**1.SDK接入**

### **1.1SDK包的导入**

#### **导入SDK依赖的包**

**在项目级的build.gradle中添加maven的url**

allprojects {  
 repositories {  
 maven {  
 url "http://test.vlion.cn:8081/nexus/content/repositories/overseas

/"  
 }  
 }  
}

在应用级(app)的build.gradle中添加依赖包：

dependencies {  
 implementation ('cn.vlion.overseas:overseas:3.0')  
}

**1.2 AndroidManifest配置**

**1.2.1 添加权限**

在manifest配置权限：

<uses-permission android:name="android.permission.INTERNET"/>  
<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"/>  
<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE"/>

**注意：**

<!--如若申请下载类广告，6.0以上系统需要开发者动态申请写入权限-->

<!--<uses-permission android:name="android.permission.READ\_PHONE\_STATE"/>-->

**1.2.2 适配Anroid7.0**

在AndroidManifest.xml的application标签中配置网络安全权限：

<application  
 android:networkSecurityConfig="@xml/network\_security\_config">

</application>

其中，network\_security\_config.xml文件，是res目录下新建一个xml的文件，将文件复制xml下面.把AndroidManifest.xml文件复制到res目录下的xml文件下面

<network-security-config>

...  
 <domain-config cleartextTrafficPermitted="true">  
 <domain includeSubdomains="true">127.0.0.1</domain>  
 </domain-config>

...  
</network-security-config>

**注意：**

<!--如若申请下载类广告，7.0以上系统需要开发者在res文件下创建xml文件：res\xml\vlion\_file\_provider.xml,并在AndroidManifest

.xml文件中进行配置，如下：-->

<provider  
 android:name="android.support.v4.content.FileProvider"  
 android:authorities="${applicationId}.fileprovider"  
 android:exported="false"  
 android:grantUriPermissions="true">  
 <meta-data  
 android:name="android.support.FILE\_PROVIDER\_PATHS"  
 android:resource="@xml/vlion\_file\_provider"/>  
</provider>

其中，Vlion\_file\_privider.xml文件内容如下：

<?xml version="1.0" encoding="utf-8"?>  
<paths xmlns:android="http://schemas.android.com/apk/res/android">  
 <!--  
 ## 说明  
 这个文件为开发者适配Android 7.0系统以上才需要配置的相关内容，如果没有兼容到的话，可以不管  
 -->  
  
 <cache-path  
 name="apk1"  
 path="."/>  
 <external-path  
 name="apk2"  
 path="."/>  
   
</paths>

**1.2.3 配置资源**

在资源文件res下的values--->strings中需要配置string资源，该字符串请向运营相关人员寻要，其中，字符串命名需固定为：

<string name="G\_id">该值需要有瑞狮相关人员提供</string>

**1.3 初始化SDK**

请向瑞狮联盟相关人员申请测试的 瑞狮appid和瑞狮tid以及广告位adId。

开发者需要在Application#onCreate()方法中调用以下代码来初始化，

ADManager.*getInstance*().init(上下文, "瑞狮后台提供id", getResources().getString(R.string.*G\_id*));//其中，G\_id必须和string资源中保持一致

**2.加载广告**

**2.1 构建BANNER广告对象 :**

##### ****横幅广告主要API****

|  |  |
| --- | --- |
| **方法名** | **方法说明** |
| New BannerManager | BannerManager初始化 |
| getBannerAd(Context context, String adId, ViewGroup rootGroup, VlionAdListener vlionAdListener ) | 获取横幅广告视图。参数说明：context(上下文)，rootGroup(banner广告的容器) ,adId(广告位id)，VlionAdListener (广告回调) |
| onDestory() | 释放BannerManager所占资源 |

##### ****BannerView回调监听****

|  |  |
| --- | --- |
| **方法名** | **方法说明** |
| onRequestSuccess(String adId); | 广告请求成功。参数说明：adId(广告位id) |
| onRequestFailed(String adId, int code, String errorMsg) | 广告请求失败。参数说明：adId(广告位id)，code(错误码)，errorMsg(错误信息) |
| onShowSuccess(String adId) | 广告展示成功 |
| onClicked(String adId) | 广告点击事件 |
| OnClosed(String adId) | 广告关闭 |
| onImpression(String adId); | 横幅广告曝光回调 |

构建banner广告实例：

mBannerManager = new BannerManager();//初始化  
mBannerManager.getBannerAd(BannerActivity.this,adId,ll\_show, new VlionAdListener() {  
 //广告请求成功

@Override  
 public void onRequestSucceed(String adId) {  
 Log.*e*(TAG,"onRequestSucceed adId="+adId);  
 }

//广告展示成功  
 @Override  
 public void onShowSucceed(String adId) {  
 Log.*e*(TAG,"onShowSucceed adId="+adId);  
 }  
 //广告请求失败  
 @Override  
 public void onRequestFailed(String adId, int code, String errorMsg) {  
 Log.*e*(TAG,"onRequestFailed adId="+adId);  
 }  
 //广告点回调  
 @Override  
 public void onClicked(String adId) {  
 Log.*e*(TAG,"onRequestFailed adId="+adId);  
 }  
  
 //广告关闭

@Override  
 public void onClosed(String adId) {  
 Log.*e*(TAG,"onClosed adId="+adId);  
 }

//广告曝光回调  
 @Override  
 public void onImpression(String adId) {  
 Log.*e*(TAG,"onImpression adId="+adId);  
 }  
});

在activity的onDestroy()调用销毁

@Override  
protected void onDestroy() {  
 super.onDestroy();  
 if (null != mBannerManager ) {  
 mBannerManager .onDestroy();  
}}

详细使用请参考demo中BannerActivity

**2.2 构建插页式广告对象 :**

|  |  |
| --- | --- |
| **方法名** | **方法说明** |
| New InterstitialManager() | 插页式广告初始化 |
| getInterstitialAd(Context context, String adId  , VlionAdListener vlionAdListener) | 展示插页式广告，参数说明：context(上下文)，adId（广告位id）,vlionAdListener（广告回调） |
| onDestroy() | 不需要展示的时候，释放interstitialManager所占资源 |

##### ****插屏广告主要API****

##### ****回调监听****

|  |  |
| --- | --- |
| **方法名** | **方法说明** |
| onRequestSuccess(String adId) | 广告请求成功 |
| onRequestFailed(String adId, int code, String errorMsg) | 请求失败。参数说明：adId(广告位id)，code(错误码)，errorMsg(错误信息) |
| onShowSucceed(String adId) | 插页式广告展示成功 |
| onImpression(String adId) | 插页式广告曝光回调 |
| onClicked(String adId) | 插页式广告点击，此时SDK内部关闭了广告 |
| onClosed(String adId) | 插页式广告关闭 |

请求插页式广告实例：

InterstitialManager interstitialManager = new InterstitialManager();  
interstitialManager.getInterstitialAd(InterstitialActivity.this, adId, new VlionAdListener() {  
 @Override  
 public void onRequestSucceed(String adId) {  
 Log.*e*(TAG, "onRequestSucceed adId=" + adId);  
 }  
  
 @Override  
 public void onShowSucceed(String adId) {  
 Log.*e*(TAG, "onShowSucceed adId=" + adId);  
 }  
  
 @Override  
 public void onRequestFailed(String adId, int code, String errorMsg) {  
 Log.*e*(TAG, "onRequestFailed adId = " + adId + "----code = " + "---errorMsg = " + errorMsg);  
 }  
  
 @Override  
 public void onClicked(String adId) {  
 Log.*e*(TAG, "onClicked adId = " + adId);  
 }  
  
 @Override  
 public void onClosed(String adId) {  
 Log.*e*(TAG, "onClosed adId = " + adId);  
 }  
  
 @Override  
 public void onImpression(String adId) {  
 Log.*e*(TAG, "onImpression = " + adId);  
 }  
});

在activity的ondestroy调用销毁

@Override  
protected void onDestroy() {  
 super.onDestroy();  
 if (null != mInterstitialManager) {  
 mInterstitialManager.onDestory();  
 }  
}

详细使用请参考demo中InterstitialActivity

**2.3 构建原生广告对象 :**

|  |  |
| --- | --- |
| **方法名** | **方法说明** |
| New NativeManager() | 构建NativeManager对象 |
| getNativeAd(Activity context, String adId, NativteVlionAdListener nativeListener) | 请求原生广告。参数说明：context(上下文)，adId(广告位id)，nativeListener(回调) |
| onRequestSuccess(NativeVlionAd data) | 广告数据获取成功。其中，NativeVlionAd 为广告内容实体类 |
| onRequestFailed(String adId, int code, String message) | 广告数据获取失败。参数说明：adId(广告位id)，code(错误码)，message(错误信息) |
| onClicked(String adId) | 广告点击 |
| onClosed(String adId) | 广告关闭 |
| onImpression(String adId) | 广告曝光 |
| onDestroy() | 释放NativeManager所占资源 |

##### ****原生广告****NativeVlionAd 对象的

|  |  |
| --- | --- |
| **属性** | **介绍** |
| String title | 原生广告的标题 |
| String lable | 原生广告的标签 |
| String adBody | 原生广告的内容描述 |
| String icon | 原生广告的图标 |
| List<String>images | 广告的图片 |
| String adSocialContext | 广告 |
| String callToAction | 广告呼叫操作 |
| Boolean callToAction |  |

****原生广告****NativeFeedsData进行曝光

|  |  |
| --- | --- |
| onExposured（ViewGroup containerView, View clickView  ） | viewGroup(广告容器)，点击的广告View(广告点击)可选参数 |

请求原生广告实例：

nativeManager.getNativeAd(NativeActivity.this, adId, new NativeVlionAdListener() {  
 @Override  
 public void onRequestSucceed(String adId, NativeVlionAd nativeVlionAd) {  
 initNativeAd(nativeVlionAd);  
 @Override  
 public void onRequestFailed(String adId, int code, String errorMsg) {  
 }

@Override  
 public void onClicked(String adId) {  
 }

@Override  
 public void onClosed(String adId) {  
 }

@Override  
 public void onImpression(String adId) {  
 }  
});

其中，initNativeAd()方法是自定义NativeAd布局

private void initNativeAd(NativeVlionAd nativeVlionAd) {  
 NativeActivity.this.nativeVlionAd=nativeVlionAd;  
 TextView nativeAdTitle = findViewById(R.id.*native\_ad\_title*);  
 TextView nativeAdSocialContext = findViewById(R.id.*native\_ad\_social\_context*);  
 TextView nativeAdBody = findViewById(R.id.*native\_ad\_body*);  
 TextView sponsoredLabel = findViewById(R.id.*native\_ad\_sponsored\_label*);  
 Button nativeAdCallToAction = findViewById(R.id.*native\_ad\_call\_to\_action*);  
  
  
 nativeAdTitle.setText(nativeVlionAd.getTitle());  
 nativeAdBody.setText(nativeVlionAd.getAdBody());  
 nativeAdSocialContext.setText(nativeVlionAd.getAdSocialContext());  
 nativeAdCallToAction.setVisibility(nativeVlionAd.hasCallToAction() ? View.*VISIBLE* : View.*INVISIBLE*);  
 nativeAdCallToAction.setText(nativeVlionAd.getAdCallToAction());  
 sponsoredLabel.setText(nativeVlionAd.getLable());  
  
 if (!TextUtils.*isEmpty*(nativeVlionAd.getIcon())) {  
 ImageView imageView = new ImageView(NativeActivity.this);  
 imageView.setLayoutParams(new ViewGroup.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*, ViewGroup.LayoutParams.*MATCH\_PARENT*));  
 iconView.addView(imageView);  
 Glide.*with*(NativeActivity.this).load(nativeVlionAd.getIcon()).into(imageView);  
 }  
  
 // Create a list of clickable views  
 List<View> clickableViews = new ArrayList<>();  
 clickableViews.add(nativeAdTitle);  
 clickableViews.add(nativeAdCallToAction);  
  
 nativeVlionAd.onRegisterView(NativeActivity.this, rootView, iconView, native\_ad\_title, mediaView, ad\_choices\_container, clickableViews);  
}

原生广告的销毁

if (null != nativeManager)  
 nativeManager.onDestroy();

参考NativeActivity里面

**2.4 构建激励视频广告对象 :**

##### ****激励视频广告主要API****

|  |  |
| --- | --- |
| **方法名** | **方法介绍** |
| New VideoManager() | 初始化VideoManager |
| getVideoAd(Context context, String adId  , RewardVlionAdListener rewardVlionAdListener) | 获取激励视频广告，参数说明：context(上下文)，adId（广告位id）,rewardVlionAdListener（广告回调） |
| onRewardLoad(String adId) | 缓存成功 |
| onClicked(String adId) | 点击广告 |
| onShowSucceed(String adId) | 广告展示成功 |
| onRequestFailed(String adId, int code, String msg) | 广告请求失败 |
| onClosed(String adId) | 视频关闭回调 |
| onRewardStarted(String adId) | 视频开始激励 |
| onRewardCompleted(String adId) | 视频激励完成回调 |
| onReward(String adId) | 激励用户回调 |

请求激励视频广告实例：

videoManager = new VideoManager();  
videoManager.getVideoAd(VideoActivity.this, adId, new RewardVlionAdListener() {  
 @Override  
 public void onRequestFailed(String adId, int code, String errorMsg) {  
 Log.*e*(TAG, "onRequestFailed:adId=" + adId);  
 }  
  
 @Override  
 public void onClicked(String adId) {  
 Log.*e*(TAG, "onClicked:adId=" + adId);  
 }  
  
 @Override  
 public void onClosed(String adId) {  
 Log.*e*(TAG, "onClosed:adId=" + adId);  
 }  
  
 @Override  
 public void onImpression(String adId) {  
 Log.*e*(TAG, "onImpression:adId=" + adId);  
 }  
  
 @Override  
 public void onRewardLoaded(String adId) {  
 Log.*e*(TAG, "onRewardLoaded:adId=" + adId + "+++isReady()=" + videoManager.isReady());  
 if (videoManager.isReady()) {  
 videoManager.show();  
 Log.*e*(TAG, "onRewardLoaded:adId=" + adId + "+++show");  
 }  
 }  
  
 @Override  
 public void onShowSucceed(String adId) {  
 Log.*e*(TAG, "onShowSucceed:adId=" + adId);  
 }  
  
 @Override  
 public void onRewardStarted(String adId) {  
 Log.*e*(TAG, "onRewardStarted:adId=" + adId);  
 }  
  
 @Override  
 public void onRewardCompleted(String adId) {  
 Log.*e*(TAG, "onRewardCompleted:adId=" + adId);  
 }  
  
 @Override  
 public void onRewarded(String adId) {  
 Log.*e*(TAG, "onRewarded:adId=" + adId);  
 }  
 }  
  
);

在activity的onResume()、onPause()、onDestroy()分别调用

@Override  
protected void onPause() {  
 super.onPause();  
 if (null != videoManager)  
 videoManager.onPause();  
}  
@Override  
protected void onResume() {  
 super.onResume();  
 if (null != videoManager)  
 videoManager.onResume();  
}  
@Override  
protected void onDestroy() {  
 super.onDestroy();  
 if (null != videoManager) {  
 videoManager.onDestory();  
 }  
}

详细使用请参考demo中VideoActivity

**3.混淆配置**

如果您需要使用proguard混淆代码，需确保不要混淆SDK的代码。 请在proguard.cfg文件(或其他混淆文件)尾部添加如下配置:

**-dontwarn** com.google.android.gms.\*\*  
**-keep** class com.google.android.gms.\*\*  
**-keepnames** class com.google.android.gms.\*\* { \*; }  
**-keepclassmembers** class com.google.android.gms.\*\* {\*;}  
  
**-keep** class com.google.android.gms.internal.\*\*{  
 public \*;

**-dontwarn** com.facebook.ads.internal.\*\*  
**-keeppackagenames** com.facebook.\*  
**-keep** public class com.facebook.ads.\*\* {  
 public protected \*;  
}  
**-dontwarn** com.facebook.ads.\*\*

**-keep** class cn.vlion.internation.ad.\*\*{\*;}

#忽略警告  
**-ignorewarning**

如果遇到对接出现的其他问题，请联系瑞狮相关人员，非常感谢！