快手广告iOS-SDK接入文档

快手广告iOS-SDK接入文档

- 升级iOS14注意事项
  - 配置ATT支持
- 1. 接入准备
- 2. 接入入SDK
  - 2.1 CocoaPods 方式接入
  - 2.2 手动引入
  - 2.3 添加权限
  - 2.4 SDK初始化
- 3. 接口使用
  - 3.1 激励视频广告
    - 初始化
    - 展示激励视频
    - 激励相关回调
    - 激励视频再看一个场景
    - 激励视频的服务端回调支持
    - 激励视频内部广告监听
  - 3.2 全屏视频广告
    - 初始化
    - 展示全屏视频
    - 全屏相关回调
  - 3.3 媒体自渲染广告
    - 初始化
    - 自渲染相关回调
    - 开放字段
    - NativeFeed页面展示广告的注意事项
  - 3.4 信息流广告
    - 初始化
    - 信息流相关回调
  - 3.5 Draw竖版信息流广告
    - 初始化
    - Draw相关回调
  - 3.6 开屏广告
    - 初始化
    - 屏蔽摇一摇
    - 开屏相关回调
  - 3.7 插屏广告
    - 初始化
    - 插屏相关回调

3.8 媒体二次议价
3.9 广告曝光失败后上报失败原因
4. 错误码
5. FQA

升级iOS14注意事项

App Tracking Transparency (ATT) 适用于请求用户授权，访问与应用相关的数据已跟踪用户或者设备。访问https://developer.apple.com/documentation/apptrackingtransparency了解更多信息

配置ATT支持

从 iOS 14 开始，若开发者设置 App Tracking Transparency 向用户申请跟踪授权，在用户授权之前IDFA 将不可用。 如果用户拒绝此请求，应用获取到的 IDFA 将是0000-0000-0000-0000，可能会导致广告收入降低
要获取 App Tracking Transparency 权限，请更新 Info.plist，添加 NSUserTrackingUsageDescription 字段和自定义文案描述。
注意：iOS 14.5之前，应用不配置ATT也能正常获取idfa，从iOS14.5之后，应用必须配置ATT，调用授权方法向用户申请ATT权限。
代码示例：

```
<key>NSUserTrackingUsageDescription</key>
<string>{{使用描述，需要开发者自行填写}}</string>
```

向用户申请权限：

```
#import <AppTrackingTransparency/AppTrackingTransparency.h>
[ATTrackingManager
requestTrackingAuthorizationWithCompletionHandler:^(ATTrackingManagerAuthorizationStatu
s status) {
    // Tracking authorization completed. Start loading ads here.
    // [self loadAd];
}];
```

1. 接入准备

接入SDK前请您按照要求在快手手平台申请appId，申请appId时需要您填写应用名称和应用包名等信息。

2. 接入入SDK

推荐使用 CocoaPods 方式接入，CocoaPods 会协助自动处理编译问题。

2.1 CocoaPods 方式接入

在 Podfile 里指明依赖：

```
pod 'KSAdSDK', 'version'
#version为SDK指定版本号，版本列表可通过 pod search KSAdSDK 命令查看或者通过官网（需要登录）
https://u.kuaishou.com/access 获取
```

注意：如果pod install找不到最新版本，可以尝试先pod repo update后再pod install，或者pod install --repo-update

2.2 手动引入

1. 将 SDK （KSAdSDK.framework）文件加入工程（手动拖动时，选择 Copy items if needed）；
2. 在工程上链接上这些系统库:

```
'Foundation', 'UIKit', 'MobileCoreServices', 'CoreGraphics', 'Security',
'SystemConfiguration', 'CoreTelephony', 'AdSupport', 'CoreData', 'StoreKit',
'AVFoundation', 'MediaPlayer', 'CoreMedia', 'WebKit', 'Accelerate', 'CoreLocation',
'AVKit', 'MessageUI', 'QuickLook', 'AddressBook', 'libz.tbd', 'resolv.9', 'sqlite3',
'c++', 'c++abi', 'CoreMotion'
```

3. 注意：为了方便模拟器开发，SDK 带有 x86_64, i386 架构。在打发布到 AppStore 的安装包时需要移除这两个架构（CocoaPods 方式接入会自动移除）。移除脚本可以参考：https://stackoverflow.com/questions/30547283/submit-to-app-store-issues-unsupported-architecture-x86

2.3 添加权限

1. 工程plist文件设置，点击右边的information Property List后边的 "+" 展开
2. 添加 App Transport Security Settings，先点击左侧展开箭头，再点右侧加号，Allow Arbitrary Loads 选项自动加入

```
<key>NSAppTransportSecurity</key>
<dict>
     <key>NSAllowsArbitraryLoads</key>
     <true/>
</dict>
```

2.4 SDK初始化

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

NSString *appId = @"90010";
[KSAdSDKManager setAppId:appId];
```

```
// 根据需要设置日志级别
[KSAdSDKManager setLoglevel:KSAdSDKLogLevelOff];

// 个性化推荐开关：关闭后，看到的广告数量不变，相关度将降低。
// 是否允许开启广告的个性化推荐（NO-关闭，YES-开启），由开发者通过SDK的接口来设置。不设置的话则默认为YES。
[KSAdSDKManager setEnablePersonalRecommend:YES];

// 程序化推荐开关：关闭后，看到的广告数量不变，但将不会为你推荐程序化广告。
// 是否允许开启广告的程序化推荐（NO-关闭，YES-开启），由开发者通过SDK的接口来设置。不设置的话则默认为YES。
[KSAdSDKManager setEnableProgrammaticRecommend:YES];
```

3. 接口使用

3.1 激励视频广告

初始化

目前已支持服务端回调（需要在 ssp后台配置），对应参数可通过KSRewardedVideoModel传递，具体实现如下：

```
NSString *posId = @"90010001";
KSRewardedVideoModel *model = [KSRewardedVideoModel new];
// 如果开启服务端回调的话，userId和extra会通过回调接口回传给接入方
model.userId = @"123234";
model.extra = @"test extra";
self.rewardedVideoAd = [[KSRewardedVideoAd alloc] initWithPosId:self.posId
rewardedVideoModel:model];
self.rewardedVideoAd.delegate = self;
// 可设置竖屏or横屏，见KSAdShowDirection，若不设置默认为竖屏
self.rewardedVideoAd.showDirection = KSAdShowDirection_Vertical;
// 加载广告
[self.rewardedVideoAd loadAdData];
```

展示激励视频

```
- (BOOL)showAdFromRootViewController:(UIViewController *)rootViewController;
```

激励相关回调

```
/**
 * 激励|已加载广告数据
 */
- (void)rewardedVideoAdDidLoad:(KSRewardedVideoAd *)rewardedVideoAd;
/**
 * 激励|视频广告从请求到缓存到展示发生任何失败均调用此方法
```

```
 */
- (void)rewardedVideoAd:(KSRewardedVideoAd *)rewardedVideoAd didFailWithError:(NSError *_Nullable)error;
/**
 * 激励|缓存视频等资源成功时调用此方法，收到此回调后广告才能开始展示，否则资源是未下载完成的
 */
- (void)rewardedVideoAdVideoDidLoad:(KSRewardedVideoAd *)rewardedVideoAd;
/**
 * 激励|视频广告即将可见
 */
- (void)rewardedVideoAdWillVisible:(KSRewardedVideoAd *)rewardedVideoAd;
/**
 * 激励|视频广告已经可见
 */
- (void)rewardedVideoAdDidVisible:(KSRewardedVideoAd *)rewardedVideoAd;
/**
 * 激励|视频广告即将关闭
 */
- (void)rewardedVideoAdWillClose:(KSRewardedVideoAd *)rewardedVideoAd;
/**
 * 激励|视频广告已经关闭
 */
- (void)rewardedVideoAdDidClose:(KSRewardedVideoAd *)rewardedVideoAd;
/**
 * 激励|视频广告被点击
 */
- (void)rewardedVideoAdDidClick:(KSRewardedVideoAd *)rewardedVideoAd;
/**
 * 激励|视频广告播放完成或发生错误
 */
- (void)rewardedVideoAdDidPlayFinish:(KSRewardedVideoAd *)rewardedVideoAd didFailWithError:(NSError *_Nullable)error;
/**
 * 激励|视频广告开始播放
 */
- (void)rewardedVideoAdStartPlay:(KSRewardedVideoAd *)rewardedVideoAd;

#pragma mark - 用户相关行为回调
/**
 * 激励|广告被点击跳过
 */
- (void)rewardedVideoAdDidClickSkip:(KSRewardedVideoAd *)rewardedVideoAd;
/**
 * 激励|用户获得奖励时回调
 */
- (void)rewardedVideoAd:(KSRewardedVideoAd *)rewardedVideoAd hasReward:(BOOL)hasReward;
/**
 * 激励|分阶段奖励回调（激励广告新玩法，相关政策请联系商务或技术支持）
 */
```

```
- (void)rewardedVideoAd:(KSRewardedVideoAd *)rewardedVideoAd
              hasReward:(BOOL)hasReward
               taskType:(KSAdRewardTaskType)taskType
         currentTaskType:(KSAdRewardTaskType)currentTaskType;
```

激励视频再看一个场景

再看一个需要设置一个新的delegate对象，回调事件与普通激励视频一样

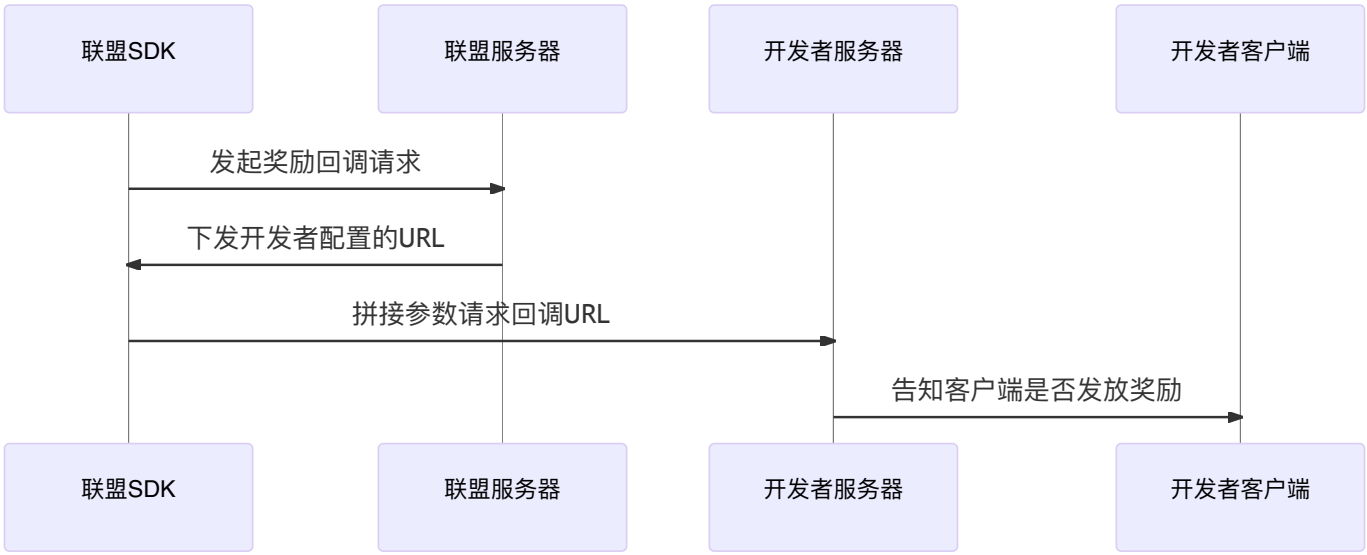
```
//设置 "再看一个" 的delegate，和激励视频的普通delegate不能是同一个对象
self.rewardedVideoAd.rewardPlayAgainInteractionDelegate =
self.expressRewardedVideoAgainDelegateObj;
```

> 具体请参考Demo中`KSRewardedVideoDemoViewController`类

激励视频的服务端回调支持

激励视频的有效性回调，支持开发者服务端回调，SDK会在激励视频有效的时候，GET请求回调 url（该url需要开发者在SSP平台中进行配置），这样开发者可以在自己的服务端对激励视频的有效性进行再次验证。

整体的调用流程如下：



在SSP平台配置回调url的时候，需要按照以下格式进行配置：

```
https://your_callback_url?
userId=__UID__&transId=__TRANSID__&sign=__SIGN__&amount=__RAMOUNT__&name=__RNAME__&extr
a=__EXTRA__
```

注意URL中不要带有空格、换行等特殊字符

其中 “https://your_callback_url” 为客户的服务端回调url的地址，后面为各个参数的设置，客户可以按照需要进行配置。例如：

```
https://your_callback_url?userId=__UID__&transId=__TRANSID__
```

这样配置的话，则SDK在调用客户服务端url的时候，只会包含 `userId` 和 `transId` 这两个参数，而不包含其他的。

> 注意：默认不再支持http请求，所以建议回调url都使用 https 协议。

在SSP配置完成后，开发者使用该功能的时候，需要在请求激励视频的时候，通过KSRewardedVideoModel对象设置相关的参数，参数示例如下：

```
KSRewardedVideoModel *model = [KSRewardedVideoModel new];
// 开发者系统中的用户id，会在请求客户的回调url中带上
model.userId = @"123234";
// 开发者自定义的附加参数，会在请求客户的回调url中带上
model.extra = @"test extra";
self.rewardedVideoAd = [[KSRewardedVideoAd alloc] initWithPosId:self.posId
rewardedVideoModel:model];
```

联盟SDK在调用开发者指定的回调url的时候，会附加一些参数（形式为GET请求的参数，配置方法在上文中）供开发者服务端使用，参数示例如下：

```
http://your_callback_url?userId=your-
uerid&transId=test_trans_id&sign=11121222&amount=0&name=name&extra=your-extra-data
```

GET请求的各个参数说明：

参数名称	参数类型	参数说明
extra	String	开发者在SDK中请求激励视频时设置的自定义附加参数“extraData”
userId	String	开发者在SDK中请求激励视频时设置的 “thirdUserId” 参数
transId	String	完成观看的唯一交易ID
sign	String	签名
name	String	奖励名称
amount	int	奖励数量

> 上表中提到的sign为唯一签名，计算方式为:
>
> sign = md5(appSecurityKey:transId) ，所有字母均为小写。
>
> 其中 appSecurityKey 为在SSP平台设置回调url时获得，transId为请求中的参数。

SDK在请求开发者的指定url的时候，开发者需要按照一定的格式返回给SDK结果，返回数据为json的格式，详情如下:

字段名称	字段定义	字段类型	备注
isValid	校验结果	bool	判定结果，是否发放奖励。

```
{
  "isValid" : true
}
```

激励视频内部广告监听

激励视频的内部，会在一些场景中请求并展示广告（目前有聚合页和回流页，相关功能的开通请联系技术支持人员），SDK支持设置对内部广告的监听，具体设置代码可参考 KSRewardedVideoDemoViewController。

1. 监听设置

```
self.rewardedVideoAd = [[KSRewardedVideoAd alloc] initWithPosId:self.posId
rewardedVideoModel:model];
self.rewardedVideoAd.delegate = self;
//新增内部广告监听（可选）
self.rewardedVideoAd.innerDelegate = self;
```

2. 内部广告回调实现

```
#pragma mark - KSInnerAdDelegate Optional
```

```
//内部广告曝光事件回调
- (void)onInnerAdShow:(KSInnerVideoAd *)innerAd {
    if (innerAd.adType == KSInnerAdReflow) {
        NSLog(@"【激励视频回流页】曝光");
        return;
    }
    if (innerAd.adType == KSInnerAdAggregation) {
        NSLog(@"【激励视频中间聚合广告】曝光");
        return;
    }
    NSLog(@"【未知来源广告】曝光");
}
//内部广告点击事件回调
- (void)onInnerAdClick:(KSInnerVideoAd *)innerAd {
    if (innerAd.adType == KSInnerAdReflow) {
        NSLog(@"【激励视频回流页】点击");
        return;
    }
    if (innerAd.adType == KSInnerAdAggregation) {
        NSLog(@"【激励视频中间聚合广告】点击");
        return;
    }
    NSLog(@"【未知来源广告】点击");
}
```

3.2 全屏视频广告

初始化

```
self.posId = @"90010002";
self.fullscreenVideoAd = [[KSFullscreenVideoAd alloc] initWithPosId:self.posId];
self.fullscreenVideoAd.delegate = self;
// 可设置竖屏or横屏，见KSAdShowDirection，若不设置默认为竖屏
self.fullscreenVideoAd.showDirection = KSAdShowDirection_Vertical;
// 加载广告
[self.fullscreenVideoAd loadAdData];
```

展示全屏视频

```
- (BOOL)showAdFromRootViewController:(UIViewController *)rootViewController;
```

全屏相关回调

```
/**
 * 全屏|已加载广告数据
 */
- (void)fullscreenVideoAdDidLoad:(KSFullscreenVideoAd *)fullscreenVideoAd;
/**
 * 全屏|视频广告从请求到缓存到展示发生任何失败均调用此方法
 */
- (void)fullscreenVideoAd:(KSFullscreenVideoAd *)fullscreenVideoAd didFailWithError:
(NSError *_Nullable)error;
/**
 * 全屏|缓存视频等资源成功时调用此方法，收到此回调后广告才能开始展示，否则资源是未下载完成的
 */
- (void)fullscreenVideoAdVideoDidLoad:(KSFullscreenVideoAd *)fullscreenVideoAd;
/**
 * 全屏|视频广告即将可见
 */
- (void)fullscreenVideoAdWillVisible:(KSFullscreenVideoAd *)fullscreenVideoAd;
/**
 * 全屏|视频广告已经可见
 */
- (void)fullscreenVideoAdDidVisible:(KSFullscreenVideoAd *)fullscreenVideoAd;
/**
 * 全屏|视频广告即将关闭
 */
- (void)fullscreenVideoAdWillClose:(KSFullscreenVideoAd *)fullscreenVideoAd;
/**
 * 全屏|视频广告已经关闭
 */
- (void)fullscreenVideoAdDidClose:(KSFullscreenVideoAd *)fullscreenVideoAd;
/**
 * 全屏|视频广告被点击
 */
- (void)fullscreenVideoAdDidClick:(KSFullscreenVideoAd *)fullscreenVideoAd;
/**
 * 全屏|视频广告播放完成或发生错误
 */
- (void)fullscreenVideoAdDidPlayFinish:(KSFullscreenVideoAd *)fullscreenVideoAd
didFailWithError:(NSError *_Nullable)error;
/**
 * 全屏|视频广告开始播放
 */
- (void)fullscreenVideoAdStartPlay:(KSFullscreenVideoAd *)fullscreenVideoAd;

#pragma mark - 用户相关行为回调
/**
 * 全屏|广告被点击跳过
 */
```

```
- (void)fullscreenVideoAdDidClickSkip:(KSFullscreenVideoAd *)fullscreenVideoAd;
```

> 具体请参考Demo中`KSFullscreenVideoDemoViewController`类

3.3 媒体自渲染广告

初始化

```
self.nativeAdsManager = [[KSNativeAdsManager alloc] initWithPosId:posId];
self.nativeAdsManager.delegate = self;
[self.nativeAdsManager loadAdDataWithCount:5];
```

自渲染相关回调

```
#pragma mark - KSNativeAdsManagerDelegate
/**
 * 自渲染|广告数据加载成功
 */
- (void)nativeAdsManagerSuccessToLoad:(KSNativeAdsManager *)adsManager nativeAds:
(NSArray<KSNativeAd *> *_Nullable)nativeAdDataArray {
    NSInteger count = self.dataArray.count;
    for (KSNativeAd *nativeAd in nativeAdDataArray) {
        NSInteger index = arc4random() % count;
        [self.dataArray insertObject:nativeAd atIndex:index];
    }
    [self.tableView reloadData];
}
/**
 * 自渲染|广告数据加载失败
 */
- (void)nativeAdsManager:(KSNativeAdsManager *)adsManager didFailWithError:(NSError
*_Nullable)error;

#pragma mark - KSNativeAdDelegate
/**
 * 自渲染|广告素材获取成功
 */
- (void)nativeAdDidLoad:(KSNativeAd *)nativeAd;
/**
 * 自渲染|广告素材获取失败
 */
- (void)nativeAd:(KSNativeAd *)nativeAd didFailWithError:(NSError *_Nullable)error;
/**
 * 自渲染|每次广告展示都会调用此方法,请不要用于曝光计数,请使用"nativeAdDidShow"作为曝光计数。
 */
- (void)nativeAdDidBecomeVisible:(KSNativeAd *)nativeAd;
/**
```

```
 * 自渲染|广告被点击
 */
- (void)nativeAdDidClick:(KSNativeAd *)nativeAd withView:(UIView *_Nullable)view;
/**
 * 自渲染|广告详情页打开
 */
- (void)nativeAdDidShowOtherController:(KSNativeAd *)nativeAd interactionType:
(KSAdInteractionType)interactionType;
/**
 * 自渲染|广告详情页关闭
 */
- (void)nativeAdDidCloseOtherController:(KSNativeAd *)nativeAd interactionType:
(KSAdInteractionType)interactionType;
/**
 * 自渲染|广告曝光（每个广告只回调一次，可用于广告曝光计数）
 */
- (void)nativeAdDidShow:(KSNativeAd *)nativeAd;
/**
 * 自渲染|视频广告资源准备就绪
 */
- (void)nativeAdVideoReadyToPlay:(KSNativeAd *)nativeAd;
/**
 * 自渲染|视频广告开始播放
 */
- (void)nativeAdVideoStartPlay:(KSNativeAd *)nativeAd;
/**
 * 自渲染|视频广告播放结束
 */
- (void)nativeAdVideoPlayFinished:(KSNativeAd *)nativeAd;
/**
 * 自渲染|视频广告播放错误
 */
- (void)nativeAdVideoPlayError:(KSNativeAd *)nativeAd;
/**
 * 自渲染|视频广告播放暂停（包括系统暂停和用户手动暂停）
 */
- (void)nativeAdVideoPause:(KSNativeAd *)nativeAd;
/**
 * 自渲染|视频广告播放恢复（包括系统恢复和用户恢复）,视频第一次开始播放时不调用该方法
 */
- (void)nativeAdVideoResume:(KSNativeAd *)nativeAd;
```

开放字段

```
typedef NS_ENUM(int, KSAdSourceLogoType) {
    KSAdSourceLogoTypeWhite,
    KSAdSourceLogoTypeGray,
};
```

```objc
@interface KSMaterialMeta : NSObject

///广告支持的交互类型
@property (nonatomic, assign) KSAdInteractionType interactionType;

/// material pictures.
@property (nonatomic, strong) NSArray<KSAdImage *> *imageArray;

/// ad logo,可能为空
- (NSString *)adSourceLogoURL:(KSAdSourceLogoType)type;
/// ad source.
@property (nonatomic, copy) NSString *adSource;

@property (nonatomic, strong, nullable) KSAdImage *appIconImage;

/// 0-5
@property (nonatomic, assign) CGFloat appScore;
/// downloadCountDesc.
@property (nonatomic, copy) NSString *appDownloadCountDesc;


/// ad description.
@property (nonatomic, copy) NSString *adDescription;

/// text displayed on the creative button.
@property (nonatomic, copy) NSString *actionDescription;

/// display format of the in-feed ad, other ads ignores it.
@property (nonatomic, assign) KSAdMaterialType materialType;

// video duration
@property (nonatomic, assign) NSInteger videoDuration;

@property (nonatomic, strong) KSAdImage *videoCoverImage;
@property (nonatomic, copy) NSString *videoUrl;
// app name
@property (nonatomic, copy) NSString *appName;
// product name (for h5)
@property (nonatomic, copy) NSString *productName;
```

> 具体请参考Demo中`KSNativeAdsManagerFeedDemoViewController`类

NativeFeed页面展示广告的注意事项

请参照`- (void)refreshWithData:(KSNativeAd *)nativeAd`方法

```
// 这两句是必须写的
[self.relatedView refreshData:nativeAd];
// 这句如果不加 视频无法自动播放 clickableViews数组里的view会被添加统一的点击事件，跳转和上报，没有可以传nil 此方法默认contentView可点击
[nativeAd registerContainer:self.contentView withClickableViews:@[self.button, self.descriptionLabel]];
/*
 * 若希望控制contentView是否可响应点击事件，可使用如下方法
 * [nativeAd registerContainer:self.contentView withClickableViews:@[self.button, self.descriptionLabel] containerClickable:clickable];
 * clickable:contentView是否可响应点击事件
 */
```

3.4 信息流广告

初始化

```
self.feedAdsManager = [[KSFeedAdsManager alloc] initWithPosId:self.posId size:CGSizeMake(proposalWidth, 0)];
self.feedAdsManager.delegate = self;
[self.feedAdsManager loadAdDataWithCount:3];
```

信息流相关回调

```
#pragma mark - KSFeedAdsManagerDelegate
/**
 * 信息流|广告数据返回，开始渲染
 */
- (void)feedAdsManagerReadyToRender:(KSFeedAdsManager *)adsManager feedAds:(NSArray<KSFeedAd *> *_Nullable)feedAdDataArray;
/**
 * 信息流|广告数据加载成功
 */
- (void)feedAdsManagerSuccessToLoad:(KSFeedAdsManager *)adsManager nativeAds:(NSArray<KSFeedAd *> *_Nullable)feedAdDataArray {
    [self refreshWithData:adsManager];
}
/**
 * 信息流|广告数据加载失败
 */
- (void)feedAdsManager:(KSFeedAdsManager *)adsManager didFailWithError:(NSError *_Nullable)errorl;

#pragma mark - KSFeedAdDelegate
/**
 * 信息流|每次广告展示都会调用此方法,请不要用于曝光计数,请使用"feedAdDidShow"作为曝光计数。
 */
- (void)feedAdViewWillShow:(KSFeedAd *)feedAd;
```

```
/**
 * 信息流|广告被点击
 */
- (void)feedAdDidClick:(KSFeedAd *)feedAd;
/**
 * 信息流|用户手动点击不喜欢按钮
 */
- (void)feedAdDislike:(KSFeedAd *)feedAd {
    NSMutableArray *ary = [self.dataArray mutableCopy];
    [ary removeObject:feedAd];
    self.dataArray = ary;
    [self.tableView reloadData];
}
/**
 * 信息流|广告详情页打开
 */
- (void)feedAdDidShowOtherController:(KSFeedAd *)nativeAd interactionType:
(KSAdInteractionType)interactionType;
/**
 * 信息流|广告详情页关闭
 */
- (void)feedAdDidCloseOtherController:(KSFeedAd *)nativeAd interactionType:
(KSAdInteractionType)interactionType;
/**
 * 信息流|广告曝光（每个广告只回调一次，可用于广告曝光计数）
 */
- (void)feedAdDidShow:(KSFeedAd *)feedAd;
```

> 具体请参考Demo中`KSFeedAdsManagerFeedDemoViewController`类
> 注意：3.3.2版本之后的Feed 广告支持动态模板，广告布局样式可以通过后台配置，Demo中的KSFeedAdCell中不需要再手动设置间距

```
@implementation KSFeedAdCell
- (void)refreshWithFeedAd:(KSFeedAd *)feedAd {
    UIView *view = [self.contentView viewWithTag:100];
    [view removeFromSuperview];
    feedAd.feedView.tag = 100;
    [self.contentView addSubview:feedAd.feedView];
    [feedAd.feedView mas_remakeConstraints:^(MASConstraintMaker *make) {
        make.edges.equalTo(@0);
    }];
}
@end
```

3.5 Draw竖版信息流广告

初始化

```
self.drawAdsManager = [[KSDrawAdsManager alloc] initWithPosId:posId];
self.drawAdsManager.delegate = self;
[self.drawAdsManager loadAdDataWithCount:1];
```

Draw相关回调

```
#pragma mark - KSDrawAdsManagerDelegate
/**
 * Draw|广告数据加载成功
 */
- (void)drawAdsManagerSuccessToLoad:(KSDrawAdsManager *)adsManager drawAds:
(NSArray<KSDrawAd *> *_Nullable)drawAdDataArray {
    NSInteger index = MIN(self.dataArray.count / 2, 2);
    NSMutableArray *mutAry = [self.dataArray mutableCopy];
    for (NSInteger i = 0; i < adsManager.data.count; i++) {
        KSDrawAd *drawAd = adsManager.data[i];
        [mutAry insertObject:drawAd atIndex:(index + i)];
    }
    self.dataArray = mutAry;
    [self.tableView reloadData];
}
/**
 * Draw|广告数据加载失败
 */
- (void)drawAdsManager:(KSDrawAdsManager *)adsManager didFailWithError:(NSError
*_Nullable)error;

#pragma mark - KSDrawAdDelegate
/**
 * Draw|广告即将展示
 */
- (void)drawAdViewWillShow:(KSDrawAd *)drawAd;
/**
 * Draw|广告被点击
 */
- (void)drawAdDidClick:(KSDrawAd *)drawAd;
/**
 * Draw|广告详情页打开
 */
- (void)drawAdDidShowOtherController:(KSDrawAd *)drawAd interactionType:
(KSAdInteractionType)interactionType;
/**
 * Draw|广告详情页关闭
 */
```

```
- (void)drawAdDidCloseOtherController:(KSDrawAd *)drawAd interactionType:
(KSAdInteractionType)interactionType;
```

> 具体请参考Demo中`KSDrawAdDemoViewController`类

3.6 开屏广告

使用KSSplashAdView实现开屏（KSAdSplashManager已废弃， 3.3.10版本开始支持ssp设置跳过按钮出现时间和是否显示倒计时，需要使用KSSplashAdView实现）示例如下：

初始化

```
//初始化开屏广告
self.splashAdView = [[KSSplashAdView alloc] initWithPosId:posId];
self.splashAdView.delegate = self;
//开屏广告建议设置rootViewController，未设置取keyWindow的VC
self.splashAdView.rootViewController = root;
```

屏蔽摇一摇

屏蔽开屏转化区域样式，比如摇一摇等，如果需要设置，请在请求广告前，添加以下代码：

```
KSAdSplashAdExtraDataModel *extraModel = [[KSAdSplashAdExtraDataModel alloc] init];
extraModel.disableShake = YES;///是否屏蔽摇一摇，false或者不赋值，不屏蔽，true屏蔽
```

开屏相关回调

注：小窗功能相关接口已废弃

```
#pragma -mark KSSplashAdDelegate
/**
 * 开屏|已加载广告数据
 */
- (void)ksad_splashAdDidLoad:(KSSplashAdView *)splashAdView;
/**
 * 开屏|已加载完毕广告素材,准备展示，在此处展示广告
 */
- (void)ksad_splashAdContentDidLoad:(KSSplashAdView *)splashAdView {
    //展示开屏广告
    self.splashAdView.frame = [UIScreen mainScreen].bounds;
    [self.splashAdView showInView:self.window];
}
/**
 * 开屏|广告已经可见
 */
- (void)ksad_splashAdDidVisible:(KSSplashAdView *)splashAdView;
/**
```

```
 * 开屏|视频广告开始播放（仅适用于视频广告）
 */
- (void)ksad_splashAdVideoDidBeginPlay:(KSSplashAdView *)splashAdView;
/**
 * 开屏|广告数据加载失败
 */
- (void)ksad_splashAd:(KSSplashAdView *)splashAdView didFailWithError:(nonnull NSError
*)error {
    [self removeSplash]; // 需移除开屏广告
    [self removeSplashBg];
    [self.navigationController popViewControllerAnimated:NO];
}
/**
 * 开屏|点击跳过按钮调用此方法
 */
- (void)ksad_splashAd:(KSSplashAdView *)splashAdView didSkip:
(NSTimeInterval)playDuration {
    // 需移除开屏广告
    [self removeSplash];
    [self removeSplashBg];
    [self.navigationController popViewControllerAnimated:NO];
}
/**
 * 开屏|广告被点击
 */
- (void)ksad_splashAdDidClick:(KSSplashAdView *)splashAdView {
    //点击跳转了
    self.splashBgView.hidden = self.splashAdView.isHidden;
}
/**
 * 开屏|广告自动关闭
 */
- (void)ksad_splashAdDidAutoDismiss:(KSSplashAdView *)splashAdView {
    [self removeSplash]; // 需移除开屏广告
    [self removeSplashBg];
    [self.navigationController popViewControllerAnimated:NO];
}
/**
 * 开屏|广告关闭转化视图控制器或使用deeplink跳转
 */
- (void)ksad_splashAdDidCloseConversionVC:(KSSplashAdView *)splashAdView
interactionType:(KSAdInteractionType)interactType {
    self.splashBgView.hidden = self.splashAdView.isHidden;
}
- (void)removeSplash {
    [self.splashAdView removeFromSuperview];
    self.splashAdView = nil;
}
```

3.7 插屏广告

插屏广告支持图片，视频，实现方式如下（注意：插屏广告展示时不支持屏幕旋转，需要接入方限制）：

初始化

```
// 获取插屏视频广告
self.interstitialAd = [[KSInterstitialAd alloc] initWithPosId:self.posId];
self.interstitialAd.delegate = self;
[self.interstitialAd loadAdData];
```

插屏相关回调

```
#pragma mark - KSInterstitialAdDelegate
/**
 * 插屏|已加载广告数据
 */
- (void)ksad_interstitialAdDidLoad:(KSInterstitialAd *)interstitialAd;
/**
 * 插屏|广告渲染成功
 */
- (void)ksad_interstitialAdRenderSuccess:(KSInterstitialAd *)interstitialAd;
/**
 * 插屏|广告渲染失败
 */
- (void)ksad_interstitialAdRenderFail:(KSInterstitialAd *)interstitialAd error:(NSError
*)error;
/**
 * 插屏|广告即将可见
 */
- (void)ksad_interstitialAdWillVisible:(KSInterstitialAd *)interstitialAd;
/**
 * 插屏|广告已经可见
 */
- (void)ksad_interstitialAdDidVisible:(KSInterstitialAd *)interstitialAd;
/**
 * 插屏|广告被点击
 */
- (void)ksad_interstitialAdDidClick:(KSInterstitialAd *)interstitialAd;
/**
 * 插屏|广告被点击跳过
 */
- (void)ksad_interstitialAdDidClickSkip:(KSInterstitialAd *)interstitialAd;
/**
 * 插屏|广告即将关闭
 */
- (void)ksad_interstitialAdWillClose:(KSInterstitialAd *)interstitialAd;
```

```
/**
 * 插屏|广告已经关闭
 */
- (void)ksad_interstitialAdDidClose:(KSInterstitialAd *)interstitialAd;
/**
 * 插屏|广告详情页关闭
 */
- (void)ksad_interstitialAdDidCloseOtherController:(KSInterstitialAd *)interstitialAd
interactionType:(KSAdInteractionType)interactionType;
```

3.8 媒体二次议价

在获取到广告 model 后媒体可在 winnotice 时设置二次议价，所有广告 model 都支持设置二次议价，接口如下:

```
@interface KSAd : NSObject

// 单位:分
@property (nonatomic, readonly) NSInteger ecpm;
/**
 * @brief 设置竞价价格，单位（分）
 * @param ecpm 竞价价格
 */
- (void)setBidEcpm:(NSInteger)ecpm;
/**
 * @brief 设置竞价价格和最大竞败方出价，单位（分）
 * @param ecpm               竞价价格
 * @param highestLossEcpm    最大竞价失败方出价
 */
- (void)setBidEcpm:(NSInteger)ecpm
   highestLossEcpm:(NSInteger)highestLossEcpm;

@end
```

3.9 广告曝光失败后上报失败原因

```
@interface KSAd : NSObject

/**
 * @brief 广告曝光失败后上报失败原因
 * @param failureCode 曝光失败原因类型
 * @param reportParam 曝光失败原因描述
 *        reportParam.winEcpm 胜出者的ecpm报价（单位：分）
 *        reportParam.adnType 胜出方，见KSAdExposureReportParam.h 中KSAdExposureAdnType定义
 *        reportParam.adnName 胜出平台名，见KSAdExposureReportParam.h 中KSAdADNType平台定义
 */
- (void)reportAdExposureFailed:(KSAdExposureFailureCode)failureCode reportParam:(KSAdExposureReportParam *)reportParam;

@end
```

4. 错误码

code	说明
40001	没有网络
40002	数据解析失败
40003	广告数据为空
40004	缓存视频资源失
100001	参数有误
100002	服务器错误
100003	不允许的操作
100004	服务不可用
310001	appId未注册
310002	appId无效
310003	appId已封禁
310004	packageName与注册的packageName不一致
310005	操作系统与注册的不一致
320002	appId对应账号无效
320003	appId对应账号已封禁
330001	posId未注册
330002	posId无效
330003	posId已封禁
330004	posid与注册的appId信息不一致

5. FQA

1. [MediaRemote] OutputDeviceUID is nil 扬声器: (null)

https://stackoverflow.com/questions/56243550/new-outputdeviceuid-is-nil-msg-when-instantiating-mpvolumeview

2. 请求视频广告失败：

   1.请核对广告SDK初始化的AppID是否正确；

   2.请核对请求广告时传入的广告场景参数posId是否正确。

   3.请根据返回的错误吗，参考错误码对照表知晓问题