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最低兼容的系统版本为iOS 9.0.

2.2 术语介绍

- APPID：媒体ID，是您在推啊创建媒体时获得的ID，这个ID是我们在广告网络中识别您应用的唯一ID
- SLOTID：广告位ID，是您在推啊创建广告位时获得的ID，一个APPID可以拥有多个SLOTID
- AppKey、AppSecret：您在推啊创建媒体时可获得，每个应用对应一个AppKey、一个AppSecret

2.3 媒体及广告位申请

1.登录推啊媒体管理后台，创建要接入的媒体，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，可自行添加或和@推啊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@推啊运营 进行处理

创建广告位流程

流量合作 - 媒体审核通过后，开始进行广告位创建及管理

详细操作指南，请查看“推啊SDK白皮书”，了解更多信息

支持的广告类型

推啊互动广告平台支持以下几种广告类型，您可以根据开发需要选择合适的广告：

广告类型	简介	适用场景
开屏广告	App 启动时，供广告展示	app 启动时，使用开屏广 告
插屏广告	在应用开启、暂停、退出时以半屏 或全屏的形式弹出，展示时机巧妙 避开用户对应用的正常体验	视频暂停页，打开新页面可弹出插屏广告
横幅/banner	常见的移动广告样式，多位于 app 顶部或底部，横跨 app 页面	可用于小说底部，详情页 面底部，应用界面顶部等
信息流	媒体可以获取广告素材，包括广告标题、文字描述及图片	媒体需要主导广告样式 时，可以使用信息流广告
浮标	多位于页面的右下角可移动广告位	任意页面
自定义（自渲染）	媒体可以根据 APP 原生入口场景，自由渲染各种尺寸的广告	原生广告（模板方式）不能满足开发需求，可以使用自定义

支持的活动类型

推啊互动广告平台支持以下几种活动类型，您可以根据场景需要选择合适的活动：

活动类型	活动形式	适用场景	活动优势
			适用于所有常规运营位场景

轻互动	常规互动	常规运营场景	1000+活动工具，多样化、规模化的玩法内容
	激励互动	激励运营场景	适用于具备用户积分/货币体系的媒体，可深度结合运营场景发放道具奖品、积分翻倍、复活机会等权益。用户高参与度、高复参，整体收益高
深度互动	种红包	深度运营场景	红包激励：持续的红包激励并可提现，提升参与度 长线留存：平衡体验变现，留存率平均提升3-5倍 提升运营指标：结合金币/任务体系、用户行为引导
	原生插屏 （推荐）	完成页场景	案例：滚金蛋活动 兼顾体验及收益：活动出现时机可控，可根据媒体需求配置金蛋从小变大的过程时间 助力媒体做增量收益：在不影响当前页面原有广告的基础上，通过趣味性的玩法促进用户参与
		签到页场景	案例：大富翁活动 高参与率：通过奖励刺激用户参与。游戏地图中含有多种触发互动广告条件和场景，同时获得用户体验及收益 高留存率：通过每日挑战不同地图及场景中的随机事件，增加用户对签到行为的期待感

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

在进行代码接入前，先作一些简要说明。随着版本的更迭，SDK可能会累计有多个version的接口供使用，媒体方可以根据自己的需要进行选择使用。现在我们的所有广告类型，除了自定义是返回一个model，其他的都会返回TATBaseView对象。如果有需要，媒体可以接收这个TATBaseView对象，设置点击、关闭等事件的回调。为了适应一个应用中可能使用多个appKey的需求场景，我们增加了TATAdConfiguration对象作为传参，媒体如果在请求广告时想要动态配置appKey、appSecret，可调用含TATAdConfiguration参数的接口。没有需要的媒体，可调用不含此参数的接口，内部会默认使用初始化时的appKey、appSecret。

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

开屏广告从V2.0开始，支持一些参数配置，包括等待时长、显示区域、显示时长等，详细可见TATLaunchAdConfiguration.h，注意其中的sourceType，它表示App所设置的启动图类型（LaunchImage或LaunchScreen），需要媒体方根据实际情况进行设置，默认为LaunchImage方式。默认会使用defaultConfiguration，如果不想改动，可使用不传此参数的接口。

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
    }];
    adView.clickAdBlock = ^(NSString * _Nullable slotId) {
       // 点击事件回调
    };
    return YES;
}
// 或者使用TATLaunchAdConfiguration配置
TATLaunchAdConfiguration *config = [TATLaunchAdConfiguration
defaultConfiguration];
config.sourceType = TATLaunchSourceTypeScreen;
config.waitDuration = 3;
config.showDuration = 5;
config.frame = CGRectMake(0, 0, [UIScreen mainScreen].bounds.size.width ,
[UIScreen mainScreen].bounds.size.height - 180);
[TATMediaCenter showLaunchAdWithSlotId:@"322000" configuration:config
resultBlock:^(BOOL result, NSError *error) {

}];
```

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 TATBaseAdView
*adView = [TATMediaCenter initSimpleAdWithSlotId:@"321995"resultBlock:^(BO
OL 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
  }
```

```
}];
adView.clickAdBlock = ^(NSString * _Nullable slotId) {
    // 点击事件回调
};
[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 信息流

```
TATBaseAdView *adView = [TATMediaCenter initInfoFlowAdWithSlotId:@"321996" resultBlock:^(BOOL result, NSError *error) {
    if (result) {
        // 返回成功
     } else {
        // 错误处理
     }
}];
adView.clickAdBlock = ^(NSString * _Nullable slotId) {
    // 点击事件回调
};
adView.closeBlock = ^{
    // 关闭事件回调
};

```

4.5 插屏广告

Objective-C

```
- (void)showInterstitialAd {
  TATBaseAdView *adView =
[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
  }];
  adView.clickAdBlock = ^(NSString * _Nullable slotId) {
      NSString *message = [NSString stringWithFormat:@"点击广告位回调:%@", slotId];
  };
}
```

Swift

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

4.6 原生插屏（支持滚金蛋、大富翁等活动玩法）

目前原生插屏分为嵌入式和插屏式两种类型，主要是UI展示上的差别，广告位需要配置相对应的活动，即用于嵌入式显示的广告位须配置适于嵌入显示的活动。

Objective-C

```
// 全屏展示的原生插屏
[TATMediaCenter showFullModeAdWithSlotId:self.slotId loadingOption:YES resultBlock:^(BOOL result, NSError * _Nonnull error) {

}];
// 嵌入式展示
__block TATBaseAdView *adView = [TATMediaCenter initEmbedAdWithSlotId:self.slotId loadingOption:NO resultBlock:^(BOOL result, NSError * _Nonnull error) {
    // 处理返回结果（UI布局或者错误处理）
}];
[self.view addSubview:adView];
self.adView = adView
```

Swift

```
// 插屏式
TATMediaCenter.showFullModeAd(withSlotId: slotId, loadingOption:true) {(result, error)in
   // 结果处理
}
// 嵌入式
let adView:TATBaseAdView = TATMediaCenter.initEmbedAd(withSlotId: slotId, loadingOption: true) { (result, error) in
}
self.adView = adView
```

4.7 自定义广告

自定义广告由媒体方自行处理入口素材的展示以及活动的跳转，**在入口素材展示成功时，须调用曝光事件上报方法；当用户点击广告入口时，须调用点击事件上报方法。加载活动时可选择使用SDK提供的加载接口，也可以自行实现。如果是媒体自行实现，则需要在activity url后拼接上&userId=yourUserId；若使用SDK的加载接口，且之前已经通过setUserId:设置了userId，则不需要再手动拼接。注：userId禁止以固定值填充。**

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，可选自行实现或者使用SDK加载接口
}];

// 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)
```

加载活动页面，我们推荐使用SDK提供的加载接口。

```
/**
 加载自定义广告的活动页面（推荐使用）
 @param urlString 活动链接，必传
 @param slotId 广告位ID，非必传
 @param pageTitle 活动标题，非必传
 */
+ (BOOL)loadActivityURL:(NSString *)urlString slotId:(NSString *)slotId title:(NSString *)pageTitle;
```

如果要自行实现，则在加载活动的webView中处理激励相关回调，主要就是实现WKScriptMessageHandler这个协议，处理TAHandlerReward、TAHandlerClose这两个前端调用方法。

Objective-C

```
static NSString * const kJSH5RewardProtocol = @"TAHandlerReward";
static NSString * const kJSH5CloseProtocol = @"TAHandlerClose";
// 初始化webview的时候，加上configuration:
WKWebViewConfiguration *configuration = [[WKWebViewConfiguration alloc] init];
WKUserContentController *userController = [[WKUserContentController alloc] init];
configuration.userContentController = userController;
[userController addScriptMessageHandler:self name:kJSH5RewardProtocol];
[userController addScriptMessageHandler:self name:kJSH5CloseProtocol];
self.webView = [[WKWebView alloc] initWithFrame:frame configuration:configuration];
//然后实现WKScriptMessageHandler协议的方法:
- (void)userContentController:(WKUserContentController *)userContentController didReceiveScriptMessage:(WKScriptMessage *)message{
    if ([message.name isEqualToString:kJSH5RewardProtocol]) {
        id body = message.body;
        // 在这里处理激励回调，具体数据格式见4.5激励类活动接入
    } else if ([message.name isEqualToString:kJSH5CloseProtocol]) {
        id body = message.body;
         // 在这里处理关闭回调
    }
}

```

另外，在加载webView的视图控制器中，导航栏返回按钮的action需要自定义，处理webView的返回，代码类似下面:

```
- (void)returnPress:(id)sender {
    if (self.webView.canGoBack) {
        [self.webView goBack];
    } else {
        [self close];
    }
}
```

5.【重要】激励类活动接入

注：接入激励互动等场景，必看指南！！

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。

注：userId禁止以固定值填充！

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" : "1684080706291",
    "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" : "1684080706291",
    "userId" : "123",
    "timestamp" : "1566791113031",
    "prizeFlag" : "XXX",
    "appKey" : "4W8ReCvDm4fy3Fpn52MgPgUWmdfS",
    "sign" : "5093659d6bf802d1a407df81d6aab9f9",
    "score" : "1",
} */
TATMediaCenter.sharedInstance().rewardHandler = { [weak self] rewardJson -> Voidin

}

// 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

6.【重要】种红包-服务端发奖接口

重要：接入种红包等场景，必看指南！！

1. 媒体应用提供虚拟商品 API 充值 URL 及该虚拟商品在媒体系统内的唯一标识符， 由推啊配置在特定虚拟商品上。
2. 媒体必须提供完全匹配企业应用可信域名，推啊测试可用后可在线上使用。

参数说明：

请求方式：POST application/x-www-form-urlencoded			
参数	类型	必须	说明
userId	String	是	用户 id，用户在媒体下的唯一识别信息，来源于活动链接中的 &userId=xxx，由媒体拼接提供
timestamp	Long	是	时间戳，系统当前毫秒数
prizeFlag	String	是	请求充值的虚拟商品在对接方媒体系统内的标识 符，用于标识具体的虚拟商品，具体由媒体提供
account	String	否	用户领取奖品时输入

			的账号，多账号用逗号拼接
orderId	String	是	推啊订单号，具有唯一性，幂等由媒体保障
appKey	String	是	媒体信息
sign	String	是	签名
score	Integer	否	如果充值的是数值类型的虚拟商品，则同时请求充值对应的数值score，比如积分、金币等
reason	String	是	充值理由

响应说明：

参数	类型	必须	说明
code	String	是	“0”：成功，“-1”：重填，“其他”：充值异常。注意：响应 code 类型需为 String
msg	String	是	充值失败信息
orderId	String	是	推啊订单号

extParam	String	是	用户设备id，Android：imei；ios：idfa
			用户id：用户唯一标识
			{
			”deviceId”:”867772035090410”,
			”userId”:”123456”
			}

响应示例：

```
{
"code":"0",
"msg":"成功",
"orderId":"12345678123",
"extParam":
{
"deviceId":"867772035090410",
"userId":"123456"
}
```

```
}
```

验证请求：

请求验证分为时效性验证和签名验证，验证通过再充值。

时效性验证：5 分钟失效，如：

```
Date date =null; try {
date = new Date(Long.valueOf(timestamp));
} catch (Exception e)
{ return false;
}
//5 分钟失效
return date.after(new Date(new Date().getTime()-5*60*1000L));
```

签名验证：

```
StringBuilder sb = new StringBuilder(); sb.append(timestamp);//时间戳
sb.append(prizeFlag);//虚拟商品标识
sb.append(orderId);//订单号
sb.append(appKey);//媒体信息sb.append(appSecret);//媒体密钥try {
String sign = MD5.md5(sb.toString()); if(map.get("sign").equals(sign)){
return true;
}
} catch (NoSuchAlgorithmException | UnsupportedEncodingException e) {
log.warn("虚拟商品充值签名失败 msg={}",e.getMessage(),e);
}
return false;
```

安全策略：

连续积累300 个订单请求无响应，将停止为该媒体充值服务； 补偿策略减少无响应订单数量后，自劢开启充值服务；

补偿策略：

2 秒超时无响应订单，进入重试补偿机制，重试间隔策略为30s，60s，120s；

重试次数为3 次，否则进入人工补偿。

更详细内容可参考激励广告对接文档-服务端上报

7.FAQ

7.1 各种事件回调怎么设置

关闭活动页的回调：

```
[TATMediaCenter sharedInstance].closeH5Block = ^(NSString * _Nullable
slotId){
     NSString *message = [NSString stringWithFormat:@"关闭活动页回调:%@",
slotId];
};
```

广告位的点击跟关闭：部分接口参数可传递这些block，也可以通过返回的TATBaseView对象进行设置，后续如果有新的事件回调也会在这里添加。

```
@interface TATBaseAdView : UIView
/**
* 点击事件的回调
*/
@property (nonatomic, copy) TATClickAdBlock _Nullable clickAdBlock;
/**
* 关闭事件的回调
*/
@property (nonatomic, copy) TATCloseAdBlock _Nullable closeBlock;
@end

```

7.2 设置userId

userId是由媒体传入的用户唯一标识，禁止以固定值填充。媒体在用户登录之后调用即可，切换登录时需重新设置。

```
/**
 当用户登录之后设置用户ID，切换账号登录时记得重新设置
 @param userId 用户唯一标识，含奖品的活动需要用到
*/
+ (void)setUserId:(NSString *)userId;
```