广告SDK接入说明文档

1. SDK简介

1.1 广告类型

提供5种广告：横幅广告、视频广告、开屏广告、插屏广告、原生广告

> 注：具体广告效果参照广告sdk的Demo程序运行效果。

1.2 广告SDK版本升级

针对广告SDK版本升级，参照以下步骤：

1. 从官网下载最新版本(https://www.yousuode.cn/download/sdk) 解压获取aar包（在 ngad-sdk-all-*.zip\04-DependentLibs\aar\ngad-sdk-release-*.aar 里），对比版本号，如果已经是最新版本则无需替换；如果不是最新版本则需要**替换更新aar包**。

2. Android Studio工程，比较简单。**删除本地工程的旧版aar包，替换最新版本**；

3. Eclipse 工程，**解压aar包分别拷贝到旧版内容的目录包括：assets、res、libs、libs/armeabi，覆盖替换相关的文件，**

> 注意：旧版本sdk的多余资源文件需要删除。

1.3. ChangeLog

版本号	接入变更
v2.6.40	1、新增魅族sdk_2.5.0.1 2、更新huawei_13.4.28.305,oppo_3.2.9
v2.6.30	1、支持广点通插屏、banner2.0接口 2、修复了一些已知bug
v2.6.20	1、更新头条sdk_2.7.5.2, 广点通_4.150.1020，mintegral_10.2.1 2、广点通banner，插屏支持2.0接口
v2.5.70	1、更新vivo_3.2.0.1, sigmob_2.13.1，兼容android 10系统 2、新增Xiaomi_2.4.3，Huawei渠道广告支持
v2.5.44	1、AndroidManifest.xml新增FileProvider组件声明变更 2、更新UniplayAdSdk_6.0.2.jar，删除UniplayAdSdk_5.8.2.jar
v2.5.30	1、解决包冲突问题 2、AndroidManifest.xml权限和组件声明变更 3、去除依赖库gamesdk-framework-shellbase-7.1.5.61.aar(jar)、alisdk-utdid-20160516.jar、adp_sdk_shell_base-1.0.6.0.jar、assets/ucgamesdk/lib(如果集成游戏SDK，则不用去除) 4、新增assets/adpsdk/lib、adp-sdk-shell-2.0.0.12.jar
v2.4.10	1、支持Android9运行，Android8适配 2、AndroidManifest.xml权限和组件声明变更 3、去除依赖库lmjoy_video-*.jar 4、依赖依赖库support-v4要求升级到26+版本
v2.3.50	1、变更所有错误码
v2.2.90	1、添加检测工具（05-CheckTool）
v2.2.60	1、新增了原生广告接口
v2.2.50	1、AndroidManifest.xml的权限和组件声明 2、Proguard文件 3、libs目录引用的AAR或JAR文件版本号及数量 4、res目录文件 **5、新增targetVersion>=24时，FileProvider路径配置**
v2.2.30	1、AndroidManifest.xml的权限和组件声明 2、Proguard文件 3、libs目录引用的AAR或JAR文件版本号及数量 4、res目录文件 5、添加广告加载异常处理说明：**单次启动应用，如果连续5次加载广告失败，不再发起请求。**

2. SDK嵌入

2.1 步骤1：添加SDK到工程中

解压开ngad-sdk-all-*.zip文件，在04-DependentLibs目录下可以看到NGASDK提供的两种接入方式的依赖库。

【Android Studio工程引入】（推荐）

1. **把“aar”目录下的libs目录内容拷贝到游戏的Android项目工程对应的libs目录中**

2. **添加Gradle脚本**

```
//指定本地aar文件的存放目录，libs为build.gradle的相对地址
repositories {
    flatDir {
        dirs'libs'
    }
}

dependencies {
    //集成工程中libs目录下的jar文件（alisdk-utdid-20160516.jar）
    compile fileTree(include: ['*.jar'], dir: 'libs')
    compile 'com.android.support:support-v4:23.1.1'

    //集成工程中libs目录下的aar文件（ngad-sdk-release-*.aar）
    fileTree(dir: 'libs', include: ['*.aar']).each { file ->
        compile(name: file.name.lastIndexOf('.').with {
            it != -1 ? file.name[0..<it] : file.name
        }, ext: 'aar')
    }
}
```

4. **废弃资源删除**

- 检查libs目录或jni目录，删除旧版在libs目录中提供的 **libffmpeg.so**、 **libinitHelper.so**、**libng_util.so**、 **librotate.so**、 **libu3player.so**、 **libun7z.so**
- 去除lmjoy_video-*.jar、gamesdk-framework-shellbase-7.1.5.61.aar(jar)、alisdk-utdid-20160516.jar、adp_sdk_shell_base-1.0.6.0.jar、assets/ucgamesdk/lib(如果集成游戏SDK，则不用去除)**

5. **解决冲突**

> 1. **如果已经集成九游的游戏SDK8.x版本，则广告SDK版本需升级至2.5.30以上版本**
> 2. **其他冲突解决方案就联系技术支撑人员**

【Eclipse工程引入】（不推荐）

1. **把“jar接入方式”目录下的libs、assets、res目录内容拷贝到游戏的Android项目工程中对应的libs、assets、res目录中**
2. **引入android-support-v4.jar （可以从网上获取，若已存在则忽略）**
3. **废弃资源删除**
   a) 检查libs目录或jni目录，删除旧版在libs目录中提供的 **libffmpeg.so**、 **libinitHelper.so**、**libng_util.so**、 **librotate.so**、 **libu3player.so**、 **libun7z.so**、**lmjoy_video-*.jar**
4. **解决冲突**

> 1. **如果已经集成九游的游戏SDK8.x版本，则广告SDK版本需升级至2.5.30以上版本**
> 2. **通常通过jar包方式接入的厂商，经常会漏添加或更新res、assets目录下的资源、混淆规则，所以在尽可能保持**
> 3. **其他冲突解决方案就联系技术支撑人员**

2.2 步骤2：修改AndroidManifest.xml文件

添加权限声明：

```
    <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" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"
/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
    <uses-permission
android:name="android.permission.REQUEST_INSTALL_PACKAGES" />
    <uses-permission android:name="android.permission.WRITE_SETTINGS" />

    <!-- 更多的权限配置以输出的AndroidManifest.xml定义为准 -->
```

> 如果您打包App时的targetSdkVersion >= 23：请在先获取到SDK要求的所有权限，然后再调用SDK的广告接口。否则SDK将无法工作，我们建议您在App启动时就去获取SDK需要的权限，Demo工程中也提供了基本的权限处理示例代码供开发者参考。
>
> 如果您打包App时的targetSdkVersion >= 24：除了需要处理好权限申请以外，还需要处理好文件访问的兼容性。

添加SDK组件声明(更多组件配置以输出的AndroidManifest-must.xml定义为准)

```
        <!-- NGASDK START -->

        <!-- targetSDKVersion >= 24时才需要添加这个provider。provider的authorities
属性的值为${applicationId}.fileprovider，请开发者根据自己的${applicationId}来设置这个
值 -->
        <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/adp_file_path" />
        </provider>
```

```
        <activity
            android:name="cn.sirius.nga.activity.ProxyActivity"
            android:configChanges="keyboardHidden|orientation|screenSize"
            android:theme="@android:style/Theme.Translucent.NoTitleBar"
            android:windowSoftInputMode="adjustResize">
        </activity>

        <!-- 更多组件配置以输出的AndroidManifest-must.xml定义为准 -->

<!-- NGASDK END -->
```

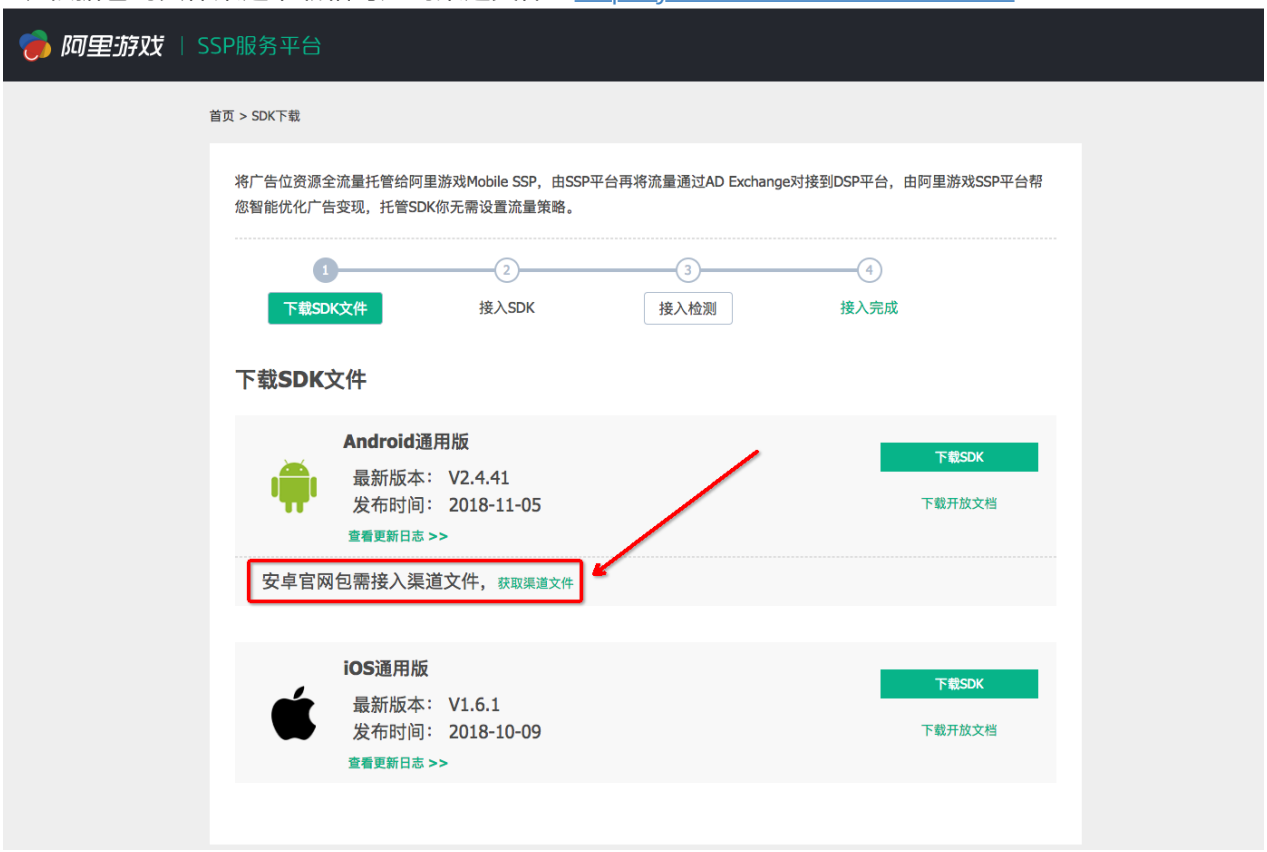
注意：这里的 ${applicationId} 不是广告的appId，而是Android工程（接入游戏包名）applicationId，比如 com.brianbaek.popstar

在项目结构下的`res`目录下添加一个`xml`文件夹，拷贝`adp_file_path.xml`文件到xml文件夹中，文件内容如下：

```
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- 这个下载路径不可以修改，必须是GDTDOWNLOAD -->
    <external-path name="gdt_sdk_download_path" path="GDTDOWNLOAD" />
    <root-path name="download" path="" />
    <external-path name="external_files" path="."/>
    <external-files-path name="external_files_path" path="Download" />
</paths>
```

2.3 步骤3：非九游渠道加入渠道文件（九游渠道包可省略该步骤）

1、根据包的具体渠道下载相对应的渠道文件：http://yousuode.cn/download/sdk



2、解压出文件 UCGameConfig.ini，放入游戏包的assets文件夹下

2.4 步骤4：使用检测工具验证是否存在接入问题**

1. 解压checktool.zip到任意目录下
2. 在该目录下，mac和linux系统运行 start.sh，Windows运行 start.bat
3. 按步骤执行，根据检测结果修正接入问题

> 注：九游渠道包不需要勾选复选框，官网渠道包需要勾选复选框


广告SDK检测工具2.0.1
请选择apk包位置(支持文件拖放) Please select apk(support file drag and drop)
2
选择
select
开始检测
starting test
是否官网渠道包 whether the channel package
1 check the check box
文件: ad-sdk-demo-cp-release-2.5.50.apk
3 select target app or game package
03-DEMO
名称
修改日期
ngad-sdk-demo-cp-release-2.5.50...
2019年7月4日 星期四 下午4:15
ngad-sdk-demo-src.zip
2019年7月4日 星期四 下午4:12
ngad-sdk-demoJar-src.zip
2019年7月4日 星期四 下午4:12
文件格式:
所有文件
4 choose
新建文件夹
取消
选择


广告SDK检测工具2.0.1
/leo/Downloads/checkTool (3)/../ngad-sdk-demo-cp-release-2.5.50 (1).apk
选择
5
检测
是否官网渠道包
checktool-version=2.0.1
6 Review test results and correct integration issues based on test logs
========== 成功：10，警告：1，错误：0 ==========
【检测项】清除缓存
> 检测说明：清除上次检测的缓存数据
> 检测结果：成功
【检测项】检查 广告SDK 版本资源
> 检测说明：游戏包接入SDK版本:2.5.50
> 检测结果：成功
【检测项】检查 游戏SDK 版本资源
> 检测说明：游戏SDK未接入或未正确接入
> 检测结果：成功
【检测项】检查官网渠道号文件
> 检测说明：非九游渠道游戏,需要在assets下放置渠道文件UCGameConfig.ini
> 检测结果：警告
> 修复建议：官网渠道号文件不存在,请检查
【检测项】第三方必需jar检查
> 检测说明：检查必需jar内容：[android.support.v4]

3. 接入代码

3.1 初始化SDK

在LauncherActivity / WelcomeActivity / SplashActivity 中进行SDK的初始化。

NGASDK初始化是使用SDK所提供功能可以执行的前提，否则会导致广告加载失败。游戏在应用启动时LauncherActivity的onCreate方法中插入广告SDK初始化逻辑。

```
public void onCreate() {
    super.onCreate();
    initSdk(this, new NGASDK.InitCallback() {
        @Override
        public void success() {
            showAd(WelcomeActivity.this);
        }

        @Override
        public void fail(Throwable throwable) {
            throwable.printStackTrace();
        }
    });
}

private static void initSdk(Activity activity, final NGASDK.InitCallback
initCallback) {
    // 重新初始化sdk
    Log.d(TAG, AdConfig.toStringFormat());
    NGASDK ngasdk = NGASDKFactory.getNGASDK();
    Map<String, Object> args = new HashMap<>();
    args.put(NGASDK.APP_ID, AdConfig.appId);
    //打Release包的时候，需要把DebugMode设置为false
    args.put(NGASDK.DEBUG_MODE, BuildConfig.DEBUG);
    ngasdk.init(activity, args, initCallback);
}
```

NGASDK主要API

public class NGASDKFactory

限定符和类型	方法和说明
static NGASDK	getNGASDK()

public interface NGASDK

限定符和类型	方法和说明
<T extends NGAProperties>	loadAd() 加载广告请求 注意：为了提高广告的填充率以及曝光量，建议重新加载广告时候重新调用此方法，重新请求新的广告内容。

3.2 横幅广告

- **广告接入代码示例**

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_ad_control);
}

private NGABannerController mController;
private NGABannerProperties mProperties;
private ViewManager mWindowManager;
private RelativeLayout mBannerView;

//注意：请在Activity成员变量保存，使用匿名内部类可能导致回收
NGABannerListener mAdListener = new NGABannerListener() {
    @Override
    public void onRequestAd() {
        ToastUtil.show(TAG, "onRequestAd");
    }

    @Override
    public <T extends NGAdController> void onReadyAd(T controller) {
        mController = (NGABannerController) controller;
        ToastUtil.show(TAG, "onReadyAd");
    }

    @Override
    public void onShowAd() {
        ToastUtil.show(TAG, "onShowAd");
    }

    @Override
    public void onCloseAd() {
        //广告关闭之后mController置null，鼓励加载广告重新调用loadAd，提高广告填充率
        mController = null;
        ToastUtil.show(TAG, "onCloseAd");
        mBannerView.setVisibility(View.GONE);
    }
```

```
    @Override
    public void onErrorAd(int code, String message) {
        ToastUtil.show(TAG, "onErrorAd errorCode:" + code + ", message:" +
message);
    }

    @Override
    public void onClickAd() {
        ToastUtil.show(TAG, "onClickAd");
    }
};

//为了提高广告的填充率以及曝光量，建议重新加载广告时候重新调用此方法，重新请求新的广告内容
private void loadAd(Activity activity) {
    if (mBannerView != null && mBannerView.getParent() != null) {
        mWindowManager.removeView(mBannerView);
    }
    mBannerView = new RelativeLayout(activity);
    WindowManager.LayoutParams params = new WindowManager.LayoutParams();
    params.width = ViewGroup.LayoutParams.WRAP_CONTENT;
    params.height = ViewGroup.LayoutParams.WRAP_CONTENT;
    params.gravity = Gravity.BOTTOM | Gravity.CENTER;
    params.flags = WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE;
    mWindowManager = (WindowManager)
activity.getSystemService(Context.WINDOW_SERVICE);
    mWindowManager.addView(mBannerView, params);

    mProperties = new NGABannerProperties(activity, AdConfig.appId,
AdConfig.bannerPosId, mBannerView);
    mProperties.setListener(mAdListener);
    NGASDK ngasdk = NGASDKFactory.getNGASDK();
    ngasdk.loadAd(mProperties);

    // 若需要默认横幅广告不展示
    mBannerView.setVisibility(View.GONE);
}

private void destroyAd(Activity activity) {
    if (mWindowManager != null) {
        mWindowManager.removeView(mBannerView);
        mWindowManager = null;
    }
    if (mController != null) {
        mController.closeAd();
        mController = null;
    }
}

private void showAd(Activity activity) {
```

```
    if (mController != null) {
        mController.showAd();
        mBannerView.setVisibility(View.VISIBLE);
    }
}

private void closeAd(Activity activity) {
    if (mController != null) {
        mBannerView.setVisibility(View.GONE);
        mController.closeAd();
    }
}
```

> 注：更多详细代码示例，请参考Demo工程代码：**cn.sirius.adsdkdemo.BannerActivity**

> 注意：设置的广告回调对象应该是成员对象，不应该是new 的临时对象。因为广告sdk内部对回调对象做了一次软引用WeakReference包装，临时对象没有被其他逻辑引用可能会回收释放，最终会导致调用方收不到回调事件。

- **Banner广告主要API**

public class NGABannerProperties extends NGAProperties<NGABannerController,NGABannerListener>

限定符和类型	方法和说明
NGABannerListener	getListener() 获取广告的监听（回调）对象
void	setListener() 设置广告的监听（回调）对象，设置时机是在发起广告请求之前。

public abstract class NGAProperties<Controller extends NGAdController,Listener extends NGAdListener>

限定符和类型	方法和说明
Activity	getActivity() 返回广告显示的Activity对象
java.lang.String	getAppId() 返回App ID
ViewGroup	getContainer() 返回广告显示的容器
java.lang.Object	getExtraData() 返回扩展属性（内部属性）
abstract Listener	getListener() 获取广告的监听（回调）对象
java.lang.String	getPositionId() 返回广告位ID
void	setExtraData(java.lang.Object data) 设置扩展属性（内部属性）
abstract void	setListener 设置广告的监听（回调）对象，设置时机是在发起广告请求之前。

public interface NGABannerListener extends NGAdListener

public interface NGAdListener

限定符和类型	方法和说明
void	onClickAd() 点击广告
void	onCloseAd() 关闭广告
void	onErrorAd(int code, java.lang.String message) 广告过程发生错误
<T extends NGAdController>	onReady(T controller) 广告已准备好，通知广告逻辑生成并获取得到控制对象
void	onRequestAd() 通知广告请求
void	onShowAd() 显示广告

public interface NGABannerController extends NGAdController

public interface NGAdController

限定符和类型	方法和说明
void	closeAd() 关闭广告
void	showAd() 显示广告

3.3 插屏广告

- **广告接入代码示例**

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_ad_control);

}

private NGAInsertProperties mProperties;
private NGAInsertController mController;

//注意：请在Activity成员变量保存，使用匿名内部类可能导致回收
NGAInsertListener mAdListener = new NGAInsertListener() {

    @Override
    public void onShowAd() {
        ToastUtil.show(TAG, "onShowAd");
    }


    @Override
    public void onRequestAd() {
        ToastUtil.show(TAG, "onRequestAd");
    }

    @Override
    public <T extends NGAdController> void onReadyAd(T controller) {
        mController = (NGAInsertController) controller;
        ToastUtil.show(TAG, "onReadyAd");
    }

    @Override
    public void onCloseAd() {
        mController = null;
        ToastUtil.show(TAG, "onCloseAd");
    }
```

```
    @Override
    public void onClickAd() {
        ToastUtil.show(TAG, "onClickAd");
    }

    @Override
    public void onErrorAd(int code, String message) {
        ToastUtil.show(TAG, "onErrorAd errorCode:" + code + ", message:" +
message);
    }

};

//为了提高广告的填充率以及曝光量，建议重新加载广告时候重新调用此方法，重新请求新的广告内容
private void loadAd(Activity activity) {
    mProperties = new NGAInsertProperties(activity, AdConfig.appId,
AdConfig.insertPosId, null);
    mProperties.setListener(mAdListener);
    NGASDK ngasdk = NGASDKFactory.getNGASDK();
    ngasdk.loadAd(mProperties);
}

public void destroyAd(Activity activity) {
    if (mController != null) {
        mController.cancelAd();
        mController.closeAd();
        mController = null;
    }
}

private void showAd(Activity activity) {
    if (mController != null) {
        mController.showAd();
    }
}

private void closeAd(Activity activity) {
    if (mController != null) {
        //mController.show(false);
        mController.cancelAd();
        mController.closeAd();
    }
}
```

> 注：更多详细代码示例，请参考Demo工程代码：**cn.sirius.adsdkdemo.InsertActivity**

- **插屏广告主要API**

public class NGAInsertProperties extends NGAProperties>

限定符和类型	方法和说明
NGAInsertListener	getListener() 获取广告的监听（回调）对象
void	setListener (NGAInsertListener listener) 设置广告的监听（回调）对象，设置时机是在发起广告请求之前。

public interface NGAInsertListener extends NGAdListener

public interface NGAInsertController extends NGAdController

限定符和类型	方法和说明
void	cancelAd() 取消广告请求

3.4 视频广告

- **广告接入代码示例**

```
public class VideoActivity extends BaseActivity {
   private static final String TAG = "VideoActivity";
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_video_ad_control);
   }
   private NGAVideoController mController;
   NGAVideoListener mAdListener = new NGAVideoListener() {

        @Override
        public void onShowAd() {
         ToastUtil.show(TAG, "onShowAd");
        }

        @Override
        public void onClickAd() {
         ToastUtil.show(TAG, "onClickAd");
        }

        @Override
      public void onCloseAd() {
         mController = null;
         ToastUtil.show(TAG, "onCloseAd");
      }

      @Override
      public void onErrorAd(final int code, final String message) {
```

```
            ToastUtil.show(TAG, "onErrorAd code=" + code);
        }

        @Override
        public void onRequestAd() {
            ToastUtil.show(TAG, "onRequestAd");
        }

        @Override
        public <T extends NGAdController> void onReadyAd(T controller) {
            mController = (NGAVideoController) controller;
            ToastUtil.show(TAG, "onReadyAd");
        }

        @Override
        public void onCompletedAd() {
            ToastUtil.show(TAG, "onCompletedAd");
        }
    };

    //为了提高广告的填充率以及曝光量，建议重新加载广告时候重新调用此方法，重新请求新的广告内容
    public void loadAd(Activity activity) {
        final NGAVideoProperties properties = new NGAVideoProperties(activity,
AdConfig.appId, AdConfig.videoPosId);
        properties.setListener(mAdListener);
        NGASDK ngasdk = NGASDKFactory.getNGASDK();
        ngasdk.loadAd(properties);
    }

    public void onClick(View view) {
        switch (view.getId()) {
            case R.id.btn_video_ad_init:
                loadAd(this);
                break;
            case R.id.btn_video_ad_uninit:
                if (mController != null) {
                    mController.destroyAd();
                }
                break;
            case R.id.btn_video_ad_has_cache:
                if (mController != null) {
                    boolean hasCacheAd = mController.hasCacheAd();
                    ToastUtil.show(TAG, "hasCacheAd=" + hasCacheAd);
                }
                break;
            case R.id.btn_video_ad_show:
                if (mController != null) {
                    mController.showAd();
                }
```

```
            break;
        }
    }
}
```

> 注：更多详细代码示例，请参考Demo工程代码：**cn.sirius.adsdkdemo.VideoActivity**

- **视频广告主要API**

  public class NGAVideoProperties extends NGAProperties>

限定符和类型	方法和说明
NGAVideoListener	getListener() 获取广告的监听（回调）对象
void	setListener (NGAVideoListener listener) 设置广告的监听（回调）对象，设置时机是在发起广告请求之前。

public interface NGAVideoListener extends NGAdListener

限定符和类型	方法和说明
void	onCompletedAd() 视频广告播放完成

public interface NGAVideoController extends NGAdController

限定符和类型	方法和说明
void	destroyAd() 销毁广告
boolean	hasCacheAd() 查询是否有缓存广告

> 注意： loadAd加载视频广告，如果有　onReady 回调，表明有新广告返回，调用showAd播放新的视频广告，而不是缓存的广告，
> 如果没有 onReady 回调，接入逻辑使用旧的mController 对象调用showAd 就会播放缓存的广告。

3.5 开屏广告

通常开屏广告会前嵌入在LauncherActivity / WelcomeActivity / SplashActivity里面，们在“NGASDK初始化”的success回调中插入请求加载开屏逻辑

- **广告接入代码示例**

```
public class WelcomeActivity extends Activity {
```

```
private static final String TAG = "WelcomeActivity";

private ViewGroup container;
// [非必须]根据场景需求决定是否需要自定义的跳过按钮
//private TextView skipView;
private ImageView splashHolder;
private static final String SKIP_TEXT = "点击跳过 %d";

@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_splash);
   container = (ViewGroup) this.findViewById(R.id.splash_container);
   //skipView = (TextView) findViewById(R.id.skip_view);
   //skipView.setVisibility(View.VISIBLE);

   splashHolder = (ImageView) findViewById(R.id.splash_holder);

   initSdk(this, new NGASDK.InitCallback() {
     @Override
     public void success() {
         //NGASDK初始化成功后，开始加载开屏广告
         showAd(WelcomeActivity.this);
     }

     @Override
     public void fail(Throwable throwable) {
         throwable.printStackTrace();
     }
 });

   splashHolder.postDelayed(new Runnable() {
      @Override
      public void run() {
         if (splashHolder.getVisibility() == View.VISIBLE) {
            // 超时3s开屏广告还没加载出来则关闭广告
            closeAd();
         }
      }
   }, 3000);
}

private NGAWelcomeProperties properties;
private boolean canCloseAd = false;
//注意：请在Activity成员变量保存，使用匿名内部类可能导致回收
NGAWelcomeListener mAdListener = new NGAWelcomeListener() {

   @Override
   public void onClickAd() {
```

```
            ToastUtil.show(TAG, "onClickAd");
        }

        @Override
        public void onErrorAd(int code, String message) {
            ToastUtil.show(TAG, "onErrorAd errorCode:" + code + ", message:" +
message);
            closeAd();
        }

        @Override
        public void onShowAd() {
            splashHolder.setVisibility(View.INVISIBLE); // 广告展示后一定要把预设的开
屏图片隐藏起来
            ToastUtil.show(TAG, "onShowAd");
        }

        @Override
        public void onCloseAd() {
            //无论成功展示成功或失败都回调用该接口，所以开屏结束后的操作在该回调中实现
            ToastUtil.show(TAG, "onCloseAd");
            canCloseAd = true;
        }

        @Override
        public <T extends NGAdController> void onReadyAd(T controller) {
            // 开屏广告是闪屏过程自动显示不需要NGAdController对象，所以返回controller为
null;
            ToastUtil.show(TAG, "onReadyAd");
        }

        @Override
        public void onRequestAd() {
            ToastUtil.show(TAG, "onRequestAd");
        }


        /**
         * 倒计时回调，返回广告还将被展示的剩余时间。
         * 通过这个接口，开发者可以自行决定是否显示倒计时提示，或者还剩几秒的时候显示倒计时
         *
         * @param millisUntilFinished 剩余毫秒数
         */
        @Override
        public void onTimeTickAd(long millisUntilFinished) {
            //skipView.setText(String.format(SKIP_TEXT,
Math.round(millisUntilFinished / 1000f)));
        }
    };
```

```
   /**
    * 开始广告，建议：闪屏Activity显示后延迟再加载广告
    * @param activity
     */
   public void showAd(Activity activity) {
      properties = new NGAWelcomeProperties(activity, AdConfig.appId,
AdConfig.welcomeId, container);
      // 支持开发者自定义的跳过按钮。SDK要求skipContainer一定在传入后要处于VISIBLE状态，
且其宽高都不得小于3x3dp。
      // 如果需要使用SDK默认的跳过按钮，可以选择上面两个构造方法。
      //properties.setSkipView(skipView);

      properties.setListener(mAdListener);
      NGASDK ngasdk = NGASDKFactory.getNGASDK();
      ngasdk.loadAd(properties);
   }

   /**
    * 关闭广告，当通知广告失败{@link NGAWelcomeListener#onErrorAd} 或关闭事件时候调用
{@link NGAWelcomeListener#onCloseAd}
    */
   private void closeAd() {
      // 如果是因为点击广告而关闭广告，则不能finish Activity
      if (canCloseAd) {
         this.startActivity(new Intent(this, MainActivity.class));
         this.finish();
      } else {
         canCloseAd = true;
      }
   }

   @Override
   protected void onPause() {
      super.onPause();
      canCloseAd = false;
   }

   @Override
   protected void onResume() {
      super.onResume();
      // 如果是因为点击广告而返回，则finish Activity
      if (canCloseAd) {
         this.startActivity(new Intent(this, MainActivity.class));
         this.finish();
      }
      canCloseAd = true;
   }
```

```
   /** 开屏页一定要禁止用户对返回按钮的控制，否则将可能导致用户手动退出了App而广告无法正常曝光和计费 */
   @Override
   public boolean onKeyDown(int keyCode, KeyEvent event) {
      if (keyCode == KeyEvent.KEYCODE_BACK || keyCode == KeyEvent.KEYCODE_HOME) {
         return true;
      }
      return super.onKeyDown(keyCode, event);
   }
}
```

> 注：更多详细代码示例，请参考Demo工程代码： **cn.sirius.adsdkdemo.WelcomeActivity**

- **开屏广告主要API**

  public class NGAWelcomeProperties extends NGAProperties>

限定符和类型	方法和说明
NGAWelcomeListener	getListener() 获取广告的监听（回调）对象
View	getSkipView() 获取自定义的跳过按钮
void	setListener (NGAWelcomeListener listener) 设置广告的监听（回调）对象，设置时机是在发起广告请求之前。
void	setSkipView(View skipView) 可以通过传入传入skipContainer参数，支持开发者自定义的跳过按钮。

public interface NGAWelcomeListener extends NGAdListener

限定符和类型	方法和说明
void	onTimeTickAd(long millisUntilFinished) 倒计时回调，返回广告还将被展示的剩余时间。

public interface NGAWelcomeController extends NGAdController

3.6 模板插屏广告

模板插屏广告与插屏广告有展示方式类似，再普通插屏广告的基础上，提供了UI的模板，具有一定的自定义属性，使得模板插屏有更好的融入性，同时也能提升广告收益

- **广告接入代码示例：**

```
public class TemplateActivity extends BaseActivity {

    private static final String TAG = "TemplateActivity";
    Map<String, String> mShowParam = new HashMap<>();
    RadioGroup mTemplateSelector;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_template);
        mTemplateSelector = (RadioGroup)findViewById(R.id.template_choice);
        //UI参数准备，具体参数名称请查阅文档
        mShowParam = new HashMap<>();
        //bgRes：弹窗背景框图片资源（模板1、模板2支持）
        mShowParam.put("bgRes", "drawable/bg");
        //dialogTextColor：弹窗文案文字颜色（仅模板2支持）
        mShowParam.put("dialogTextColor", "#edeef2");
        //front：弹窗文字字体资源（仅模板2支持，仅支持assets资源）
        mShowParam.put("front", "assets/heiti.ttf");
        //btnText：弹窗操作按钮文案（仅模板2支持）
        mShowParam.put("btnText", "点击领取");
        //btnTextColor：弹窗操作按钮文案文字颜色（仅模板2支持）
        mShowParam.put("btnTextColor", "#36561f");
        //btnRes：弹窗操作按钮背景图片（仅模板2支持）
        mShowParam.put("btnRes", "drawable/button_bg");
    }

    private NGAGeneralProperties mProperties;
    private NGAGeneralController mController;

    //注意：请在Activity成员变量保存，使用匿名内部类可能导致回收
    NGAGeneralListener mAdListener = new NGAGeneralListener() {

        @Override
        public void onEvent(NGAdEvent event) {
            //具体eventId代表含义请查阅文档
            if (event.eventId == 1) {
                ToastUtil.show(TAG, "onGameBtnClick");
                //再次调用showAd(Map<String,String> param)，可以动态更换素材参数
                mShowParam.put("dialogText", "点击领取");
                mController.showAd(mShowParam);
            }
        }

        @Override
        public void onShowAd() {
            ToastUtil.show(TAG, "onShowAd");
        }
```

```
        @Override
        public void onRequestAd() {
            ToastUtil.show(TAG, "onRequestAd");
        }

        @Override
        public <T extends NGAdController> void onReadyAd(T controller) {
            mController = (NGAGeneralController) controller;
            ToastUtil.show(TAG, "onReadyAd");
        }

        @Override
        public void onCloseAd() {
            ToastUtil.show(TAG, "onCloseAd");
            mController = null;
        }

        @Override
        public void onClickAd() {
            ToastUtil.show(TAG, "onClickAd");
        }

        @Override
        public void onErrorAd(int code, String message) {
            ToastUtil.show(TAG, "onErrorAd errorCode:" + code + ", message:" +
message);
        }

    };

    //为了提高广告的填充率以及曝光量，建议重新加载广告时候重新调用此方法，重新请求新的广告内
容
    private void loadAd(Activity activity) {
        Map<String, Object> param = new HashMap<>();
        param.put(NGAGeneralProperties.APP_ID, AdConfig.appId);
        //不同广告位可以配置不同的模板效果，可以与我们联系进行选择
        param.put(NGAGeneralProperties.POSITION_ID, getSelectedPosId());
        param.put(NGAGeneralProperties.AD_TYPE, 20);
        mProperties = new NGAGeneralProperties(activity, null, param);
        mProperties.setListener(mAdListener);

        NGASDK ngasdk = NGASDKFactory.getNGASDK();
        ngasdk.loadAd(mProperties);
    }

    public void destroyAd(Activity activity) {
        if (mController != null) {
            mController.closeAd();
```

```
            mController = null;
        }
    }

    private void showAd(Activity activity) {
        if (mController != null) {
            mShowParam.put("dialogText", "小小礼物请领取");
            mController.showAd(mShowParam);
        }
    }

    private void closeAd(Activity activity) {
        if (mController != null) {
            //mController.show(false);
            mController.closeAd();
        }
    }

    public void onClick(View view) {
        if (view.getId() == R.id.btn_banner_create) {
            loadAd(this);
        } else if (view.getId() == R.id.btn_banner_destroy) {
            destroyAd(this);
        } else if (view.getId() == R.id.btn_banner_show) {
            showAd(this);
        } else if (view.getId() == R.id.btn_banner_close) {
            closeAd(this);
        }
    }

    private String getSelectedPosId() {
        switch (mTemplateSelector.getCheckedRadioButtonId()) {
            case R.id.rb_temp_1st:
                return AdConfig.templateId;
            case R.id.rb_temp_2nd:
                return AdConfig.templateId_2;
            default:
                return "";
        }
    }
}
```

> 注：更多详细代码示例，请参考Demo工程代码： **cn.sirius.adsdkdemo.TemplateActivity**

- **模板插屏广告主要API**

public class NGAGeneralProperties extends NGAProperties<NGAGeneralController, NGAGeneralListener>

限定符和类型	方法和说明
NGAGeneralListener	getListener() 获取广告的监听（回调）对象
void	setListener(NGAGeneralListener listener) 设置广告的监听（回调）对象，设置时机是在发起广告请求之前。

public interface NGAGeneralListener extends NGAdListener

public interface NGAWelcomeController extends NGAdController

模板传入参数详细

在请求到广告，获得controller后，可以调用showAd(Map<String,String> param)方法来传入自定义参数，来调整模板的样式，具体参数描述如下：

参数key	参数意义	参数格式	使用示例	备注	使用范围
bgRes	弹窗背景图资源	String	mShowParam.put("bgRes", "drawable/bg");	获取drawable文件夹下的资源，支持drawable、mipmap	1号模板、2号模板
dialogText	弹窗游戏文案	String	mShowParam.put("dialogText", "点击领取");	直接传入文字内容传参	2号模板
dialogTextColor	弹窗游戏文案颜色	String	mShowParam.put("dialogTextColor", "#edeef2");	使用色码来表示颜色，需要“#”开头	2号模板
btnText	操作按钮文案	String	mShowParam.put("btnText", "点击领取");	直接传入文字内容传参	2号模板
btnTextColor	操作按钮文案颜色	String	mShowParam.put("btnTextColor", "#36561f");	使用色码来表示颜色，需要“#”开头	2号模板
btnRes	操作按钮图资源	String	mShowParam.put("btnRes", "drawable/button_bg");	获取drawable文件夹下的资源，支持drawable、mipmap，支持png、xml格式	2号模板
front	字体文件	String	mShowParam.put("btnRes", "assets/heiti.ttf");	获取assets文件夹下的资源，获得字体资源文件	2号模板

showAd(Map<String,String> param)可以重复调用，动态改变弹窗样式，但曝光仅计算一次

如果上述参数不填写，则由sdk补充默认参数

模板事件回调id

当模板上触发特殊事件时，sdk会回调onEvent(NGAdEvent event)，接入方可根据event.eventId判断回调事件类型，具体参数如下：

eventId	参数意义
1	操作按钮被点击（首次）
2	操作按钮被点击（第二次或者以上）

3.7 原生广告

原生广告提供了高度自定义的广告类型，使得广告能与app高度融合，给开发者提供了高自由度的开发接口

- **广告接入代码示例：**

```
public class NativeAdActivity extends BaseActivity {
    private static final String TAG = "NativeAdActivity";

    private NGANativeProperties mProperties;
    private NGANativeController mController;

    private NGANativeAdData mAdDataItem;

    private LinearLayout mMainContainer;
    private RelativeLayout mAdContainer;
    private ImageView mIvAppIcon;
    private TextView mTvAppTitle;
    private TextView mTvAppDesc;
    private TextView mTvAppScore;
    private ImageView mIvAppImg;
    private TextView mBtnClick;
    private ImageView mIvAdLogo;

    //注意：请在Activity成员变量保存，使用匿名内部类可能导致回收
    NGANativeListener mAdListener = new NGANativeListener() {

        @Override
        public void onAdStatusChanged(NGANativeAdData ngaNativeAd, int i,
Map<String, String> map) {
            Log.i(TAG, "onAdStatusChanged " + i);
        }

        @Override
        public void onShowAd() {
            ToastUtil.show(TAG, "onShowAd");
        }


        @Override
        public void onRequestAd() {
            ToastUtil.show(TAG, "onRequestAd");
        }

        @Override
        public <T extends NGAdController> void onReadyAd(T controller) {
            mController = (NGANativeController) controller;
            ToastUtil.show(TAG, "onReadyAd");
        }
```

```
        @Override
        public void onCloseAd() {
            ToastUtil.show(TAG, "onCloseAd");
            mController = null;
        }

        @Override
        public void onClickAd() {
            ToastUtil.show(TAG, "onClickAd");
        }

        @Override
        public void onErrorAd(int code, String message) {
            ToastUtil.show(TAG, "onErrorAd errorCode:" + code + ", message:" +
message);
        }

    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ad_native);
        mMainContainer = (LinearLayout)
findViewById(R.id.rl_control_container);
    }

    private void refreshAdContainer() {
        if (mAdContainer != null) {
            mMainContainer.removeView(mAdContainer);
        }
        mAdContainer = (RelativeLayout)
LayoutInflater.from(this).inflate(R.layout.native_ad_contaner, null, false);
        mIvAppIcon = (ImageView) mAdContainer.findViewById(R.id.iv_app_icon);
        mTvAppTitle = (TextView) mAdContainer.findViewById(R.id.tv_app_title);
        mTvAppDesc = (TextView) mAdContainer.findViewById(R.id.tv_app_desc);
        mTvAppScore = (TextView) mAdContainer.findViewById(R.id.tv_app_score);
        mIvAppImg = (ImageView) mAdContainer.findViewById(R.id.iv_app_img);
        mBtnClick = (TextView) mAdContainer.findViewById(R.id.btn_app_click);
        mIvAdLogo = (ImageView) mAdContainer.findViewById(R.id.iv_ad_logo);
        mMainContainer.addView(mAdContainer);
    }

    private void loadAd(Activity activity) {
        //如果使用原有的容器再次请求广告，必须关闭原有的广告
        closeAd(activity);
        //每次加载广告必须使用新的广告容器view
        refreshAdContainer();
        Map<String, Object> param = new HashMap<>();
```

```
        //一次加载所提供的广告数量。不一定能够给到传入的数量，请以最终返回的广告数量为准
        param.put(NGANativeProperties.KEY_AD_COUNT, 1);
        param.put(NGANativeProperties.APP_ID, AdConfig.appId);
        param.put(NGANativeProperties.POSITION_ID, AdConfig.nativeId);
        mProperties = new NGANativeProperties(activity, param);
        mProperties.setListener(mAdListener);
        NGASDK ngasdk = NGASDKFactory.getNGASDK();
        ngasdk.loadAd(mProperties);
    }

    public void destroyAd(Activity activity) {
        if (mController != null) {
            mController.closeAd();
            mController = null;
        }
    }

    private void showAd(Activity activity) {
        if (mController != null) {
            mAdDataItem = mController.getAdList().get(0);
            new LoadImageUtil(mAdDataItem.getIconUrl()){
                @Override
                public void onReceived(Drawable result) {
                    mIvAppIcon.setImageDrawable(result);
                    //广告曝光后，一定一定要调用该方法，否则无法计算曝光数量
                    mAdDataItem.exposure(mAdContainer);
                }
            }.execute();
            new LoadImageUtil(mAdDataItem.getImgList().get(0)){
                @Override
                public void onReceived(Drawable result) {
                    mIvAppImg.setImageDrawable(result);
                }
            }.execute();
            //根据相关规定，广告必须要有广告标识，请将该广告logo放置在广告右下角
            new LoadImageUtil(mAdDataItem.getAdLogo()) {
                @Override
                public void onReceived(Drawable result) {
                    mIvAdLogo.setImageDrawable(result);
                }
            }.execute();
            mTvAppTitle.setText(mAdDataItem.getTitle());
            mTvAppDesc.setText(mAdDataItem.getDesc());
            if (mAdDataItem.getRating() > 0) {
                mTvAppScore.setText("评分：" + mAdDataItem.getRating());
            } else {
                mTvAppScore.setText("暂无评分");
            }
            mBtnClick.setText(mAdDataItem.getButtonText());
```

```
            mBtnClick.setBackgroundColor(Color.GRAY);
        }
    }

    private void closeAd(Activity activity) {
        if (mController != null) {
            mController.closeAd();
            mController = null;
        }
    }

    @Override
    protected void onDestroy() {
        //原生广告必须调用关闭，否则影响广告计费
        closeAd(this);
        super.onDestroy();
    }

    public void onClick(View view) {
        if (view.getId() == R.id.btn_native_create) {
            loadAd(this);
        } else if (view.getId() == R.id.btn_native_destroy) {
            destroyAd(this);
        } else if (view.getId() == R.id.btn_native_show) {
            showAd(this);
        } else if (view.getId() == R.id.btn_native_close) {
            closeAd(this);
        }
    }
}
```

> 注：更多详细代码示例，请参考Demo工程代码： **cn.sirius.adsdkdemo.NativeAdActivity**

- **原生广告主要API**

public class NGANativeProperties extends NGAProperties<NGANativeController,NGANativeListener>

限定符和类型	方法和说明
NGANativeListener	getListener() 获取广告的监听（回调）对象
void	setListener(NGANativeListener listener) 设置广告的监听（回调）对象，设置时机是在发起广告请求之前。

public interface NGANativeListener extends NGAdListener

public interface NGANativeController extends NGAdController

限定符和类型	方法和说明
`List<NGANativeAdData>`	getAdList() 获取广告对象列表
void	closeAd()标记广告关闭，务必调用，否则影响计量

public interface NGANativeAdData

限定符和类型	方法和说明
String	getTitle(); 获得广告标题文字
String	getDesc(); 获得广告详细描述文字
String	getIconUrl(); 获得广告图标url
List	getImgList(); 获得广告图片url集合
double	getRating(); 获得广告评分
boolean	isApp(); 广告是否app类型
String	getButtonText(); 广告按钮文案
String	getAdLogo(); 获得广告logo图标url，根据相关规定，请将该logo放在广告右下角
void	exposure(View container); 标记广告曝光，并启动监听点击功能，曝光后务必调用，否则影响计量

4. Android 6.0以上版本添加运行时权限

4.1 接入说明

google对于Android 6.0以上的版本，要求添加运行时权限，否则SDK可能由于无权限而导致播放视频失败。

建议在第三方apk初始化时主动请求获取这些权限，权限请求越早越好。

4.2 接入示例

如第三方接入已经有运行时权限检查的功能，请将1.2步骤AndroidManifest.xml中SDK运行需要的权限自行添加。

如第三方接入未添加运行时权限检查功能，根据实际业务情况，参考如下步骤添加，兼容Android 6.0以上的版本。

4.2.1 Acitivity中加入代码

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  onBeforeRequestPermission(savedInstanceState);
  // 如果targetSDKVersion >= 23，就要申请好权限。如果您的App没有适配到
Android6.0（targetSDKVersion < 23），那么只需要在这里直接调用fetchSplashAD接口。
  if (Build.VERSION.SDK_INT >= 23) {
    checkAndRequestPermission();
  } else {
    onRequestPermissionSuccess();
  }
}

protected void onBeforeRequestPermission(Bundle savedInstanceState) {

}

/**
 * 权限请求成功的回调函数
 */
protected void onRequestPermissionSuccess() {

}

protected void setTitle(String title){
  ((TextView)findViewById(R.id.title)).setText(title);
}

public void backPressed(View v){
  onBackPressed();
}

public void switchOrientation(View v) {
  if (getRequestedOrientation() == ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE)
{
    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
  } else {
    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
  }
}
```

```
protected List<String> getNecessaryPermissions() {
    // 根据实际需要，申请必需权限
  return
Arrays.asList(Manifest.permission.READ_PHONE_STATE,Manifest.permission.WRITE_E
XTERNAL_STORAGE,Manifest.permission.ACCESS_FINE_LOCATION);
}
/**
 *
 * ----------非常重要----------
 *
 * Android6.0以上的权限适配简单示例：
 *
 * 如果targetSDKVersion >= 23，那么必须要申请到所需要的权限，再调用SDK，否则SDK不会工
作。
 *
 * Demo代码里是一个基本的权限申请示例，请开发者根据自己的场景合理地编写这部分代码来实现权限申
请。
 * 注意：下面的`checkSelfPermission`和`requestPermissions`方法都是在Android6.0的SDK
中增加的API，如果您的App还没有适配到Android6.0以上，则不需要调用这些方法，直接调用SDK即
可。
 */
@TargetApi(23)
private void checkAndRequestPermission() {
  List<String> lackedPermission = new ArrayList<String>();
  List<String> necessaryPermissions = getNecessaryPermissions();
  for (String necessaryPermission : necessaryPermissions) {
    if (!(checkSelfPermission(necessaryPermission)
==PackageManager.PERMISSION_GRANTED)) {
      lackedPermission.add(necessaryPermission);
    }
  }

  // 权限都已经有了，那么直接调用SDK
  if (lackedPermission.size() == 0) {
    onRequestPermissionSuccess();
  } else {
    // 请求所缺少的权限，在onRequestPermissionsResult中再看是否获得权限，如果获得权就可
以调用SDK，否则不要调用SDK。
    String[] requestPermissions = new String[lackedPermission.size()];
    lackedPermission.toArray(requestPermissions);
    requestPermissions(requestPermissions, 1024);
  }
}

private boolean hasAllPermissionsGranted(int[] grantResults) {
  for (int grantResult : grantResults) {
    if (grantResult == PackageManager.PERMISSION_DENIED) {
      return false;
    }
```

```
  }
  return true;
}

@Override
public void onRequestPermissionsResult(int requestCode, String[] permissions,
in[] grantResults) {
  super.onRequestPermissionsResult(requestCode, permissions, grantResults);
  if (requestCode == 1024 && hasAllPermissionsGranted(grantResults)) {
    onRequestPermissionSuccess();
  } else {
    // 如果用户没有授权，那么应该说明意图，引导用户去设置里面授权。
    Toast.makeText(this, "应用缺少必要的权限！请点击\"权限\"，打开所需要的权
限。",Toast.LENGTH_LONG).show();
    Intent intent = new Intent(Settings.ACTION_APPLICATION_DETAILS_SETTINGS);
    intent.setData(Uri.parse("package:" + getPackageName()));
    startActivity(intent);
    finish();
  }
}
```

4.2.2 在MainActivity（或应用的主Activity类型） #onCreate 函数增加权限判断，如下：

```
@Override
protected void onBeforeRequestPermission(Bundle savedInstanceState) {
    super.onBeforeRequestPermission(savedInstanceState);
    // ... 你的初始化代码 ...
}
```

5. 兼容性

- **请保持你接入的support-v4包保持与targetSdkVersion一致，v4建议26+版本**
- 广告SDK支持的安卓平台版本为14~24，也就是4.0以上到7.0的都支持，包括视频内核。
- **targetSDKVersion >= 24**时的文件访问兼容处理 如果您打包时的`targetSDKVersion >= 24`，为了让SDK能够正常下载、安装App类广告，必须按照下面的三个步骤做兼容性处理。注意：如果您的`targetSDKVersion < 24`，不需要做这个兼容处理。
  （1）在`AndroidManifest.xml`中的`Application`标签中添加`provider`标签，接入代码如下所示：

```
<application
    ... ...

    <!-- targetSDKVersion >= 24时才需要添加这个provider。provider的authorities属性
的值为${applicationId}.fileprovider，请开发者根据自己的${applicationId}来设置这个值 -
->
```

```
    <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/adp_file_path" />
    </provider>
    ... ...
</application>
```

需要注意的是provider的authorities值为${applicationId}.fileprovider，对于每一个开发者而言，这个值都是不同的，${applicationId}在代码中和`Context#getPackageName()`值相等，是应用的唯一id。例如demo示例工程中的applicationId为"com.qq.e.union.demo"。

（2）在项目结构下的`res`目录下添加一个`xml`文件夹，再新建一个`adp_file_path.xml`的文件，文件内容如下：

```
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <external-path name="gdt_sdk_download_path" path="GDTDOWNLOAD" />
    <root-path name="download" path="" />
    <external-path name="external_files" path="."/>
</paths>
```

- **如果您打包 App 时的 targetSdkVersion >= 26：需要在 AndroidManifest.xml 增加权限声明 android.permission.REQUEST_INSTALL_PACKAGES**，详情见前面添加权限声明部分。

6. 注意事项

开发套件

确保所使用的 android-support-v4.jar 包中的 android.support.v4.app.NotificationCompat.Builder 类包含 setProgress 方法，如果不包含此方法请升级 android 开发套件

代码混淆

如果您的发布包（release包）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：

```
-keepattributes SourceFile,LineNumberTable

-keep class com.qq.e.** {*; }
-dontwarn com.qq.e.**
-keep class com.mobvista.** {*; }
-keep interface com.mobvista.** {*; }
-keep class **.R$* { public static final int mobvista*; }
-keep class com.alphab.** {*; }
```

```
-keep interface com.alphab.** {*; }
-dontwarn com.mobvista.**
-dontwarn com.alphab.**
-dontwarn com.uniplay.**
-keep class com.uniplay.** { *; }
-keep class com.lm.** { *; }
-dontwarn com.lm.**
-keep class com.inmobi.** { *; }
-dontwarn com.inmobi.**
-keep public class com.google.android.gms.**
-dontwarn com.google.android.gms.**
-dontwarn com.squareup.picasso.**
-keep class com.google.android.gms.ads.identifier.AdvertisingIdClient{
     public *;
}
-keep class com.google.android.gms.ads.identifier.AdvertisingIdClient$Info{
     public *;
}
# skip the Picasso library classes
-keep class com.squareup.picasso.** {*;}
-dontwarn com.squareup.picasso.**
-dontwarn com.squareup.okhttp.**
# skip Moat classes
-keep class com.moat.** {*;}
-dontwarn com.moat.**
# skip AVID classes
-keep class com.integralads.avid.library.** {*;}

### utdid
-keep class com.ta.utdid2.**{*;}
-keep class com.ut.device.**{*;}
-dontwarn com.ta.utdid2.**
-dontwarn com.ut.device.**

# Keep ngad-sdk classes
-keep interface cn.sirius.nga.** {*; }
-keep class cn.sirius.nga.** {*; }
-dontwarn cn.sirius.nga.**
-keep class cn.uc.** {*;}
-dontwarn cn.uc.**

-keep class cn.ninegame.library.** {*; }
-dontwarn cn.ninegame.library.**

#头条
-dontwarn com.bytedance.sdk.**
-dontwarn com.androidquery.**
-dontwarn com.ss.**
-dontwarn com.pgl.**
```

```
-keep class com.bytedance.** { *; }
-keep class com.androidquery.** {*;}
-keep interface com.pgl.sys.ces.out.** { *; }
-keep class com.ss.** {*;}
-keep public interface com.bytedance.sdk.openadsdk.downloadnew.** {*;}
-keep class com.pgl.** { *; }
-keep class com.androidquery.**{*;}
-keep class com.ss.android.socialbase.**{*;}


#汇量
-keepattributes Signature
-keepattributes *Annotation*
-keepattributes *JavascriptInterface*
-keep class com.mintegral.** {*; }
-keep interface com.mintegral.** {*; }
-keep class android.support.v4.** { *; }
-dontwarn com.mintegral.**
-keep class **.R$* { public static final int mintegral*; }
-keep class com.alphab.** {*; }
-keep class com.iab.omid.** {*; }
-keep interface com.alphab.** {*; }


#oppo
#oppo 广告 proguard 配置开始
-keep class com.opos.** { *;}
-keep class android.support.v4.**{public *;}
-keep class android.support.v7.**{public *; }
-keep class com.heytap.** {*;}
-dontwarn com.heytap.**
-keep class com.cdo.oaps.** { *; }
-dontwarn com.cdo.oaps.**
-keep class com.squareup.wire.** { *; }
-dontwarn com.squareup.wire.**
-keep class com.nearme.instant.** { *; }
-dontwarn com.nearme.instant.**
#oppo广告 proguard 配置结束

#baidu
-keep class com.baidu.mobads.** { *; }
-keep class com.baidu.mobad.** { *; }
#vivoSDK
-keep class com.vivo.*.** { *; }
-keep class vivo.util.** { *; }

#sigmobSDK
-keep class sun.misc.Unsafe { *; }
-dontwarn com.sigmob.**
```

```
-keep class com.sigmob.**.**{*;}
-keep class com.sigmob.**{*;}

#huaweiSDK
-keepclassmembers class * {
    @com.huawei.openalliance.ad.annotations.OuterVisible *;
}

-keep @com.huawei.openalliance.ad.annotations.OuterVisible enum * {
    <fields>;
    public static **[] values();
    public static ** valueOf(java.lang.String);
}

-keepclassmembers enum * {
    @com.huawei.openalliance.ad.annotations.OuterVisible *;
}

-keep,includedescriptorclasses
@com.huawei.openalliance.ad.annotations.OuterVisible interface * { *; }

-keep class com.huawei.openalliance.ad.constant.** { *; }
-keep interface com.huawei.openalliance.ad.constant.** { *; }
-keep class com.uodis.opendevice.aidl.**

-dontwarn okhttp3.**
-dontwarn okio.**

-keepnames class * implements android.os.Parcelable
-keepclassmembers class * implements android.os.Parcelable {
    public final static *** CREATOR;
}

-keep class com.huawei.openalliance.ad.** { *; }
-dontwarn com.huawei.openalliance.**

-keep class com.huawei.hms.** { *; }
-dontwarn com.huawei.hms.**

-keep class com.huawei.appmarket.** { *; }
-dontwarn com.huawei.appmarket



#xiaomi SDK
-keepattributes SourceFile,LineNumberTable

-keep class com.xiaomi.** {
    *;
```

```
}

-keep class com.miui.** {
    *;
}

-keep class oauth.signpost.** {
    *;
}

-keep class com.google.gdata.** {
    *;
}

-keep class miui.os.zeus.** {
    *;
}

############meizu
-keep class pl.droidsonroids.** { *; }
-dontwarn pl.droidsonroids.**
#baidu
-keep class com.baidu.** { *; }
-dontwarn com.baidu.**

#gdt
-keep class com.qq.e.** { *; }
-dontwarn com.qq.e.**
#汇量
-keep class com.mintegral.** { *; }
-dontwarn com.mintegral.**

-keep class com.iab.** { *; }
-dontwarn com.iab.**

#oneWay
-keep class com.liulishuo.** { *; }
-dontwarn com.liulishuo.**

-keep class mobi.oneway.** { *; }
-dontwarn mobi.oneway.**

-keep class com.didi.** { *; }
-dontwarn com.didi.**

#open ad sdk
-keep class com.bytedance.** { *; }
-dontwarn com.bytedance.**
```

```
-keep class com.ss.** { *; }
-dontwarn com.ss.**

-keep class com.pgl.** { *; }
-dontwarn com.pgl.**


#uniplay
-keep class com.uniplay.** { *; }
-dontwarn com.uniplay.**

-keep class com.joomob.** { *; }
-dontwarn com.joomob.**

-keep class com.ss.** { *; }
-dontwarn com.ss.**


#sigmob
-keep class com.sigmob.** { *; }
-dontwarn com.sigmob.**

-keep class com.meizu.** { *; }
-keep class okhttp3.** { *; }
-keep class okio.** { *; }
-dontwarn com.meizu.**
-dontwarn okhttp3.**
-dontwarn okio.**

-keep class android.databinding.** { *; }
-dontwarn android.databinding.**

-keep class android.content.** { *; }
-dontwarn android.content.**

############# meizu done
```

错误码

错误码	错误描述
9001	接入不完整，请检查遗漏jar包
9002	appId缺失，请检查初始化传入参数正确性
9003	无法获取application对象，请检查传入参数
9004	正在初始化，请稍候再试
8101	初始化sdk失败，传入参数错误，请检查appId、posId和activity
8102	该广告位无法请求广告，请检查包名是否正确
8103	短时间内请求广告过于频繁，请稍候重试或使用其他广告位
8104	activity为空,请检查
8105	网络请求错误，请检查网络状态
8120	传入UI容器为空，请检查container参数
8122	开屏广告页面处于不可见状态，请检查您的代码逻辑，保证容器可见
8123	开屏广告的高度不得小于400dp
8124	当前设备的网络类型不符合加载广告的条件，请尝试WIFI环境
8125	加载广告超时，请检查网络情况或稍候重试
8126	加载广告超时，请检查网络情况或稍候重试
8201	暂时未有合适的广告
8302	暂时未有合适的广告
8402	暂时未有合适的广告
8404	暂时未有合适的广告
8502	暂时未有合适的广告
8xxx	内部错误，请与客服联系并描述错误回调值

常见问题FAQ

问题：广告无法加载出来的场景问题

原因分析，可能有以下情况导致：

1. 没有调用sdk初始化：init ，建议在Application OnCreate做广告sdk初始化；
2. Eclipse工程引入sdk情况下，缺失jar、so文件；
3. Eclipse工程引入sdk情况下，没有引入support包；

4. 网络不通（有WiFi但无法联网），导致广告请求连接失败；
   或者：
   - App安装后首次启动初始化逻辑比较多网络请求比较多，导致广告请求连接超时失败；
   - 这种情况下，建议延迟广告请求（如延迟2s后），这样不影响你们其他正常业务的网络请求，同时也保证尽量让每次广告请求都成功。
5. release包混淆配置没有添加广告sdk过滤配置，参考接入说明文档。

问题：设置了广告回调 （NGAdListener子类对象），但是没有收到回调通知。

原因分析：设置的广告回调对象是new 的临时对象，不是成员对象，没有被其他逻辑引用可能会回收释放，

因为广告sdk内部对回调对象做了一次软引用WeakReference包装。

问题：获取到广告控制器（NGAdController对象**）之后，能不能重复调用展示/关闭广告操作（showAd/closeAd）？**

答：横幅、普通插屏、开屏、视频广告不可以， 调用closeAd关闭广告之后，再调用showAd，很可能无法正常显示广告。只有模板广告允许多次调用，以供微调自定义界面。
重新加载广告需要重新调用NGASDK#loadAd方法，等待onReadyAd事件回调获取广告控制器重新加载新的广告（showAd），此操作同时提高广告的填充率以及曝光量。

问题：如何处理广告加载失败情况?

答：根据具体错误码处理。特别地，单次启动应用，如果加载广告失败，切莫即刻再发起请求（可做定时重试），除非是用户行为。