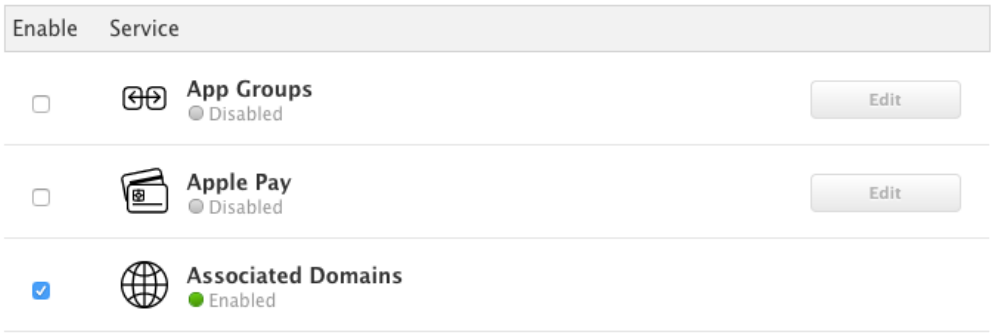
https://developer.apple.com/library/content/documentation/IDEs/Conceptual/AppDistributionGuide/ConfiguringYourApp/ConfiguringYourApp.html

Enable App Links

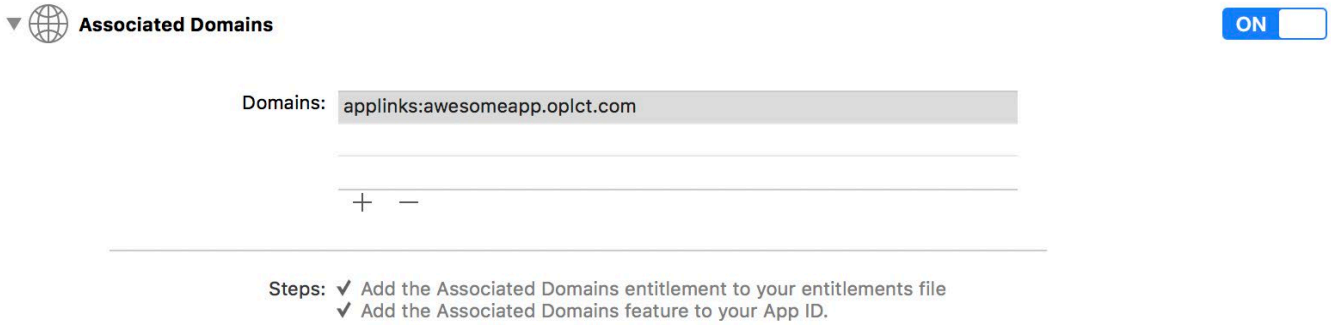
App links enable your app to be launched directly from clicks on safari and other apps. Follow the standard Apple Universal Link scenario:

https://developer.apple.com/ios/universal-links/

In your Apple iTunes developer account, under App IDs, select your application and make sure Associated Domains are enabled.



In XCode, select the project target, then click on the "Capabilities" tab. Scroll down to the "Associated Domains" option. Click on the button to turn it On, and then click on the "+" button to add the following item:



Make sure you specify the App-Name you provided to Oplytic instead of "awesomeapp" above.

NOTE: If you want to access the deep-link-path and URL data yourself, you can access it via the userActivity.webPageUrl attribute.

NOTE: Due to an issue with iOS browser security you cannot enter or copy/paste the above links directly into the Safari URL bar. However, you can embed the link in apps, web-pages, emails, or other forms of social media.

Include the Oplytic Library

Include the Oplytic SDK Cocoa-Pod found here:

https://cocoapods.org/pods/OplyticSDK
https://github.com/oplytic/OplyticSDK

Use the Oplytic Library

Be sure to include Oplytic in any files that use the library:

```
Import OplyticSDK
```

Start the Oplytic SDK

Initialize the Oplytic SDK in your app delegate class. When the app is launched, start the SDK, passing in the API-Key provided by Oplytic. Handle app-link events as well.

Add the following code to your AppDelegate class:

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
        Oplytic.start(apiKey: [your_api_key])
        return true
    }

func application(_ application: UIApplication,
                 continue userActivity: NSUserActivity,
                 restorationHandler: @escaping ([Any]?) -> Void) -> Bool {
      Oplytic.handleUniversalLink(userActivity: userActivity)
      return true
}

func application(_ application: UIApplication,
                 willContinueUserActivityWithType userActivityType: String) -> Bool
{
     return userActivityType == NSUserActivityTypeBrowsingWeb
}
```

1) Track App-Events

There are just two methods for adding events. Make these calls whenever you want to track a purchase, registration, or other important app event.

The Oplytic SDK also tracks app installs and attribution events automatically. Each time the app is reached via an app-link, the SDK will register a new "last-click" attribution and all subsequent events and purchases will be credited to that link.

AddEvent is a general-purpose method:

```
public func addEvent(eventAction: String? = nil,
                     eventObject: String? = nil,
                     eventId : String? = nil,
                     str1: String? = nil, str2: String? = nil, str3: String? = nil,
                     num1 : Double? = nil, num2: Double? = nil)
```

1) *eventAction*: a string associated with the event action, for example, "view" or "shop."
2) *eventObject*: a string associated with the target of the event action, for example "map" or an object SKU.
3) *eventId*: a unique string that you can pass along to associate with the event.
4) *str1, str2, str3*: arbitrary strings associated with the event. You can use these to pass any sort of associated data for that event.
5) *num1, num2*: arbitrary Double numeric values associated with the event. You can use these to pass any sort of associated data for that event.

AddPurchaseEvent is used specifically to track in-app purchases:

```
public func addPurchaseEvent(item : String,
                             item_id: String,
                             quantity: Double,
                             price: Double,
                             currency_unit : String)
```

1) *item*: Name of item being purchased.
2) *Item_id*: SKU or other ID associated with the item being purchased
3) *quantity*: Quantity of items being purchased.
4) *price*: Price of item being purchased.
5) *currency_unit*: String value representing the currency, for example: "USD"

Handle Click Attribution Data (optional)

If your app needs to know about the attributed click, assign an OplyticAttributionHandler protocol, like the simple ViewController example does below:

```
class ViewController: UIViewController, OplyticAttributionHandler {

    override func viewDidLoad() {
        Oplytic.OplyticAttributionHandler = self
        super.viewDidLoad()
    }

    func onAttribution(data: [String:String]){
        //handle Attributed click query params
    }
}
```

Android Walk-Through

Setup with Oplytic

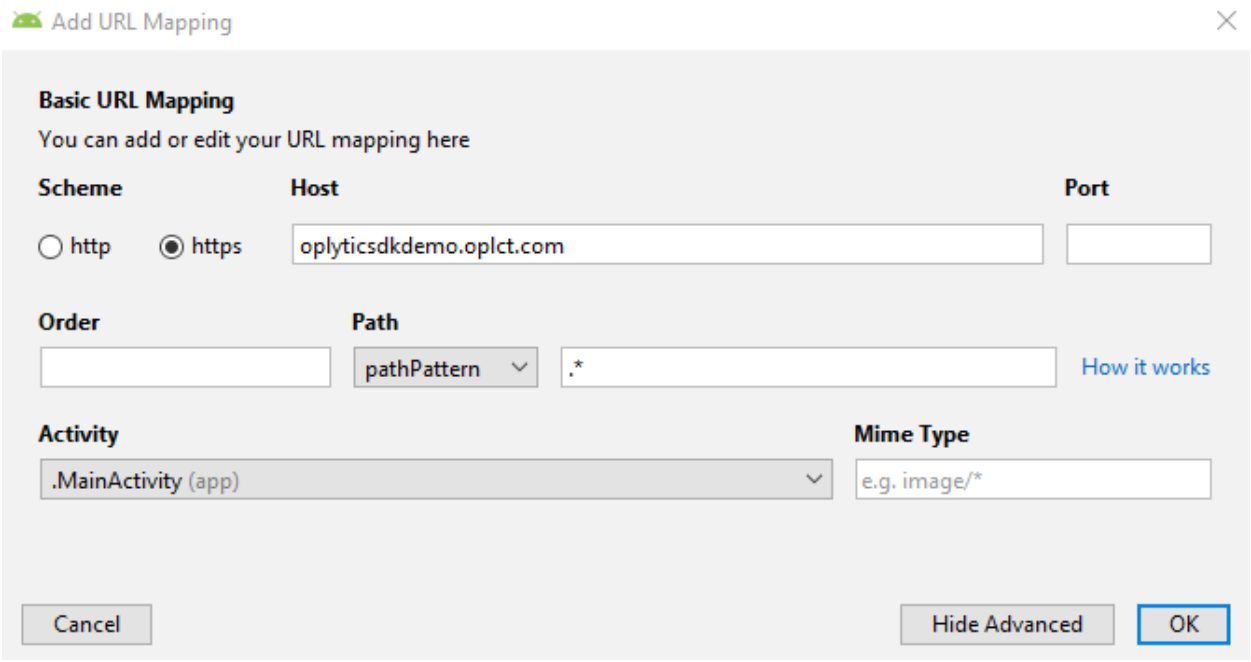
NOTE: This guide uses ***Google's Android Studio IDE***.

After receiving your Oplytic link, launch the app-linking wizard via menu *Tools > App Links Assistant*.

Click on the "Open URL Mapping Editor" button.

Click + > Show Advanced

Create deep link paths for your links to route to. In the example below, we are routing all links to the Main Activity and parsing the URL for custom deep linking there.



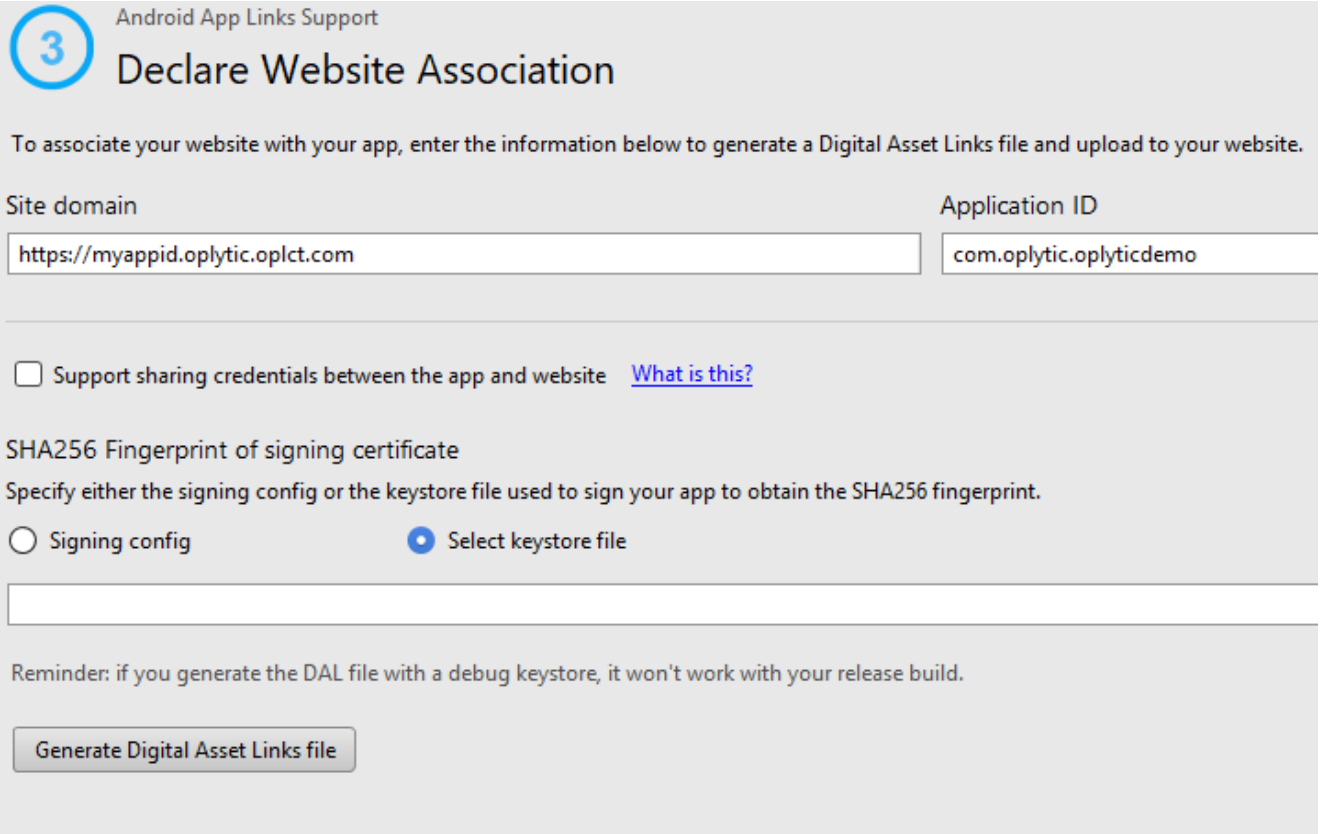
This will associate the URL with the app and activity. Once you click on "OK" you will see the selected path in the URL mapping screen.

This will turn your Deep Link into an Android App Link which is a special type of deep link that allows your URLs to immediately open the corresponding content in your Android app without requiring the user to select the app through a disambiguation dialog box for phones with an API > 22.

To do further parsing with the URI use the code below. This can help to direct the user the proper item or location in the activity.

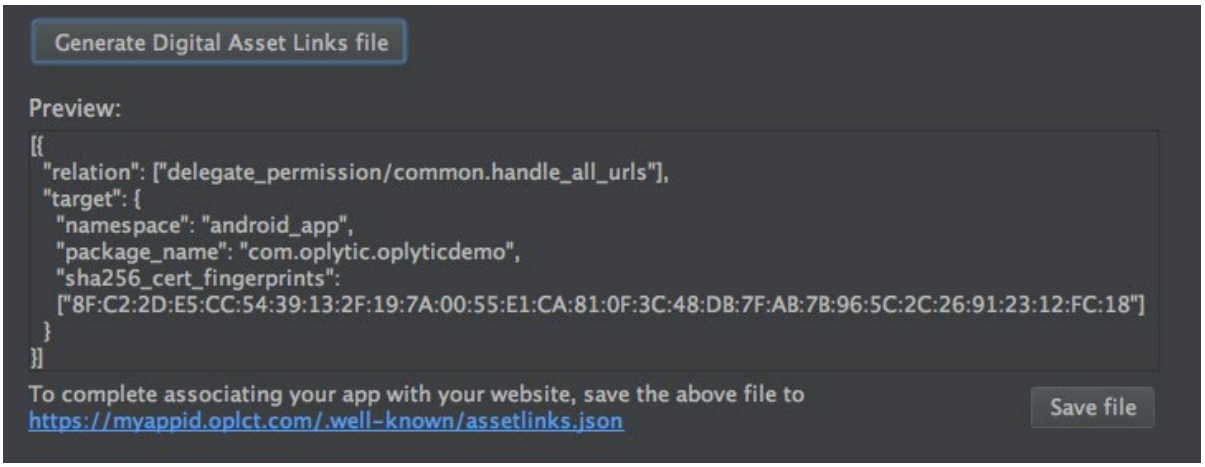
```
Uri uri = getIntent().getData();
```

Next, scroll down to the “Associate Website” section and tap on the “Open Digital Asset Links File Generator” button. In the Site Domain enter the full URL that links to your app. Enter the Application ID for the app if it isn’t already specified.



Click on the “Generate Digital Asset Links File” button with your production keystore. This will generate SHA256_cert_fingerprints string to be sent to Oplytic. If you are using Google’s app signing you will need to provide that app fingerprint instead which is located in the Google Play Console.

Google Play Console > Select App > Release Management > App Signing



To receive App Install events add this code to your BroadcastReceiver. This will allow the SDK to have access to the installation intent for attribution.

```
OplyticSDK.onReceiveInstall(intent);
```

If your app does not use a BroadcastReceiver please include Oplytic's so the SDK can receive the install broadcast. This can also be used if your broadcast receiver rebroadcasts the intent to be sent to Oplytic or optionally call the code below to pass in the install intent;

```
AppInstallListener.handleInstallIntent(intent);
```

```
<!-- Handling install for attribution -->
        <receiver
            android:name="com.oplytic.oplyticsdk.AppInstallListener"
            android:exported="true"
            android:enabled="true">
            <intent-filter>
                <action android:name="com.android.vending.INSTALL_REFERRER" />
            </intent-filter>
        </receiver>
```

Include the Oplytic Library

To access the Oplytic SDK import the whole project, source files or SDK jar into your application project and add it to your module build gradle settings:

```
dependencies {
  …
   compile project(":oplyticsdk")
  …
}
```

Be sure to import the Oplytic package in all your app's class files that use it, such as Broadcast Receiver, application class and anywhere you push events.

import com.oplytic.oplyticsdk.OplyticSDK;

Initialize the SDK

Start Signature:

```
public static void start(Application application, String advertisingId, String apiKey,
IAttributionHandler attributeHandler) {
```

In Your Application Class:

```
public class DemoApplication extends Application implements IAttributionHandler {

    static final String TAG = "OplyticSDKDemo";

    @Override
    public void onCreate(){
        Log.d("OplyticSDK", "In Application class");
        super.onCreate();
        OplyticSDK.start(this, null, "C802A387-43E8-4CCE-B9D2-6EF8E1901D9C", null);
    }
```

Or Oplytic's Application Class in the android manifest (If you are not using your own)

```
<application
    android:name="com.oplytic.oplyticsdk.OplyticApplication"
…
</application>
```

NOTE: When creating the OplyticSDK you may pass in an advertisingId (optionally can be null): to be used as a unique device token. If none are sent, a random UUID will be generated by the

SDK for the user. The ADID can be accessed asynchronously from the Google Play services package.

Events

Wherever you need to track app activity, make an event call to Oplytic. The general-purpose method for tracking an event is:

public static void *addEvent*(String eventAction, String eventObject, String eventId, String str1, String str2, String str3, Double num1, Double num2)

1. ***eventAction***: a string associated with the event action, for example, “view”, “shop.”
2. ***eventObject*****:** a string associated with the target of the event action, for example “map” or an object SKU.
3. ***eventId*****:** a unique string that you can pass along to associate with the event.
4. ***str1, str2, str3*****:** arbitrary strings associated with the event. You can use these to pass any sort of associated data for that event.
5. ***num1, num2*****:** arbitrary Double numeric values associated with the event. You can use these to pass any sort of associated data for that event.

For purchase events another method is provided. This method ensures that the server can properly track and aggregate purchase events consistently.

Receive Attribution Events (Optional)

If you would like to be notified of attribution events your Application class needs to implement the IAttributionHandler interface. We will pass a map of the parameters on the link that can be accessed in a key-value fashion. We will also pass in the raw URL.

```
public interface IAttributionHandler {

    void onAttribution(Map<String, String> parameters, String deepLinkURL);

}

@Override
public void onAttribution(Map<String, String> parameters, String deepLinkURL) {
    Log.d("OplyticSDK", "In onAttribution of Application");
    //Access the parameters here
    String cid = parameters.get("cid");
}
```

Testing

Please login to the Oplytic Dashboard to view a live stream of events coming in to our API for your app.