智能营销聚合SDK接入文档

ANDROID

war环境要求

- Android API 21+
- JDK 1.8

SDK 集成

项目的build.gradle文件中添加以下依赖:

dependencies {

implementation (name: 'noah-6.3.6', ext:'aar')//汇川广告SDK

//如果需要引入tanx预算，请添加以下Tanx SDK依赖

implementation ('com.tanx:TanxUISDK:3.4.2')

//如果需要聚合广点通广告sdk，请添加此依赖，否则可以跳过

implementation (name: 'gdt-4.502.1372', ext:'aar')

//如果需要聚合穿山甲广告sdk，请添加此依赖，否则可以跳过

implementation (name: 'pangolin-5.0.1.4', ext:'aar')

//如果需要聚合快手广告sdk，请添加此依赖，否则可以跳过

implementation (name: 'kuaishou-3.3.4.0', ext:'aar')

}

如果需要集成广点通广告SDK，需要在你的AndroidManifest.xml文件中添加以下声明:

JSON

```
<provider
  android:name="androidx.core.content.FileProvider"
  android:authorities="${applicationId}.gdt.fileprovider"
  android:exported="false"
  android:grantUriPermissions="true">
  <meta-data
    android:name="android.support.FILE_PROVIDER_PATHS"
    android:resource="@xml/gdt_file_path"/>
</provider>
```

如果需要集成穿山甲广告SDK，需要在你的AndroidManifest.xml文件中添加以下声明:

```java
<provider
    android:name="com.bytedance.sdk.openadsdk.TTFileProvider"
    android:authorities="${applicationId}.TTFileProvider"
    android:exported="false"
    android:grantUriPermissions="true">
    <meta-data
         android:name="android.support.FILE_PROVIDER_PATHS"
         android:resource="@xml/pg_file_paths" />
</provider>
<provider
     android:name="com.bytedance.sdk.openadsdk.multipro.TTMultiProvider"
     android:authorities="${applicationId}.TTMultiProvider"
     android:exported="false" />
```

SDK 初始化

```java
NoahSdkConfig sdkConfig = new NoahSdkConfig.Builder()
            .setAppKey(APP_KEY)
            .setOuterSettings(new NoahSdkConfig.NoahOuterSettings(){
             /**
               * 强烈建议返回oaid，缺失oaid会严重影响广告效果
               * 如果是荣耀手机同时有华为的oaid和荣耀的oaid的情况，这个接口返回华为的Oaid，下面的getOAID2接口返回荣耀的oaid
               */
                @Override
                public String getOAID() {
                    return "oaid"
                }
              /**
               * 如果是荣耀手机同时有华为的oaid和荣耀的oaid的情况，这个接口返回荣耀的Oaid，上面的getOAID接口返回华为的oaid,只有一个oaid的设备，getOAID2接口可返回空
               */
                @Override
                public String getOAID2() {
                    return "oaid2"
                }
             })
            .build();
GlobalConfig globalConfig = new GlobalConfig.Builder().build();
NoahSdk.init(getApplication(), sdkConfig, globalConfig);
```

原生广告(信息流广告)

1. 获取广告

Java

```java
RequestInfo requestInfo = new RequestInfo();
NativeAd.getAd(NativeAdActivity.this, SLOT, requetsInfo, new NativeAd.AdListener() {
    @Override
    public void onAdError(final AdError adError) {
        //广告返回失败
    }
    @Override
    public void onAdLoaded(final NativeAd ad) {
        //广告返回成功，展示SDK模板广告
        View sdkView = ad.getView();
        if (sdkView != null) {
          LinearLayout container = findViewById(R.id.nativead_container);
          container.removeAllViews();
          container.addView(sdkView, new LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT, LinearLayout.LayoutParams.WRAP_CONTENT));
        }
    }
    @Override
    public void onAdLoaded(final List<NativeAd> ads) {
         //如果要求SDK一次返回多条广告，SDK会回调这个接口
    }
    @Override
    public void onAdShown(final NativeAd ad) {
        //广告曝光成功
    }
    @Override
    public void onAdClosed(final NativeAd ad) {
        //广告关闭，只有使用SDK渲染方式才会有这个回调
    }
    @Override
    public void onAdClicked(final NativeAd ad) {
        //广告点击
    }
    @Override
    public void onDownloadStatusChanged(final NativeAd ad, final int apkDownloadStatus) {
       //下载类广告下载进度回调
       printLog("onDownloadStatusChanged, return status: " + apkDownloadStatus + "  getApkDownloadStatus: " + mNativeAd.getApkDownloadStatus());
    }
    @Override
    public void onAdEvent(final NativeAd ad, final int eventId, final Object extInfo) {
       //其他广告事件回调
```

```
        }
    });
```

3. 展示广告

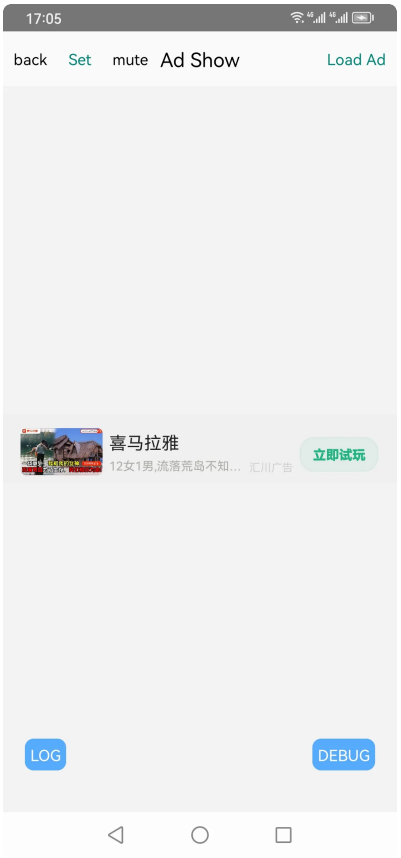
SDK支持模板渲染和自渲染两种渲染方式

模板渲染

- 模板渲染原生广告样式示例

电视直播

上图下文直播



左图右文

上图下文

电视卡片

上图下文气泡

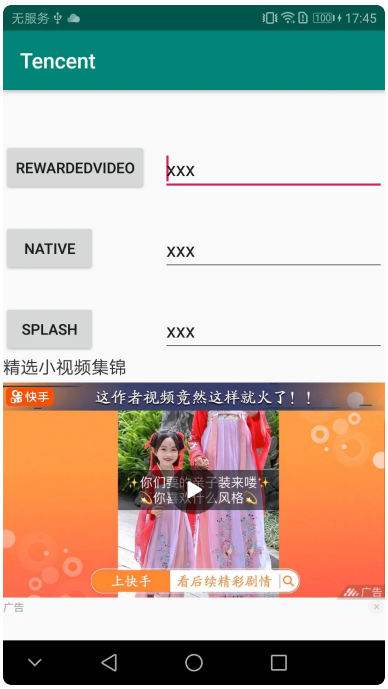
- 模板渲染广告展示方法

```java
@Override
public void onAdLoaded(final NativeAd ad)
    View sdkView = ad.getView();
    if (sdkView != null) {
        LinearLayout container = findViewById(R.id.nativead_container);
        container.removeAllViews();
        container.addView(sdkView, new LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT, LinearLayout.LayoutParams.WRAP_CONTENT));
    }
}
```

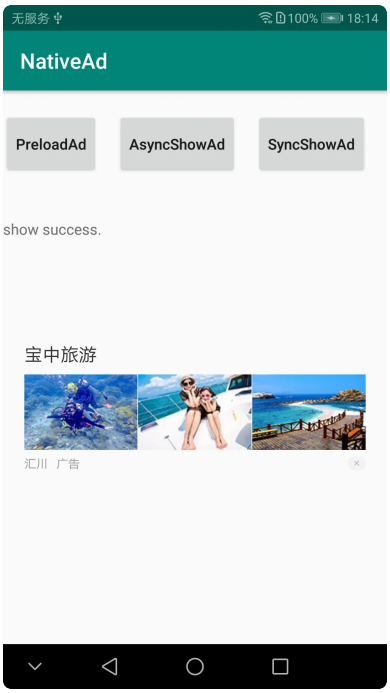
自渲染

- 自渲染原生广告样式示例

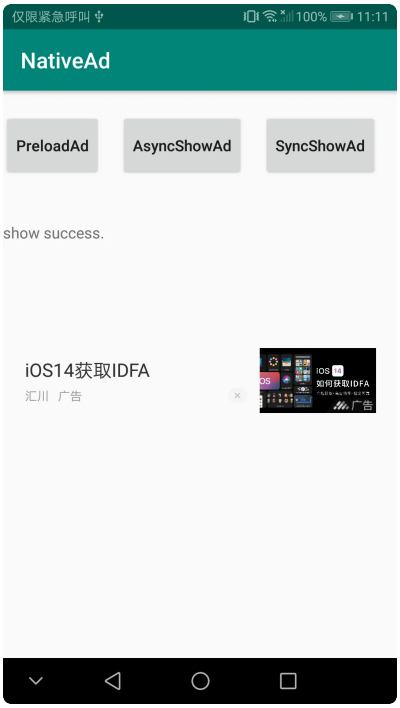
大图图文



大图视频



三图



小图图文

- 自渲染广告展示方法

a、调用NativeAd.getAdAssets()接口获取广告物料

接口名	返回类型	接口说明	能否为空
getTitle	String	广告标题，一般6-26个字符	能
getDescription	String	广告描述，一般6-26个字符	能
getCallToAction	String	广告创意区域内的点击按钮建议文本，诸如立即下载、立即查看等	能
getCreateType	int	广告样式，上文所述	否
getAdLogo	Bitmap	对应ADN的Logo，比如穿山甲的Logo、广点通的Logo、弘顺的Logo等	能
isAppAd	boolean	广告是否是应用下载类广告	否
getExpiredTime	long	广告有效期，单位毫秒，一般45分钟内有效	否
getPrice	double	广告价格，单位分/ecpm	否

getIcon	SdkAssets.Image	广告Icon，可使用SDK的AdIconView展示；也可由媒体自渲染加载展示	能
getCover	SdkAssets.Image	对于单图类型广告(大图、小图)返回广告主图片；对于三图类型广告返回第一张图片	否
getCovers	List<SdkAssets.Image>	对于单图类型广告(大图、小图)返回广告主图片队列，队列包含一张主图片；对于三图类型广告返回全部三张图片	否
getAssetId	String	广告Id	能
getSource	String	广告来源，诸如飘柔、联合利华等	能
getSlotKey	String	请求广告使用的slotId	否
isVideo	boolean	是否是视频广告	否

b、调用NativeAssets.getCreateType()获取广告尺寸

Java

```java
int createType = assets.getCreateType();
switch (createType) {
     case AdStyleConstant.CREATE_TYPE_HOR_LARGE:
          return "横向大图，宽高比16:9";
     case AdStyleConstant.CREATE_TYPE_HOR_SMALL:
          return "横向小图，宽高比3:2";
     case AdStyleConstant.CREATE_TYPE_GROUP:
          return "三图，每张图宽高比3:2";
     case AdStyleConstant.CREATE_TYPE_HOR_VIDEO:
          return "横向视频，宽高比16:9";
     case AdStyleConstant.CREATE_TYPE_VER_LARGE:
          return "纵向大图，宽高比9:16";
     case AdStyleConstant.CREATE_TYPE_VER_VIDEO:
          return "纵向视频，宽高比9:16";
     default:
           break;
}
```

c、构建广告渲染视图，添加到媒体自定义的容器中。媒体不需要关注视频/图片区域的渲染(由SDK提供的MediaView实现)，媒体只需要把MediaView添加到自定义容器中即可

Java

```java
private void showAd(NativeAd ad) {
        FrameLayout container = findViewById(R.id.main_content);
        container.removeAllViews();
        //需要使用SDK根组件NativeAdView
        final NativeAdView nativeAdView = new NativeAdView(this);
        FrameLayout.LayoutParams params = new FrameLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, ViewGroup.LayoutParams.WRAP_CONTENT);
        params.gravity = Gravity.CENTER_HORIZONTAL;
        container.addView(nativeAdView, params);
        //根据媒体自身设计，构建自定义View
        View customView = LayoutInflater.from(this).inflate(R.layout.sdk_native_default_layout, null);
        //为NativeAdView绑定自定义组件
        nativeAdView.setCustomView(customView);
        //为NativeAdView绑定NativeAd
        nativeAdView.setNativeAd(ad);
        nativeAdView.setVisibility(View.VISIBLE);

        Button cta = customView.findViewById(R.id.native_ad_call_to_action);
        TextView title = customView.findViewById(R.id.native_ad_title);
        TextView content = customView.findViewById(R.id.native_ad_content);
        AdIconView iconView = customView.findViewById(R.id.native_ad_icon);
        MediaView coverView = customView.findViewById(R.id.native_ad_media_view);

        ImageView closeView = customView.findViewById(R.id.native_ad_close);
        if (closeView != null) {
            closeView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    nativeAdView.setVisibility(View.GONE);
                }
            });
        }

        NativeAd.NativeAssets assets = ad.getAdAssets();
        if (assets != null) {
            cta.setText(assets.getCallToAction());
            title.setText(assets.getTitle());
            content.setText(assets.getDescription());
            //为AdIconView绑定自定义组件,也可以使用自定义View自行加载icon
```

```
            iconView.setNativeAd(ad);
            //为MediaView绑定自定义组件
            coverView.setNativeAd(ad);

            List<View> clickList = new ArrayList<>();
            clickList.add(cta);
            clickList.add(title);
            clickList.add(content);
            clickList.add(iconView);
            clickList.add(coverView);
            //广告渲染视图注册广告点击事件，响应广告点击
            ad.registerViewForInteraction(nativeAdView,
                                          clickList,
                                          clickList);
        }
    }
```

d、为广告渲染视图注册广告点击事件，响应广告点击。针对下载类广告，SDK支持媒体在合规(比如已展示五要素)条件下，传入直接下载组件列表。用户点击后直接触发下载行为，不会再弹出下载确认对话框

Java

```
//未传入直接下载View列表并注册广告点击事件
List<View> clickList = new ArrayList<>();
clickList.add(cta);
clickList.add(title);
clickList.add(content);
clickList.add(iconView);
clickList.add(coverView);
ad.registerViewForInteraction(nativeAdView, clickList, clickList);

//传入直接下载View列表并注册广告点击事件
List<View> clickList = new ArrayList<>();
clickList.add(title);
clickList.add(content);
clickList.add(iconView);
clickList.add(coverView);
List<View> directDownloadList = new ArrayList<>();
directDownloadList.add(cta);
ad.registerViewForInteraction(nativeAdView, clickList, clickList, directDownloadList);
```

4. 销毁广告

- 当广告使用完毕不再需要时，请调用 destroy() 接口

```java
//销毁广告
ad.destroy();
//销毁MediaView
mediaView.destroy();
```

---

5. 下载类广告六要素展示

- 获取六要素信息

```java
//原生、激励视频、横幅、闪屏广告的接口形式一致
NativeAd.getAd(NativeAdActivity.this, slot, new NativeAd.AdListener() {
    @Override
    public void onAdLoaded(final NativeAd ad) {
          //获取六要素展示
          ad.fetchDownloadApkInfo(new IFetchDownloadApkInfoCallback() {
                @Override
                public void onFinish(@Nullable final DownloadApkInfo downloadApkInfo) {
                    //应用名称
                    String appName = downloadApkInfo.appName;
                    //发行商名称
                    String authorName = downloadApkInfo.authorName;
                    //应用版本
                    String versionName = downloadApkInfo.versionName;
                    //应用所用权限信息URL
                    String permissionUrl = downloadApkInfo.permissionUrl;
                    //隐私权限声明URL
                    String privacyAgreementUrl = downloadApkInfo.privacyAgreementUrl;
                    //产品功能介绍
                    String functionDescUrl = downloadApkInfo.functionDescUrl;
                }
          });
     }
});
```

- 按照工信部要求，所有下载类广告下载前都需要弹出一个确认对话框展示要求的信息给到用户，由用户决定是否下载这个APK。SDK内部已经实现了这个要求，但是如果媒体有强烈的对话框样式适配需求，可以按此流程实现自定义样式的对话框样式适配，以原生广告举例:

Java

```java
//原生、激励视频、横幅、闪屏广告的接口形式一致，开发者可自行修改
RequestInfo requestInfo = new RequestInfo();
requestInfo.needDownloadConfirm = true;
NativeAd.getAd(NativeAdActivity.this, slot, requestInfo, new NativeAd.AdListener() {
    @Override
    public void onAdLoaded(final NativeAd ad) {
         IDownloadConfirmListener listener = new IDownloadConfirmListener() {
                @Override
                public void onDownloadConfirm(final Context context, final IDownloadConfirmCallBack callBack) {
                    //SDK会在弹窗时机回调这个方法，媒体获取相关信息
                    ad.fetchDownloadApkInfo(new IFetchDownloadApkInfoCallback() {
                        @Override
                        public void onFinish(@Nullable final DownloadApkInfo downloadApkInfo) {
                            //获取相关信息后，自行展示自定义对话框
                            //自定义对话框内部，注意回调通知SDK用户是否同意下载
                            if (downloadApkInfo != null) {
                                new DownloadApkConfirmDialog(context, ad, callBack).show();
                            }
                        }
                    });
                }
            };
            ad.setDownloadConfirmListener(listener);
    }
});
```

激励视频广告

1. 获取广告

```java
RewardedVideoAd.getAd(RewardedVideoAdActivity.this, SLOT, new RewardedVideoAd.AdListener() {
    @Override
    public void onAdError(final AdError adError) {
        //广告返回失败
    }
    @Override
    public void onAdLoaded(final RewardedVideoAd ad) {
        //广告请求成功，展示广告
        ad.show();
    }
    @Override
    public void onAdShown(final RewardedVideoAd ad) {
        //广告曝光
    }
    @Override
    public void onAdClosed(final RewardedVideoAd ad) {
        //广告关闭
    }
    @Override
    public void onAdClicked(final RewardedVideoAd ad) {
        //广告点击
    }
    @Override
    public void onVideoStart(final RewardedVideoAd ad) {
        //激励视频开始播放
    }
    @Override
    public void onVideoEnd(final RewardedVideoAd ad) {
        //激励视频播放完成
    }
    @Override
    public void onRewarded(final RewardedVideoAd ad) {
        //激励发放
    }
});
```

2. 展示广告

激励视频广告样式


```java
@Override
public void onAdLoaded(final RewardedVideoAd ad) {
    ad.show();
}
```

3. 销毁广告

- 当广告使用完毕不再需要时，请调用 destroy() 接口：

```java
ad.destroy();
```

4.异步查奖（版本8.6新增）

淘宝唤端激励浏览类的广告，不能确保同步发奖，需要结合场景在合适的时机进行历史奖励的查询

- 查奖接口

```java
/**
 * 查询历史奖励
 * @param context
 * @param slotKey, 广告位id
 * @param requestInfo, 请求参数，userId必填
 * @param callback, 查奖结果回调
 */
public static void queryRewards(Context context, String slotKey, RequestInfo requestInfo, IRewardsQueryCallback callback)
```

接口	参数	子参数	说明	是否必填
queryRewards	Context context	-	Android Context	是
	String slotKey		广告位ID	是
	RequestInfo requestInfo	userId	用户ID	是

- 查奖结果回调

```java
/**
 * 激励视频查询回调
 * date: 2024/3/1
 * author: gorky.zhy@alibaba-inc.com
 */
public interface IRewardsQueryCallback {
    int CODE_INTERNAL_TIMEOUT = -2; // 调用超时，目前tanx的接口有些情况是不会回调的，我们sdk内部弄个超时回调
    int CODE_INTERNAL_ERR = -1;     // sdk内部的错误
    int CODE_REWARD = 0;            // 查询到奖励
    int CODE_NO_REWARD = 1;         // 未查询到奖励
    int CODE_REQ_ERR = 2;           // 请求错误
    int CODE_NO_ADN = 3;            // 没有需要查询奖励的ADN

    void onResult(int code, int rewardType, @Nullable Map<String, Object> extra);
}
```

接口	参数	子参数	说明	是否必填
onResult	int code	–	回调结果 –2: 调用超时 –1: sdk内部错误 0: 查询到奖励 1: 未查询到奖励 2: 请求错误 3: 没有需要查询奖励的ADN	是
	int rewardType		奖历类型 –1：无效 0: 基础奖励 1: B奖（下单） 2：A奖+B奖（浏览+下单）	是
	Map<String, Object> extra	pid	奖励的广告位pid	否
		taskType	暂时可不关注	
		sessionId	奖励对应的某个广告的id	
		completeTime	任务完成时间	
		rewardName	媒体在平台配置的奖励名称	
		rewardCount	媒体在平台配置的奖励数量	

		<pre>1 "reward_data": {
		2</pre>
			支持多个奖励信息返回	

```
    "reward_success_id": "",
    "reward_info_list": [
        {
            "slot_key": "",
            "sid": "",
```

```
    "pid": "",
    "adn_id": "",
    "reward_time": "",
    "rewarded": "",
    "app_id": "",
```

```
12      "price": "",
13      "reward_type": ""
14    }
15  ],
16  "reward_type": ""
17 },
```

- 调用实例

异步查奖接口调用实例 Java

```java
String slot = SpUtils.getString(SdkConfig.KEY_VIDEO_SLOT, null);
if (StringUtils.isEmpty(slot)) {
    slot = SLOT_REWARD_VIDEO_KEY;
}

RequestInfo requestInfo = new RequestInfo();
requestInfo.userId = "adkfa";
RewardedVideoAd.queryRewards(MainActivity.this, slot, requestInfo, new IRewardsQueryCallback() {
    @Override
    public void onResult(int code, int rewardType, @Nullable Map<String, Object> extra) {
        ThreadManager.post(ThreadManager.THREAD_UI, new Runnable() {
            @Override
            public void run() {
                Toast.makeText(MainActivity.this, "query rewards result: " + code + " : " + rewardType, Toast.LENGTH_SHORT).show();
            }
        });
    }
});
```

5.是否为Tanx进阶广告

- 激励视频广告返回后，请调用 isTanxAdvancedAd() 接口：

Java

```java
ad.isTanxAdvancedAd();
```

开屏广告

1. 获取广告

```java
SplashAd.getAd(SplashAdActivity.this, SLOT, new SplashAd.AdListener() {
    @Override
    public void onAdError(final AdError adError) {
        //广告返回失败
    }
    @Override
    public void onAdLoaded(final SplashAd ad) {
        //广告返回成功，展示广告
        ad.showSplashAd(splashViewGroup);
    }
    @Override
    public void onAdShown(final SplashAd ad) {
        //广告曝光
    }
    @Override
    public void onAdSkip(final SplashAd ad) {
        //广告点击Skip按钮
    }
    @Override
    public void onAdClicked(final SplashAd ad) {
        //广告点击
    }
    @Override
    public void onAdTimeOver(final SplashAd splashAd) {
        //广告倒计时结束
    }
    @Override
    public void onInterceptClick(int interceptEventType, Map<String, String> eventData);
        //点击拦截事件对应的回调
    }
    @Override
    public void onSplashLpShow(boolean show);
        //闪屏落地页展示
    }
    @Override
    public void onAdExtraStat(int eventId, String arg1, Map<String, String> args);
        //附加打点
    }
    @Override
    public void onAdReward(@NonNull ISplashRewardListener callBack);
        //奖励式开屏回调
    }
});
```

2. 展示广告

开屏广告样式

```java
@Override
public void onAdLoaded(final SplashAd ad) {
    ad.showSplashAd(adContainer);
}
```

3. 销毁广告

- 当广告使用完毕不再需要时，请调用 destroy() 接口：

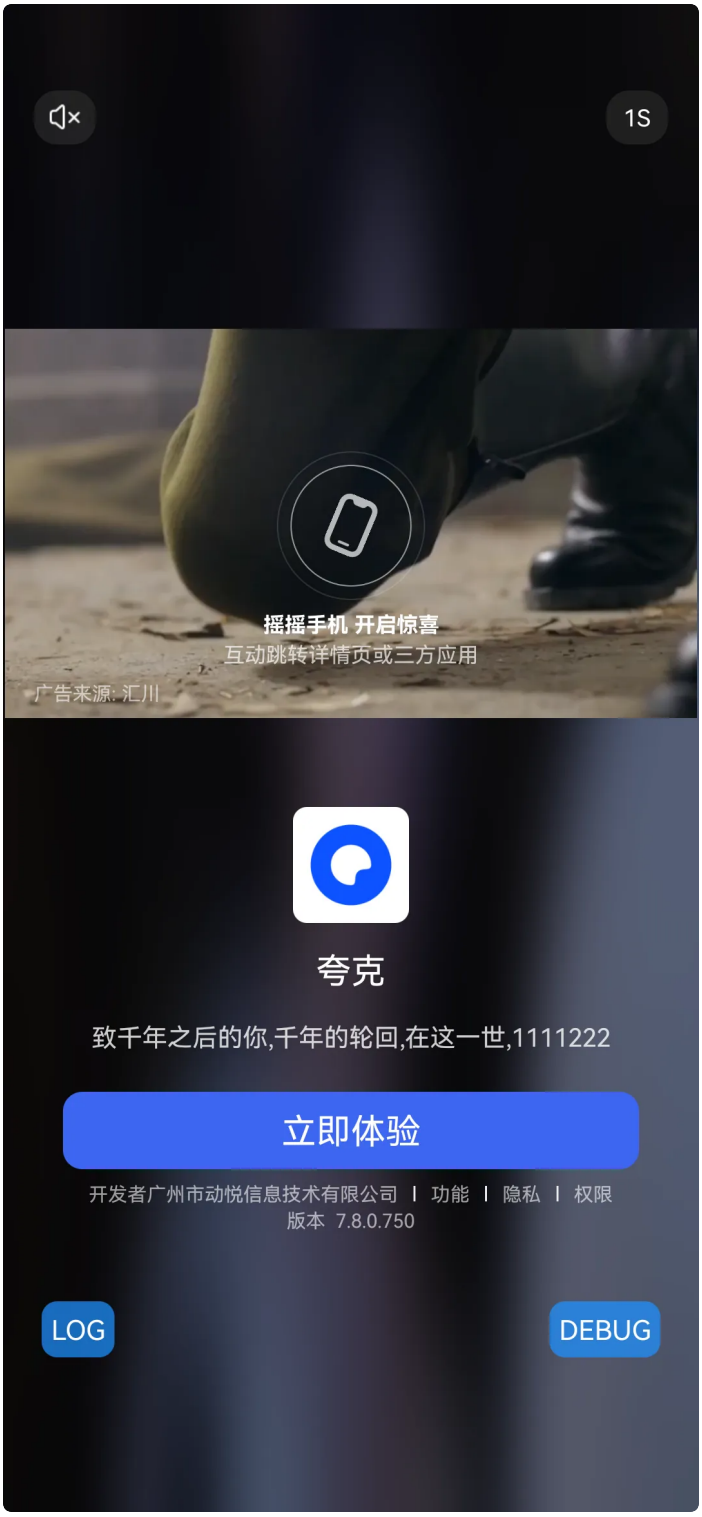
```java
ad.destroy();
```

插屏广告

1. 获取广告

```java
InterstitialAd.getAd(InterstitialAdActivity.this, SLOT, new InterstitialAd.AdListener() {
                    @Override
                    public void onAdError(final AdError adError) {
                        //广告返回失败
                    }

                    @Override
                    public void onAdLoaded(final InterstitialAd ad) {
                        //广告返回成功
                    }

                    @Override
                    public void onAdShown(final InterstitialAd ad) {
                        //广告曝光成功
                    }

                    @Override
                    public void onAdClosed(final InterstitialAd ad) {
                        //广告关闭
                    }

                    @Override
                    public void onAdClicked(final InterstitialAd ad) {
                        //广告点击
                    }

                    @Override
                    public void onVideoStart(final InterstitialAd ad) {
                        //广告视频播放开始
                    }

                    @Override
                    public void onVideoEnd(final InterstitialAd ad) {
                        //广告视频播放结束
                    }
                });
```

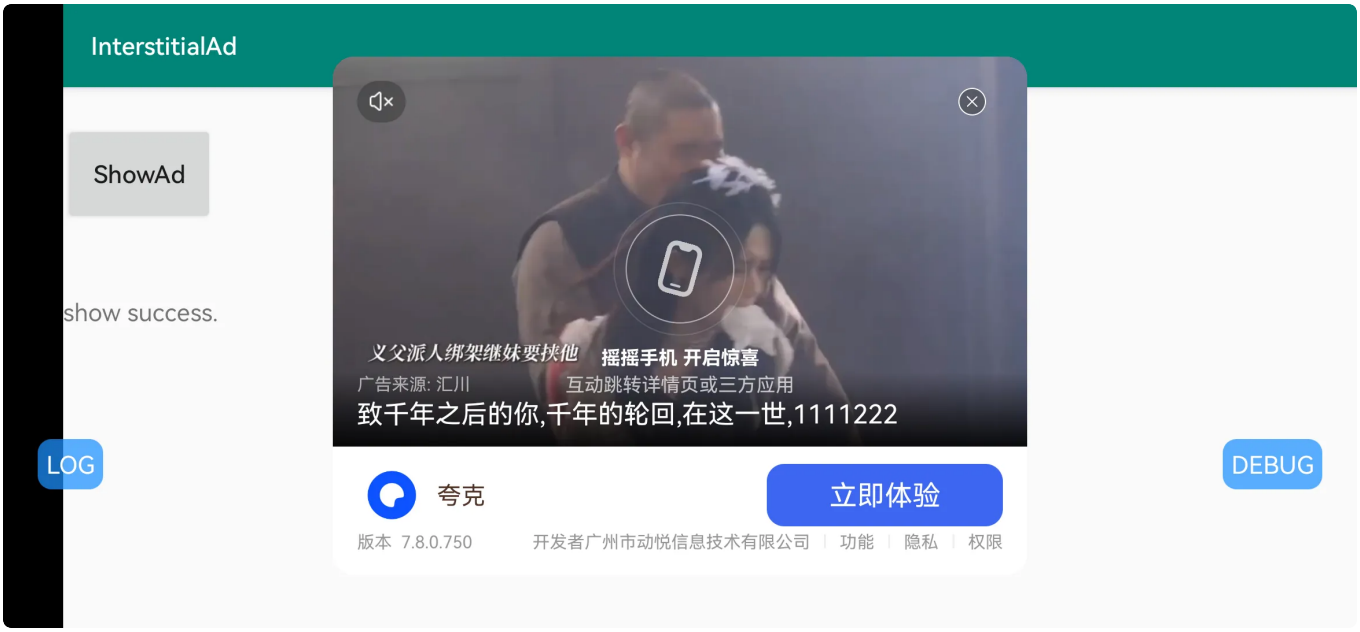
2. 展示广告



InterstitialAd

ShowAd


show
义父派人绑架继妹要挟他 摇摇手机 开启惊喜
互动跳转详情页或三方应用
广告来源: 汇川
夸克
致千年之后的你,千年的轮...
立即体验
开发者广州市动悦信息技术有限公司 | 功能 | 隐私 | 权限
版本 7.8.0.750
LOG
DEBUG


InterstitialAd
ShowAd
show success.
义父派人绑架继妹要挟他 摇摇手机 开启惊喜
广告来源: 汇川
互动跳转详情页或三方应用
致千年之后的你,千年的轮回,在这一世,1111222
夸克
立即体验
版本 7.8.0.750
开发者广州市动悦信息技术有限公司 | 功能 | 隐私 | 权限
LOG
DEBUG

```java
@Override
public void onAdLoaded(final InterstitialAd ad) {
    ad.show();
}
```

3. 销毁广告

- 当广告使用完毕不再需要时，请调用 destroy() 接口：

```java
ad.destroy();
```

与外部广告源竞价

重要：智能营销广告SDK与外部广告源竞价，竞价结束必须回传竞价胜出或失败事件，不回传竞价结果事件会影响计费：

目前我们支持以下2个方案

1）仅提供竞价胜负信息+原因

2）在1的基础上，回传所有参竞ADN信息（推荐使用：有利于汇川针对性的对媒体进行提价）

PS：媒体可以任选上述其中一个方案

- 方案1)
  - 使用sendWinNotification、sendLossNotification这2个接口
- 方案2)
  - 使用sendNotification接口完成所有信息的回传

sendWinNotification、sendLossNotification说明

```java
/**
 * 竞价胜出调用
 * @param price 如果是二价结算，传入媒体竞价的二价，如果是一价结算传入参与竞价的智能营销SDK广告的价格，单位：分/ecpm。
                                    媒体竞价二价的定义为：当智能营销SDK广告竞价胜出时，竞价队列中次高价ADN的出价 + 1分，例如次高ADN出价100分，二价就是101分。
                    媒体是采用一价结算还是二价结算由商务合同决定。
 */
public void sendWinNotification(int price) {
}

/**
 * 竞价失败调用
 * @param price 媒体竞价的最高价，单位：分/ecpm
                    建议如实传入媒体竞胜的最高价，有利于智能营销SDK广告的优化。如果不能传最高价，请传入参与竞价的智能营销SDK广告本身的价格
 * @param reason 竞价失败的原因 {@link BiddingLossReason}
 */
public void sendLossNotification(int price, int reason) {
}

或者
/**
 * 如果竞价结果需要回传ADN信息，可以使用以下接口，注意，使用以下接口回传竞价信息就不需要调用上述两个接口，使用方式可见 #各ADN竞价信息返回
 * @param isWin 竞价是否胜出
 * @param price 竞价的价格 当前Bid价格（分）
 * @param reason 竞价失败的原因
 * @param list 竞价信息
 */
 public void sendNotification(final boolean isWin, final int price, final int reason, final BiddingInfoList list) {
}
```

```java
public class BiddingLossReason {
    /**
    * 价格竞败
    */
    public static final int LOW_PRICE = 1;

    /**
    * 底价过滤
    */
    public static final int FLORR_PRICE = 2;

    /**
    * 广告超时返回
    */
    public static final int TIME_OUT = 3;

    /**
    * 广告频控
    */
    public static final int FREQUENCY_CNOTROL = 4;

    /**
    * 其他原因
    */
    public static final int OTHER = 5;
}
```

sendNotification说明

- setAdnName = 参加竞价ADN的名称（e.g 穿山甲=csj）
- setAdid = ADN本次参竞的物料ID（如无法获取可不传）
- setbBidType = RTB/PD
- setPlacementID = Pid
- setPirce = 对应的参竞价格（单位：分）
- setResult = 竞胜状态（win/loss）

```java
//ADN 1 相关信息回传
BiddingInfo biddingBean = BiddingInfo.newBuilder().setAdnName("huichuan")       // ADN名称
                                    .setAdId("adid123")                         // 胜出ADN的adid（如有）
                                    .setPrice(9)                                // ADN当前Bid价格（分）
                                    .setBidType("RT")                           // BidType
                                    .setPlacementId("placementId456")           // placementId
                                    .setResult(BiddingInfo.WIN)                 // ADN竞胜状态（胜/负）
                                    .build();
//ADN 2 相关信息回传
BiddingInfo biddingBean2 = BiddingInfo.newBuilder().setAdnName("gdt")
                                    .setAdId("adid1111")
                                    .setPrice(2)
                                    .setBidType("RT")
                                    .setResult(BiddingInfo.LOSS)
                                    .build();

//支持回传多个ADN信息
BiddingInfoList list = BiddingInfoList.newBuilder()
                                    .add(biddingBean)
                                    .add(biddingBean2)
                                    .build();
ad.sendNotification( b: false,  i: 2,  i1: 0, list);
```

```java
BiddingInfo biddingBean = BiddingInfo.newBuilder()
                                    .setAdnName("hc")
                                    .setPlacementId("placementId456")
                                    .setAdId("adid456")
                                    .setBidType("RT")
                                    .setPrice(9)
                                    .setResult(BiddingInfo.LOSS)
                                    .build();

BiddingInfoList list = BiddingInfoList.newBuilder()
                                    .add(biddingBean)
                                    .build();
ad.sendNotification( b: false,  i: 1, BiddingLossReason.LOW_PRICE, list);
```

获取价格接口 （单位：分）

Java

```java
ad.getPrice()
```

隐私接口代理

媒体可以设置隐私接口代理实现，接管SDK的隐私接口调用。媒体实现代理接口后，SDK不会再直接调用对应的系统接口，需要获取隐私信息时调用媒体实现的代理接口获取，媒体可以按照自身的合规要求，复写相关代理接口，返回空信息或者做调用频率控制。

Java

```java
        NoahSdkConfig config = new NoahSdkConfig.Builder()
                                    .setAppKey(APP_KEY)
                                    .setOuterSettings(new NoahSdkConfig.NoahOuterSettings() {
                                        /**
                                         * 强烈建议返回oaid，确实oaid会严重影响广告效果
                                         */
                                        @Override
                                        public String getOAID() {
                                            Log.d(TAG, "call getOAID.");
                                            return "oaid"; //强烈建议返回oaid，确实oaid会严重影响广告效果
                                        }

                                        /**
                                         * 媒体接管 TelephonyManager.getDeviceId()接口调用，如果媒体实现了这个接口，SDK不会再主动调用系统接口
                                         * @param telephonyManager
                                         * @return
                                         */
                                        @Override
                                        public String getDeviceId(final TelephonyManager telephonyManager) {
                                            Log.d(TAG, "call getDeviceId.");
                                            return "";
                                        }

                                        /**
                                         * 媒体接管 TelephonyManager.getDeviceId(int slotIndex)接口调用，如果媒体实现了这个接口，SDK不会再主动调用系统接口
                                         * @param telephonyManager
                                         * @param i
                                         * @return
                                         */
                                        @Override
                                        public String getDeviceId(final TelephonyManager telephonyManager, final int i) {
                                            Log.d(TAG, "call getDeviceId slotIndex.");
                                            return "";
                                        }

                                        /**

```

```
38                                      * 媒体接管 WifiInfo.getMacAddress()接
   口调用，如果媒体实现了这个接口，SDK不会再主动调用系统接口
39                                      * @param wifiInfo
40                                      * @return
41                                      */
42                                     @Override
43                                     public String getMacAddress(final Wif
   iInfo wifiInfo) {
44                                         Log.d(TAG, "call getMacAddress.")
   ;
45                                         return "";
46                                     }
47
48                                     /**
49                                      * 媒体接管 NetworkInterface.getHardwa
   reAddress()接口调用，如果媒体实现了这个接口，SDK不会再主动调用系统接口
50                                      * @param networkInterface
51                                      * @return
52                                      */
53                                     @Override
54                                     public byte[] getHardwareAddress(fina
   l NetworkInterface networkInterface) {
55                                         Log.d(TAG, "call getHardwareAddre
   ss.");
56                                         return null;
57                                     }
58
59                                     /**
60                                      *  媒体接管 TelephonyManager.getImei
   (int slotIndex) 接口调用，如果媒体实现了这个接口，SDK不会再主动调用系统接口
61                                      * @param telephonyManager
62                                      * @param i
63                                      * @return
64                                      */
65                                     @Override
66                                     public String getImei(final Telephony
   Manager telephonyManager, final int i) {
67                                         Log.d(TAG, "call getImei slotInde
   x.");
68                                         return "";
69                                     }
70
71                                     /**
72                                      * 媒体接管 TelephonyManager.getImei
   () 接口调用，如果媒体实现了这个接口，SDK不会再主动调用系统接口
73                                      * @param telephonyManager
74                                      * @return
75                                      */
```

```
                                    @Override
                                    public String getImei(final TelephonyManager telephonyManager) {
                                        Log.d(TAG, "call getImei.");
                                        return "";
                                    }
                                })
                                .build();

        GlobalConfig globalConfig = GlobalConfig.newBuilder()
                                                .setDebug(true) //需要看SDK日志可以打开这个开关，正式包务必设置为false
                                                .build();
        NoahSdk.init(this.getApplication(), config, globalConfig);
```

广告测试

正式广告位不保证百分百填充，测试阶段如果要稳定返回广告，可以在超级汇川聚合平台对应的应用下创建测试广告位，使用测试广告位进行测试。注意：正式上线的广告位务必使用正式广告位。

常见错误码

错误码	说明
-1	SDK未初始化，请调用初始化接口
1001	没有广告填充，服务器未找到合适广告填充，可以多试几次或者等待一会再请求
1002	广告位配置错误，请检查申请的广告位ID和广告类型是否匹配或者APPID和广告位ID是否对应，如果检查无问题，可以联系我们
1011	同一个广告位重复请求
1017	SDK内部广告缓存已经达到上限，不再接受新的广告请求，开发者可以直接使用getAd接口获取广告
1019	广告审核失败，一般认为广告创意有违法违规内容，不予填充

其他接入问题，欢迎随时联系我们!

IOS

接入准备

运行环境

- 支持 iOS 9 及以上
- SDK 编译环境 Xcode 14.2 及以上
- 支持架构：x86-64，arm64

ATS 配置

在 Info.plist 添加 **App Transport Security Settings**，首先点击左侧展开箭头，然后再点右侧 + 号，

添加 **Allow Arbitrary Loads** 索引，并将值设置为 YES

ATT 配置

1. 从 iOS 14 开始，需要开发者设置 App Tracking Transparency 来向用户申请跟踪授权。如果用户未授权或者拒绝此授权，获取到的 IDFA 值为 0000-0000-0000-0000，这可能会导致广告收入降低
2. 获取 App Tracking Transparency 权限，需要在 **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 iOS 14.5 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：

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


#import <AppTrackingTransparency/AppTrackingTransparency.h>

ATTrackingManagerAuthorizationStatus status = ATTrackingManager.trackingAuthorizationStatus;
if(status == ATTrackingManagerAuthorizationStatusNotDetermined) {
    [ATTrackingManager requestTrackingAuthorizationWithCompletionHandler:^(ATTrackingManagerAuthorizationStatus status) { }];
}
```

SDK 接入

推荐使用 CocoaPods 方式接入，CocoaPods 会自动处理编译问题

> **注意**：无论使用哪种方式接入 SDK，必须先将聚合 NoahAdSdks 产物添加至对应工程目录下

Cocoapods

使用 local pod 方式接入，代码示例：

```Objective-C
// 聚合SDK所在目录
pod 'NoahAdSdks' , :path=>'./xx'
```

手动引入

1. 将 NoahAdSdks 加入工程 (手动拖动时，选择 Copy items if needed)
2. 添加系统和三方依赖库

frameworks	"CoreTelephony", "SystemConfiguration", "WebKit", "ImageIO", "Accelerate", "Photos", "AssetsLibrary", "CoreServices", "AddressBook", "AVKit", "CoreData", "Security", "CoreGraphics", "MobileCoreServices", "MessageUI", "SafariServices", "StoreKit", "AVFoundation", "MediaPlayer", "JavaScriptCore", "QuickLook", "CoreMotion", "CoreMedia", "CoreLocation", "MapKit", "AdSupport", "AppTrackingTransparency"
weak frameworks	DeviceCheck
libraries	"c++abi", "sqlite3", "c++", "xml2", "resolv", "bz2.1.0", "z", "resolv.9"
vendored	SDWebImage, YYModel, AFNetworking

SDK 初始化

Objective-C

```objc
#import <NoahSDK/NoahSDK.h>

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    [self initNoahSdk];
    return YES;
}

- (void)initNoahSdk {
    // 平台申请的 App Key
    NSString *appKey = @"10029";
    NoahSdkConfig *sdkConfig = [NoahSdkConfig new];
    [sdkConfig setAppKeyValue:appKey];
    [NoahSdk initWithConfig:sdkConfig globalConfig:nil];
}
```

开屏广告

广告请求

Objective-C

```objective-c
#import <NoahSDK/NoahSDK.h>

@interface AppDelegate () <NoahSplashAdListener>

@property (nonatomic, strong, nullable) SplashAd *splashAd;

@end

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    [self loadSplashAd];
    return YES;
}

- (void)loadSplashAd {
    RequestInfo *req = [RequestInfo new];
    // 开屏底部 logo，可选
    req.splashBottomView = nil;
    // 落地页控制器
    UIViewController *vc = self.window.rootViewController;
    // 开屏超时，单位毫秒
    long timeout = 5000;
    [SplashAd getAdWithSlot:@"10000068"
                requestInfo:req
                   listener:self
                 controller:vc
                       stop:nil
                    timeout:timeout
              useCustomRender:NO];
}
```

广告回调

返回的 SplashAd 必须由外部持有，并且在使用完之后调用 destroy 和释放该对象，否则会内存泄漏

Objective-C

```
// 广告加载成功
- (void)onSplashAdLoaded:(SplashAd *)ad {
    // 内存释放，开屏广告结束后必须调用 destroy，否则会内存泄漏
    if (self.splashAd) {
        [self.splashAd destroy];
    }
    // 必须持有，不然会影响之后的曝光点击回调
    self.splashAd = ad;

    // 开屏广告手动调用展示
    [ad showSplashAdView];
}

// 广告关闭
- (void)onSplashAdClosed:(SplashAd *)ad {
    // 内存释放
    if (self.splashAd) {
        [self.splashAd destroy];
        self.splashAd = nil;
    }
}

// 广告加载失败
- (void)onSplashAdError:(AdError *)error {

}

// 广告曝光
- (void)onSplashAdShown:(SplashAd *)ad {

}

// 广告点击
- (void)onSplashAdClicked:(SplashAd *)ad {

}
```

原生广告

广告加载

为了更好的用户体验，建议在广告请求的时候设置广告 MediaView 的图片占位图：

mediaViewPlaceholderImage 或者 mediaViewPlaceholderImageName

Objective-C

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

- (void)loadNativeAd {
    RequestInfo *req = [RequestInfo new];

    // MediaView 占位图，优先级高于 mediaViewPlaceholderImageName
    req.mediaViewPlaceholderImage = [UIImage imageNamed:@"ad_placeholder"
];

    // 广告落地页控制器，必传
    UIViewController *vc = [UIApplication sharedApplication].delegate.wind
ow.rootViewController;
    [NativeAd getAdWithSlot:@"10000069"
                requestInfo:req
                   listener:self
                 controller:vc];
}
```

广告回调

```objective-c
// 广告加载成功
- (void)onNativeAdLoaded:(NativeAd *)ad {
    // 展示广告
    [self showNativeAd:ad];
}

// 广告点击
- (void)onNativeAdClick:(NativeAd *)ad {

}

// 广告展示
- (void)onNativeAdShown:(NativeAd *)ad {

}

// 广告加载失败
- (void)onNativeAdError:(AdError *)error {

}

// 原生广告事件回调
-(void)onNativeAdEvent:(NativeAd *)ad eventId:(int)eventId ext:(id)extInfo {

}
```

广告展示

返回的 NativeAd 必须由外部持有，并且在使用完之后调用 destroy 和释放该对象，否则会内存泄漏

```objective-c
- (void)dealloc {
    // 内存释放
    if (_nativeAd) {
        [_nativeAd destroy];
    }
}

- (void)showNativeAd:(NativeAd *)ad {
    // 必须持有，不然会影响之后的曝光点击回调
    self.nativeAd = ad;

    // 获取广告容器，有的三方广告一定使用其内部类作为自渲染的基础容器，如果 nil 则表示可以使用任意 UIView
    // 如果不为 nil，则必须要用返回的视图作为自渲染广告的基础容器！！！！
    UIView *adView = [ad nativeAdContainerView];
    if (adView == nil) {
        adView = [UIView new];
    }

    // 广告封面（视频 & 图片）
    UIView *mediaView = [ad getMediaView];
    [adView addSubview:mediaView];
    // 原生广告物料
    SdkAssets *assets = [ad getAdAssets];
    // 标题
    [assets getTitle];
    // 描述
    [assets getDescription];
    // 广告来源
    [assets getAdnName];

    // 展示广告
    [self.view addSubview:adView];
    // 注册点击事件，必须在 addSubview 之后
    [ad registerContainer:adView clickableViews:@[adView]];
}
```

激励视频

广告加载

```objective-c
- (void)loadRewardAd {
    RequestInfo *requestInfo = [RequestInfo new];
    [RewardedVideoAd getAdWithSlot:@"10000070"
                       requestInfo:requestInfo
                          listener:self
                        controller:self];
}
```

广告回调

Objective-C

```
- (void)onReVidoAdLoaded:(RewardedVideoAd *)ad {
    // 内存释放
    if (self.rewardedVideoAd) {
        [self.rewardedVideoAd destroy];
    }

    // 必须持有，不然会影响之后的曝光点击回调
    self.rewardedVideoAd = ad;

    // 广告展示
    [ad show];
}

// 广告关闭
- (void)onReVidoAdClosed:(RewardedVideoAd *)ad {
    if (self.rewardedVideoAd) {
        [self.rewardedVideoAd destroy];
        self.rewardedVideoAd = nil;
    }
}

// 广告点击
- (void)onReVidoAdClicked:(RewardedVideoAd *)ad {

}

// 获得奖励
- (void)onReVidoRewarded:(RewardedVideoAd *)ad {

}

// 加载失败
- (void)onReVidoAdError:(AdError *)error {

}

// 广告曝光
- (void)onReVidoAdShown:(RewardedVideoAd *)ad {

}

/**
 *  激励视频广告唤端 异步查奖回调
 *  code 对应 ERROR_CODE_REWARD_XXX
 *  reward 奖励信息 可为空
```

```
 */
-(void)onReVideoQueryReward:(int)code reward:(nullable NSDictionary *)reward;
```

异步查奖 （版本880新增）

淘宝唤端激励浏览类的广告，不能确保同步发奖，需要结合场景在合适的时机进行历史奖励的查询

Objective-C

```
/*
 * RewardedVideoAd.h
 */
/**
 *  异步查奖
 *   resultInfo 奖励信息，可为空
 *   errCode 查看  NAQueryRewardCodeDefine.h
 */
+ (void)queryAdRewardResultWithSlot:(NSString *)slotKey completion:(void(^)(NSDictionary * _Nullable resultInfo, NARewardQueryCode errCode))completion;
```

查奖回调参数说明

参数	说明	是否必填

NARewardQueryCode errCode	回调返回状态码
	typedefNS_ENUM(int, NARewardQueryCode) {
	NARewardQueryTimeout = -2,
	NARewardQueryInternalError = -1,
	NARewardQueryHasReward = 0,
	NARewardQueryNoAdn = 3,
	};
		是

NSDictionary * _Nullable resultInfo	"pid" : "mm_1_2_4", "task_type" : 3,// 发奖类型 2 浏览 3 下单 "session_id" : "mock_session_id2",// 奖励id，唯一，投放时和广告绑定 "complete_time" : "20231129 12:30:03", // 任务完成时间 "reward_name" : "mock_reward_name2",// 奖励名称 "reward_count" : 1, // 奖励数量	否

与外部广告源竞价

如果媒体有其他广告源与智能营销广告SDK竞价，竞价结束需要回传竞价胜出或失败事件，不回传竞价结果事件会影响计费：

```
typedef NS_ENUM(NSUInteger, NAAdBidLossReason)
{
    NAAdBidLossLowPrice          = 1, //价格竞败
    NAAdBidLossFloorPrice        = 2, //底价过滤
    NAAdBidLossTimeout           = 3, //广告超时返回
    NAAdBidLossFrequencyControl  = 4, //广告频控
    NAAdBidLossOther             = 5, //其他原因
};

/**
 * 竞价胜出调用
 * @param price 如果是二价结算，传入媒体竞价的二价，如果是一价结算传入参与竞价的智能营销SDK广告的价格，单位：分/ecpm。
                媒体竞价二价的定义为：当智能营销SDK广告竞价胜出时，竞价队列中次高价ADN的出价 + 1分，例如次高ADN出价100分，二价就是101分。
                媒体是采用一价结算还是二价结算由商务合同决定。
 */
- (void)sendWinNotification:(int)price;

/**
 * 竞价失败调用
 * @param price 媒体竞价的最高价，单位：分/ecpm
                建议如实传入媒体竞胜的最高价，有利于智能营销SDK广告的优化。如果不能传最高价，请传入参与竞价的智能营销SDK广告本身的价格
 * @param reason 竞价失败的原因 详见 NAAdBidLossReason
 */
- (void)sendLossNotification:(int)price reason:(NAAdBidLossReason)reason;

```

常见错误码

错误码	描述
-1	SDK 未初始化
1001	没有广告填充，服务器未找到合适广告填充，可以多试几次或者等待一会再请求

1002	广告位配置错误，请检查申请的广告位 id 和广告类型是否匹配或者 App id 和广告位 id 是否对应，如果检查无问题，可以联系我们

其他接入问题，欢迎随时联系我们!