ULinkAd SDK iOS接入文档

1. 简介

本指导手册针对ULinkAd广告SDK，如有疑问请联系商务对接人

2. 阅读对象

本文档面向所有使用ULinkAd广告SDK的iOS开发人员、测试人员、合作伙伴以及对此感兴趣的其他用户，由此产生的广告收入进行合作分配。

3. Xcode设置

开发环境

- Xcode 14.1及以上
- iOS 11.0 及以上

App Transport Security

应用传输安全 (ATS) 是 iOS 9 中引入的隐私设置功能。苹果公司在iOS9中升级了应用网络通信安全策略，默认推荐开发者使用HTTPS协议来进行网络通信，并限制HTTP协议的请求。

为确保您的广告不受 ATS 影响，请在 `Info.plist` 中添加 `NSAppTransportSecurity`：

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

App Tracking Transparency

App Tracking Transparency (ATT) 适用于请求用户授权，访问与应用相关的数据以跟踪用户或设备。

目前苹果要求在iOS 14.5及以上的版本中必须在弹窗取得用户同意后，才可以追踪用户。对其他版本暂无明确要求，开发者可根据需要配置弹窗 。

在 `Info.plist` 中添加 `NSUserTrackingUsageDescription`，描述获取IDFA等广告标识符的用途，例如：

```
<key>NSUserTrackingUsageDescription</key>
<string>获取标记权限向您提供更优质、安全的个性化服务及内容；</string>
```

向用户申请权限时需要调用 `requestTrackingAuthorizationWithCompletionHandler:` 方法。建议等待方法回调完成后再处理广告相关逻辑。代码如下：

```
#import <AppTrackingTransparency/AppTrackingTransparency.h>
#import <AdSupport/AdSupport.h>

- (void)requestIDFA {
  [ATTrackingManager requestTrackingAuthorizationWithCompletionHandler:^(ATTrackir
    //  授权完成回调

  }];
}
```

iOS17隐私策略适配

如果您的项目已经集成了 `PrivacyInfo.xcprivacy`，建议您将所有SDK的条款补充到您自身App的 `PrivacyInfo.xcprivacy` 中，在补充时，对于同一个API的声明和原因解释，无需重复添加。

如果您的项目未集成 `PrivacyInfo.xcprivacy`，您可将 `Demo` 中的 `PrivacyInfo.xcprivacy` 文件直接添加进您的代码工程中。

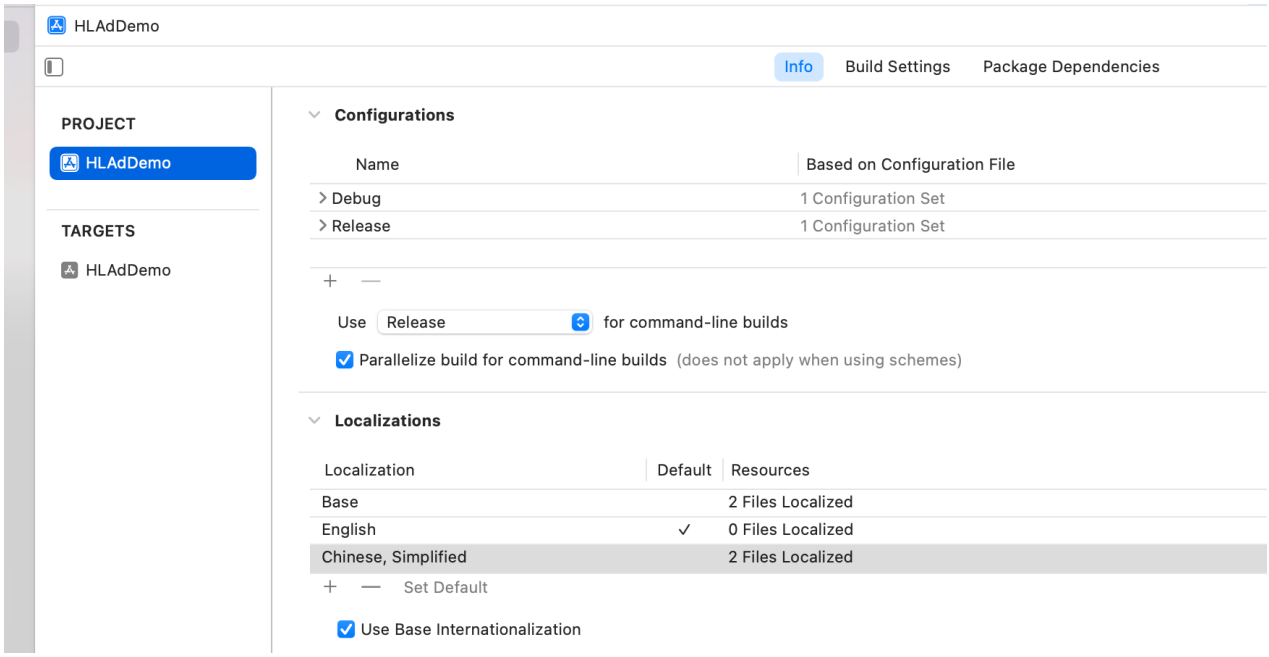
`PrivacyInfo.xcprivacy` 内容如下

```
    <key>NSPrivacyCollectedDataTypes</key>
        <array/>
    <key>NSPrivacyAccessedAPITypes</key>
        <array>
            <dict>
                <key>NSPrivacyAccessedAPIType</key>
                <string>NSPrivacyAccessedAPICategorySystemBootTime</string>
                <key>NSPrivacyAccessedAPITypeReasons</key>
                <array>
                    <string>35F9.1</string>
                </array>
            </dict>
            <dict>
                <key>NSPrivacyAccessedAPIType</key>
                <string>NSPrivacyAccessedAPICategoryFileTimestamp</string>
                <key>NSPrivacyAccessedAPITypeReasons</key>
                <array>
                    <string>C617.1</string>
                </array>
            </dict>
            <dict>
                <key>NSPrivacyAccessedAPIType</key>
                <string>NSPrivacyAccessedAPICategoryDiskSpace</string>
                <key>NSPrivacyAccessedAPITypeReasons</key>
                <array>
                    <string>7D9E.1</string>
                    <string>E174.1</string>
                </array>
            </dict>
            <dict>
                <key>NSPrivacyAccessedAPIType</key>
                <string>NSPrivacyAccessedAPICategoryUserDefaults</string>
                <key>NSPrivacyAccessedAPITypeReasons</key>
                <array>
                    <string>CA92.1</string>
                </array>
            </dict>
       </array>
```

添加语言设置

注意：开发者必须在这里设置所支持的语言，否则可能会有语言显示的问题。

例如：支持中文 添加 Chinese

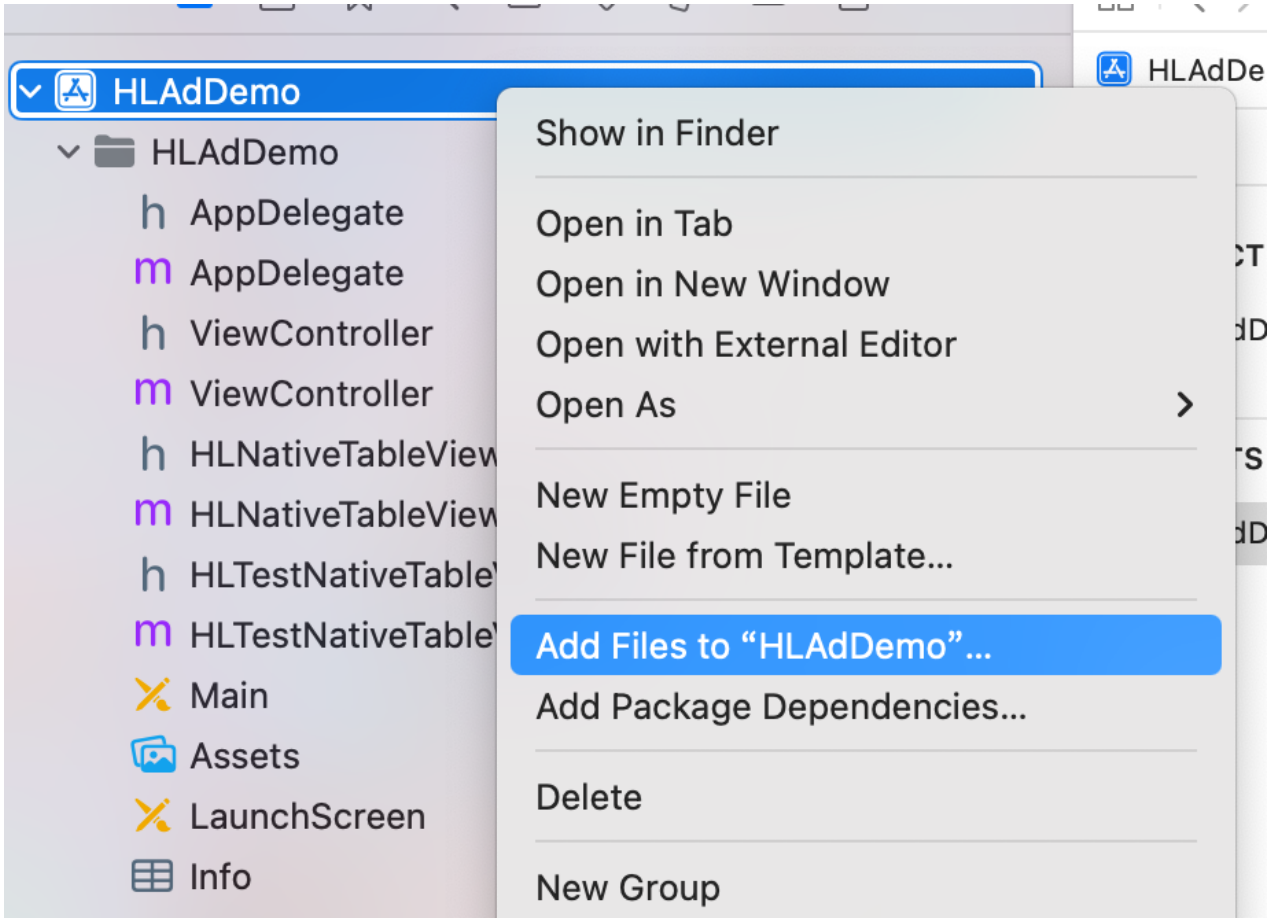


4. SDK集成

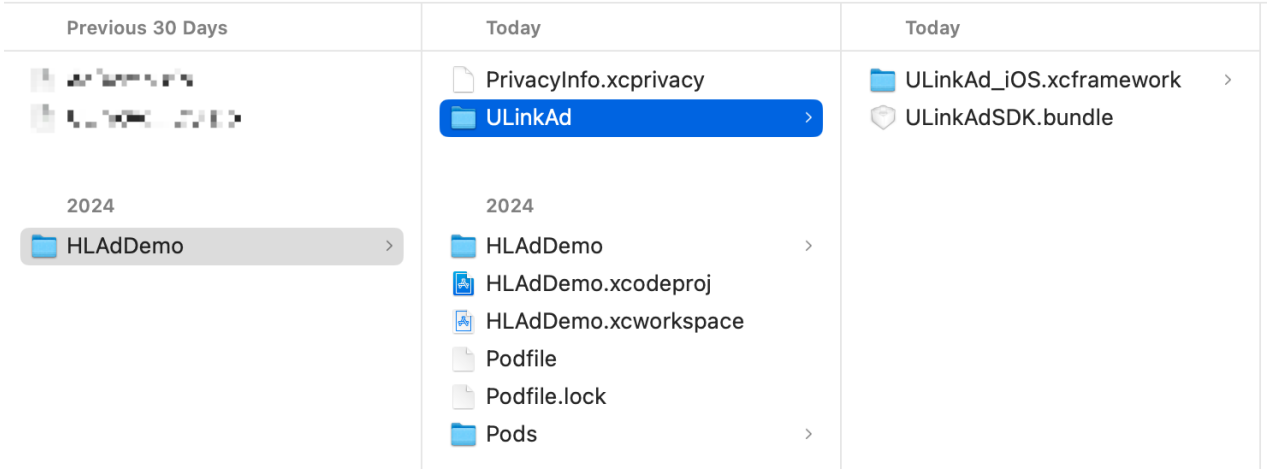
手动导入

添加ULinkAd到您的工程

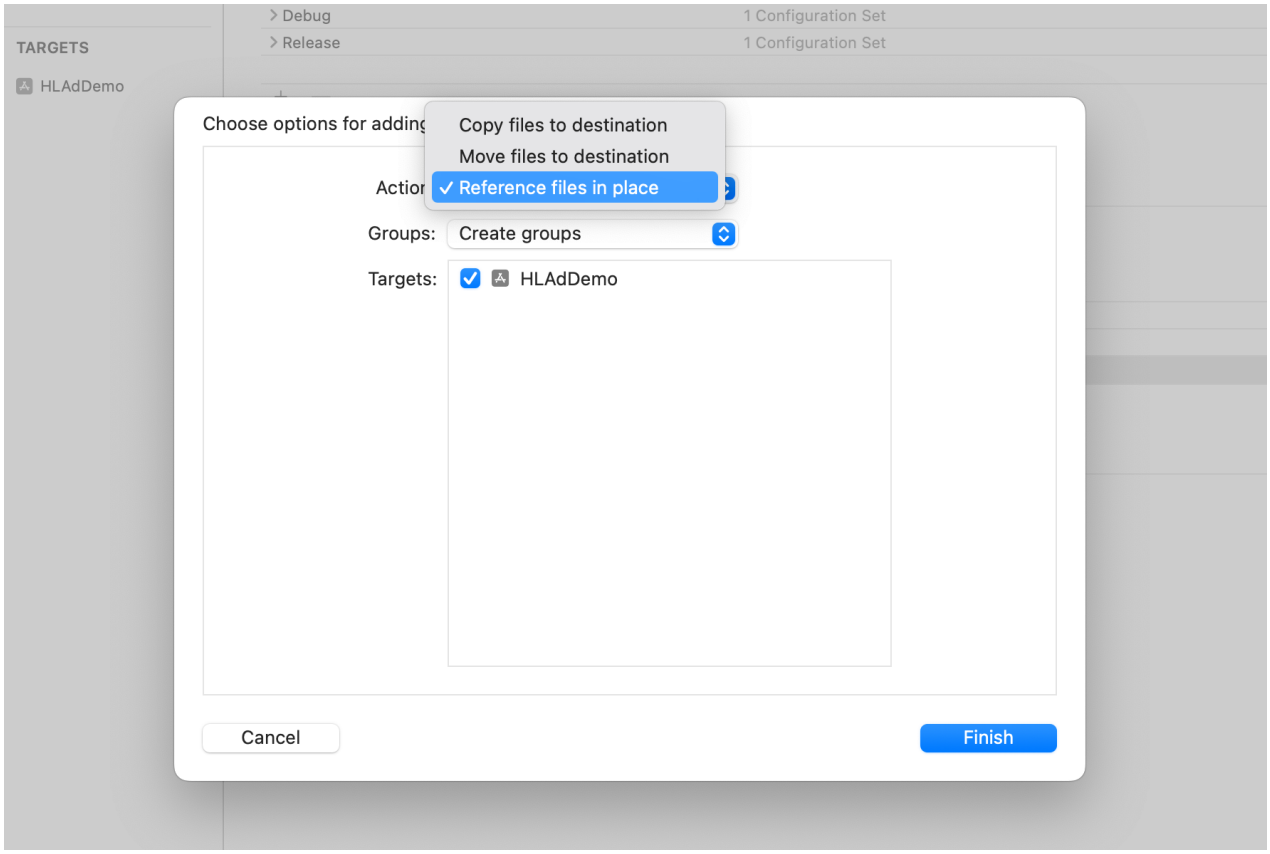
右键点击，选中 `Add Files to` 您的工程



选中 `ULinkAd`



SDK文件如果已在工程的本地路径内时， `Action` 选择 `Reference files to in place` ，否则 `Action` 选择 `Copy files to destination` 。最后勾选 `Targets` ，点击 `Finish`



5. 代码集成

初始化及绑定AppID

为了保证SDK正常使用，建议在 `AppDelegate.m` 的 `application:didFinishLaunchingWithOptions:` 方法中进行初始化

```
#import <HLAds/HMAdsSDKExtension.h>

@interface AppDelegate ()
@end

@implementation AppDelegate

 - (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(N
   //初始化
   [HMAdsSDK initWithAppId:@"YOUR_APP_ID"];
   NSLog(@"sdk版本号: %@", [HMAdsSDK sdkVersion]);
   // 启动SDK
   [HMAdsSDK startWithCompletionHandler:^(BOOL success, NSError * _Nonnull error)

   }];
 }

@end
```

SDK开启Log

```
//是否开启Log，默认关闭
[HMAdsSDK setLogLevel:HMAdsSDKLogDebug];
```

广告形式

开屏Splash

初始化及请求开屏

为了保证开屏的展示，我们推荐尽量在 App 启动时开始执行下面的方法。
例如：在 `AppDelegate.m` 的 `application:didFinishLaunchingWithOptions:` 方法中

```
#import <HLAds/HMAdsSDKExtension.h>

@interface AppDelegate () <HMAdsSplashAdDelegate>
 @property (nonatomic, strong) HMAdsSplashAd *splash;
@end

@implementation AppDelegate

  - (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(
    //SDK初始化
    [HMAdsSDK initWithAppID:@"YOUR_APP_ID"];

    //传入开屏广告尺寸adSize
    self.splash = [[HMAdsSplashAd alloc] initHMAdsSplashAdWithPlacementId:@"YOUR_S
    self.splash.splashWindow = self.window;
    self.splash.delegate = self;
    [self.splash loadAdData];
    return YES;
  }

@end
```

设置开屏展示时的window

```
  self.splash.splashWindow = self.window;
```

设置开屏加载时的背景图片

```
  // 设置开屏的背景图片，非强制
  self.splash.launchImageName = @"LaunchImageName";
```

展示开屏

```
 [self.splash showSplashAd];
```

实现代理方法

```
- (void)splashHMAdsDidStartAd:(HMAdsSplashAd *)splashAd {
    NSLog(@"开屏开始请求");
}

- (void)splashHMAdsDidReceiveAd:(HMAdsSplashAd *)splashAd {
    NSLog(@"开屏加载成功：%@ 单价：%.f", splashAd, [self.splash adEcpm]);
}

- (void)splashHMAdsDidShowAd:(HMAdsSplashAd *)splashAd {{
    NSLog(@"开屏展示成功");
}

- (void)splashHMAds:(HMAdsSplashAd *)splashAd didFailWithError:(NSError *)error {
    NSLog(@"开屏失败：%@", error);
}

- (void)splashHMAdsClickAd:(HMAdsSplashAd *)splashAd {
    NSLog(@"开屏点击");
}

- (void)splashHMAdsClosedAd:(HMAdsSplashAd *)splashAd {
    NSLog(@"开屏关闭");
}

- (void)splashHMAdDidClosedWebBrower:(HMAdsSplashAd *)splashAd {
    NSLog(@"开屏落地页关闭");
}
```

信息流Feed

初始化及请求信息流

```
#import <HLAds/HMAdsSDKExtension.h>

@interface ViewController ()<HMAdsFeedAdDelegate>
@property (nonatomic, strong) HMAdsFeedAd *feed;
@end

@implementation ViewController

- (void)viewDidLoad {
  [super viewDidLoad];
  // 传入广告尺寸adSize
   self.feed = [[HMAdsFeedAd alloc] initHMAdsFeedAdWithPlacementId:@"YOUR_SOLT_ID"
   self.feed.rootViewController = self;
   self.feed.delegate = self;
   [self.feed loadAdData];
}
@end
```

设置信息流广告的视图控制器

```
   self.feed.rootViewController = self;
```

实现代理方法

```
- (void)feedAdHMAdsDidStartAd:(HMAdsFeedAd *)feedAd {
    NSLog(@"信息流开始请求");
}

- (void)feedAdHMAds:(HMAdsFeedAd *)feedAd didReceiveAd:(UIView *)feedView {
    NSLog(@"信息流加载成功：%@ 单价：%.f",feedView, [self.feed adEcpm]);
}

- (void)feedAdHMAds:(HMAdsFeedAd *)feedAd didRenderSuccessAd:(UIView *)feedView {
    NSLog(@"信息流渲染成功:%@",feedView);
}

- (void)feedAdHMAdsDidShowAd:(HMAdsFeedAd *)feedAd view:(UIView *)view {
    NSLog(@"信息流展示:%@",view);
}

- (void)feedAdHMAds:(HMAdsFeedAd *)feedAd didFailWithError:(NSError *)error {
    NSLog(@"信息流失败:%@", error);
}

- (void)feedAdHMAdsClickAd:(HMAdsFeedAd *)feedAd view:(UIView *)view {
    NSLog(@"信息流点击:%@",view);
}

- (void)feedAdHMAdsClosedAd:(HMAdsFeedAd *)feedAd view:(UIView *)view {
    NSLog(@"信息流关闭");
}

- (void)feedAdHMAdDidClosedWebBrower:(HMAdsFeedAd *)feedAd {
    NSLog(@"信息流落地页关闭");
}
```

何时请求广告

展示信息流广告的应用，可以在实际展示广告之前随时请求这些广告。

- 注意：尽管预先获取广告是一种很好的方法，但务必不要长久保留旧广告而不进行展示。

何时加载完成

在应用调用 `loadAdData` 后，可通过 `HMAdsFeedAdDelegate` 中的以下方法获取请求的结果：

```
   // 广告加载成功
- (void)feedAdHMAds:(HMAdsFeedAd *)feedAd didReceiveAd:(UIView *)feedView;

    // 广告渲染成功
- (void)feedAdHMAds:(HMAdsFeedAd *)feedAd didRenderSuccessAd:(UIView *)feedView;

   // 广告请求失败
- (void)feedAdHMAds:(HMAdsFeedAd *)feedAd didFailWithError:(NSError *)error;
```

展示信息流广告

- 展示单条广告，可在收到广告渲染成功的回调 `- (void)feedAdHMAds:(HMAdsFeedAd *)feedAd didRenderSuccessAd:(UIView *)feedView;` 后，可以通过 `addSubView` 添加广告视图 `feedView` 来进行展示广告
- 在列表中展示多条广告，可参考Demo中的 `HLNativeTableViewController.m`

自适应高度

当传入广告尺寸adSize的高度为0时，可在收到广告加载成功后 `- (void)feedAdHMAds:(HMAdsFeedAd *)feedAd didReceiveAd:(UIView *)feedView`，通过 `feedView.frame.size.height` 获取广告高度

横幅Banner

初始化及请求横幅

```
#import <HLAds/HMAdsSDKExtension.h>

@interface ViewController ()<HMAdsBannerAdDelegate>
@property (nonatomic, strong) HMAdsBannerAd *banner;
@end

@implementation ViewController

- (void)viewDidLoad {
  [super viewDidLoad];
  //传入广告尺寸adSize，高度不能为0
  self.banner = [[HMAdsBannerAd alloc] initHMAdsBannerAdWithPlacementId:@"YOUR_SOI
  self.banner.rootViewController = self;
  self.banner.delegate = self;
  [self.banner loadAdData];
}
@end
```

设置横幅广告的视图控制器

```
  //传参self为视图控制器
  self.banner.rootViewController = self;
```

实现代理方法

```
- (void)bannerHMAdsDidStartAd:(HMAdsBannerAd *)bannerAd {
    NSLog(@"横幅开始请求");
}

- (void)bannerHMAds:(HMAdsBannerAd *)bannerAd didReceiveAd:(UIView *)view {
    NSLog(@"横幅加载成功：%@ 单价：%.f", bannerAd, [self.banner adEcpm]);
}

- (void)bannerHMAdsDidShowAd:(HMAdsBannerAd *)bannerAd view:(nonnull UIView *)view
    NSLog(@"横幅展示成功");
}

- (void)bannerHMAds:(HMAdsBannerAd *)bannerAd didFailWithError:(NSError *)error {
    NSLog(@"横幅失败：%@",error);
}

- (void)bannerHMAdsClickAd:(HMAdsBannerAd *)bannerAd view:(nonnull UIView *)view {
    NSLog(@"横幅点击");
}

- (void)bannerHMAdsClosedAd:(HMAdsBannerAd *)bannerAd {
    NSLog(@"横幅关闭");
    // 如果关闭时不再需要轮播展示，需要将self.banner置为nil
    self.banner = nil;
}

- (void)bannerHMAdDidClosedWebBrower:(nonnull HMAdsBannerAd *)bannerAd {
    NSLog(@"横幅落地页关闭:%@", bannerAd);
}
```

插屏Interstital

初始化及请求插屏

```
#import <HLAds/HMAdsSDKExtension.h>

@interface ViewController ()<HMAdsInterstitialAdDelegate>
@property (nonatomic, strong) HMAdsInterstitialAd *interstital;
@end

@implementation ViewController

- (void)viewDidLoad {
  [super viewDidLoad];
  //插屏高度不能为0
  self.interstital = [[HMAdsInterstitialAd alloc] initHMAdsInterstitialPlacementId
  self.interstital.delegate = self;
  [self.interstital loadAdData];
}
@end
```

展示插屏广告

```
  //传参self为视图控制器
    [self.interstital showInterstitialAdWithViewController:self];
```

实现代理方法

```
- (void)interstitialHMAdsDidStartAd:(HMAdsInterstitialAd *)interstitialAd {
    NSLog(@"插屏开始请求");
}

- (void)interstitialHMAdsDidReceiveAd:(HMAdsInterstitialAd *)interstitialAd {
NSLog(@"插屏加载成功：%@ 单价：%.f", interstitialAd, [self.interstital adEcpm]);
}

- (void)interstitialHMAdsDidShowAd:(HMAdsInterstitialAd *)interstitialAd {
    NSLog(@"插屏展示成功");
}

- (void)interstitialHMAds:(HMAdsInterstitialAd *)interstitialAd didFailWithError:(
    NSLog(@"插屏加载失败");
}

- (void)interstitialHMAdsClickAd:(HMAdsInterstitialAd *)interstitialAd {
    NSLog(@"插屏点击");
}

- (void)interstitialHMAdsClosedAd:(HMAdsInterstitialAd *)interstitialAd {
    NSLog(@"插屏关闭");
}

- (void)interstitialHMAdDidClosedWebBrower:(nonnull HMAdsInterstitialAd *)intersti
    NSLog(@"插屏落地页关闭：%@", interstitialAd);
}
```

激励视频RewardVideo

初始化及请求激励视频

```
#import <HLAds/HMAdsSDKExtension.h>

@interface ViewController ()<HMAdsRewardVideoAdDelegate>
@property (nonatomic, strong) HMAdsRewardVideoAd *rewardVideo;
@end

@implementation ViewController

- (void)viewDidLoad {
  [super viewDidLoad];
  self.rewardVideo = [[HMAdsRewardVideoAd alloc] initHMAdsRewardVideoPlacementId:@
  self.rewardVideo.delegate = self;
  [self.rewardVideo loadAdData];
}
@end
```

展示激励视频广告

```
  //传参self为视图控制器
   [self.rewardVideo showRewardVideoAdWithViewController:self];
```

实现代理方法

```
- (void)rewardVideoHMAdsDidStartAd:(HMAdsRewardVideoAd *)rewardVideoAd {
    NSLog(@"激励视频请求成功");
}

- (void)rewardVideoHMAdsDidReceiveAd:(HMAdsRewardVideoAd *)rewardVideoAd {
    NSLog(@"激励视频加载成功：%@ 单价：%.f", rewardVideoAd, [self.rewardVideo adEcpm])
}

- (void)rewardVideoHMAdsDidShowAd:(HMAdsRewardVideoAd *)rewardVideoAd {
    NSLog(@"激励视频展示成功");
}

- (void)rewardVideoHMAdsDidRewardEffective:(HMAdsRewardVideoAd *)rewardVideoAd {
    NSLog(@"激励视频获取奖励");
}

- (void)rewardVideoHMAdsDidPlayFinish:(HMAdsRewardVideoAd *)rewardVideoAd {
    NSLog(@"激励视频播放完成");
}

- (void)rewardVideoHMAds:(HMAdsRewardVideoAd *)rewardVideoAd didFailWithError:(NSE
    NSLog(@"激励视频加载失败：%@", error);
}

- (void)rewardVideoHMAdsClickAd:(HMAdsRewardVideoAd *)rewardVideoAd {
    NSLog(@"激励视频点击");
}

- (void)rewardVideoHMAdsClosedAd:(HMAdsRewardVideoAd *)rewardVideoAd {
    NSLog(@"激励视频关闭");
}
```

错误码

错误码	描述
5001	配置信息错误
5002	请求超时
5003	请求失败
5004	请求参数错误
5005	广告暂无填充
5006	广告渲染失败