推啊广告iOS SDK使用指南

1. 概述

感谢使用推啊广告平台服务，本文档旨在帮助iOS应用开发者快速上手推啊广告平台提供的SDK。接入SDK只需简单配置，书写少量代码即可轻松展示各种类型的广告，下面是详细说明。

2. 接入准备

2.1 运行环境

本SDK最低兼容版本为iOS9.0.

2.2 术语介绍

AppKey：是您在推啊创建媒体(应用)时获得的key，是应用的唯一标识，每个应用对应一个appKey。

AppSecret：跟AppKey一样，每个应用都有一个相应的appSecret。

SlotID：广告位ID，是您在推啊管理后台创建广告位时获得的ID，您可以创建各种类型(横幅、Bannaer、浮标、开屏、自定义、插屏等)的广告位；一个媒体应用可以创建多个广告位，但需要注意的是，广告位跟appKey是配套使用的，不同应用间的广告位不能借用，否则会报错。

2.3 媒体创建以及广告位申请

1.注册登录推啊媒体管理平台，创建您要接入的媒体，获取对应的appKey和appSecret。

2.根据要接入的广告类型，新建对应的广告位，注意事项：

- 投放类型：选择SDK投放；
- 广告位规格：根据实际情况进行选择，如需自定义规格，需和媒体运营沟通后再进行新建操作。

3.在此环节遇到任何问题，可咨询媒体运营进行处理。

3. 集成SDK

本SDK可通过CocoaPods集成，在工程的Podfile里面添加以下代码：

```
pod 'TATMediaSDK'
```

保存并执行pod install，然后用后缀为 `.xcworkspace` 的文件打开工程。

3.1 Objective-C工程配置

需在Build Settings - Linking - Other Linker Flags处添加上-ObjC，然后在需要用到的地方引入头文件：

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

3.2 Swift工程配置

如果您的Swift工程已有OC桥接文件，直接import即可；若没有，则需要新建OC桥接头文件，比如YourProjectName-Bridging-Header.h，然后在Build Settings - Swift Compiler - General的Objective-C Bridging Header设置项中添加上新建的头文件：YourProjectName/YourProjectName-Bridging-Header.h。在桥接头文件添加引用后，工程中其他文件便可直接调用：

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

4. 接入代码

4.1 初始化SDK

使用SDK之前需要先进行初始化，在工程 `AppDelegate.m` 的 `application:didFinishLaunchingWithOptions:` 方法中初始化：

- **Objective-C**

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    [TATMediaCenter startWithAppKey:@"RFH45U9e2mEpKxq2BHg6VqQm6HA"
appSecret:@"3WoqJ6MwUZCMEtSgUojW8E4X2KCirUv4EgLhTLQ"];
    return YES;
}
```

- **Swift**

```
func application(_ application: UIApplication,
didFinishLaunchingWithOptions launchOptions:
[UIApplication.LaunchOptionsKey: Any]?) -> Bool {
    TATMediaCenter.start(withAppKey: "RFH45U9e2mEpKxq2BHg6VqQm6HA",
appSecret: "3WoqJ6MwUZCMEtSgUojW8E4X2KCirUv4EgLhTLQ")
    return true
}
```

4.2 开屏广告

- **Objective-C**

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    [TATMediaCenter showLaunchAdWithSlotId:@"322000"
resultBlock:^(BOOL result, NSError *error) {
        // 处理结果
    } closeBlock:^(BOOL isClosedByUser) {
        // 关闭事件的处理，可不用
    }];
    return YES;
}
```

- **Swift**

```
func application(_ application: UIApplication,
didFinishLaunchingWithOptions launchOptions:
[UIApplication.LaunchOptionsKey: Any]?) -> Bool {
    TATMediaCenter.showLaunchAd(withSlotId: "322000", resultBlock:
{ (result, error) in
        print("Call back")
    }) { (isClosedByUser) in
        print("close")
    }
    return true
}
```

4.3 横幅/浮标/Banner广告

- **Objective-C**

```
__weak __typeof(self)weakSelf = self;
__block UIView *adView = [TATMediaCenter
initSimpleAdWithSlotId:@"321995" resultBlock:^(BOOL result, NSError
*error) {
    __strong __typeof(weakSelf)self = weakSelf;
    if (result) {
        // 返回成功，调整adView的UI布局
    } else {
        // 错误处理
    }
}];
[self.view addSubview:adView];
self.adView = adView;
```

- **Swift**

```
let adView = TATMediaCenter.initSimpleAd(withSlotId: "321995") {
(result, error) in
    // 结果处理
}
self.adView = adView
self.view.addSubview(self.adView ?? UIView.init())
```

4.4 插屏广告

- **Objective-C**

```
- (void)showInterstitialAd {
    [TATMediaCenter showInterstitialWithSlotId:@"321994"
resultBlock:^(BOOL result, NSError *error) {
        if (result) {
            // 返回成功
        } else {
            // 错误处理
        }
    } closeBlock:^{
        // 关闭事件，可不处理
    }];
}
```

- **Swift**

```
func showInterstitialAd() {
    TATMediaCenter.showInterstitial(withSlotId: "321994")
}
```

4.4 自定义广告

- **Objective-C**

```
// 第1步：获取自定义广告
[TATMediaCenter fetchCustomAdWithSlotId:@"322001"
completion:^(NSError *error, TATCustomAdModel *model) {
    // 返回成功时，媒体方可选择使用推啊提供的素材，也可以展示自己的素材，当用户点击时，跳转加载activityUrl h5活动即可。
}];

// 第2步：展示成功时，须调用曝光事件上报接口，其中exposureUrl来自于上面返回的model中
[TATMediaCenter reportExposureWithURL:exposureUrl];

// 第3步：当用户点击素材时，须调用点击事件上报接口，同样clickUrl来自于上面返回的model中
[TATMediaCenter reportClickWithURL:clickUrl];
```

- **Swift**

```
// 第1步：获取自定义广告
TATMediaCenter.fetchCustomAd(withSlotId: "322001") { (error, model) in
    // 处理返回model
}

// 第2步：展示成功时，须调用曝光事件上报接口，其中exposureUrl来自于上面返回的model中
TATMediaCenter.reportExposure(withURL: exposureURL)

// 第3步：当用户点击素材时，须调用点击事件上报接口，同样clickUrl来自于上面返回的model中
TATMediaCenter.reportClick(withURL: clickURL)
```

4.5 激励广告

激励广告不是一种独立的广告类型，而是各种类型的广告都可能导向含有激励的活动。激励活动需要媒体方传入用户唯一标识符userId，并处理激励相关回调。

- **Objective-C**

```
// 设置用户唯一标识符，userId为媒体方自行制定。当用户切换登录时，记得重新设置。
[TATMediaCenter setUserId:userId];

/**
 rewardJson示例：
 {
     "orderId" : "168408070629",
     "userId" : "123",
     "timestamp" : "1566791113031",
     "prizeFlag" : "XXX",
     "appKey" : "4W8ReCvDm4fy3Fpn52MgPgUWmdfS",
     "sign" : "5093659d6bf802d1a407df81d6aab9f9",
     "score" : "1",
 } */
[TATMediaCenter sharedInstance].rewardHandler = ^(NSString * _Nullable rewardJson) {

};

// 关闭回调，目前看是没有返回json的
[TATMediaCenter sharedInstance].closeHandler = ^(NSString * _Nullable closeJson) {

};
```

- **Swift**

```
// 设置用户唯一标识符，userId为媒体方自行制定。当用户切换登录时，记得重新设置。
TATMediaCenter.setUserId(userId)

/**
 rewardJson示例：
 {
     "orderId" : "168408070629",
     "userId" : "123",
     "timestamp" : "1566791113031",
     "prizeFlag" : "XXX",
```

```
        "appKey" : "4W8ReCvDm4fy3Fpn52MgPgUWmdfS",
        "sign" : "5093659d6bf802d1a407df81d6aab9f9",
        "score" : "1",
    } */
    TATMediaCenter.sharedInstance().rewardHandler = { [weak self]
    rewardJson -> Void in

    }

    // 关闭回调，目前看是没有返回json的
    TATMediaCenter.sharedInstance().closeHandler = { [weak self]
    closeJson -> Void in

    }
```

名称	类型	备注
userId	String	用户 id，用户在媒体的唯一识别信息，来源于活动链接中的 &userId=xxx，由媒体拼接提供
timestamp	Number	时间戳
prizeFlag	String	常量，请求上报的奖励在对接方媒体系统内的奖励标识，用于标识具体的奖励类型，由媒体提供
orderId	String	推啊订单号，具有唯一性，幂等由媒体保障
appKey	String	媒体公钥
sign	String	签名
score	Number	如果是数值类型的奖励，则同时请求充值对应的数值 score，比如积分、金币、倍数