百度百青藤

移动应用推广 SDK

用户手册（Android 版)

百度在线网络技术（北京）有限公司

概述

说明

百度移动应用推广 SDK(Android 版)是百度推出的移动应用推广 SDK 在 Android 平台上的版本（以下简称 SDK）。该文档提供了对如何使用 SDK 的一个详细说明，建议阅读时下载我们的用例工程，代码是最好的使用说明。

如有其他问题可以参考网站的 FAQ，或者及时与我们联系。

背景

开发环境

- **操作系统**：支持 Linux/Mac/Windows 系统，具体依赖开发者选择的 IDE
- **开发工具**：支持 Android studio、Eclipse、Intellij
- **部署目标**：接入百度联盟流量的应用
- **支持设备**：运行了 Android 4.0 以及以上系统的 Android 设备

注册开户

开发者需在百青藤平台（https://mssp.baidu.com/）上进行注册，在平台审核通过后，开发者就成为了百度广告联盟的正式会员。

术语介绍

- **APPSID**：媒体ID

  您在百青藤平台创建媒体时获得的ID，这个ID是我们在广告网络中识别您应用的唯一ID

- **ApID**：广告位ID

您在百青藤平台为您的应用创建的某种类型（Banner、开屏、插屏、信息流）的广告位ID

广告位创建流程

在百青藤平台完成应用审核后，参考如下步骤创建广告位

详细信息参考 *百青藤平台-帮助中心-学习中心* 中的《百青藤产品使用手册》，了解更多信息。

支持的广告类型

百青藤平台支持以下几种广告类型，您可以根据开发需要选择合适的广告：

广告类型	简介	适用场景
横幅（Banner）	常见的移动广告样式，多位于 app 顶部或底部，横跨 app 页面	可用于小说底部，详情页面底部，应用界面顶部等
插屏广告	在应用开启、暂停、退出时以半屏或全屏的形式弹出，展示时机巧妙避开用户对应用的正常体验	视频暂停页，打开新页面可弹出插屏广告
开屏广告	App 启动时，提供广告展示	app 启动时，使用开屏广告

广告类型	简介	适用场景
信息流	开发者可以获取广告素材，自由拼合广告素材，包括广告标题、文字描述、图片和视频（提供视频播放组件，支持全屏小视频播放）	媒体需要主导广告样式时，可以使用信息流广告
信息流智能优选	基于信息流，可定制样式组合，SDK 提供组件，完成相应渲染功能	便于快速集成开发。且应用可直接挂载使用智能优选的渲染组件
贴片广告	可为组件提供贴片，多见于视频组件	可在视频播缓冲，暂停，结束后，将贴片广告附加到组件上
激励视频	用户通过观看短视频广告，在 app 场景内完成“任务”，获取激励奖励	常出现在游戏单局结束，角色复活，登录奖励领取，服务增值等场景
内容联盟 WebView 渲染	提供推文/自媒体文章（内含广告），以 URL 的形式提供给媒体 WebView 渲染	常用于资讯、新闻类场景，方便媒体快速开发
内容联盟 Native 渲染	开发者可以自由渲染资讯内容和广告，包括内容标题、图片、作者等	常用于资讯、新闻类场景，媒体可以自定义 UI 样式
JSSDK	用户 web 页内嵌广告的支持，用法详见 JS 广告文档	

SDK 嵌入

如果您是初次使用百度联盟 SDK，我们建议您利用 Demo 工程来了解百度联盟 SDK 的使用规范。目前支持如下导入方式：

- **AndroidStudio**

  当您使用 **AndroidStudio** 开发环境时（这也是我们推荐的方式，请采用 AndroidStudio 2.3 以上），导入示例工程，即可编译运行 Demo

嵌入广告 SDK

步骤 1：添加 SDK 到工程中

请在工程文件根目录下创建一个名为 `libs` 的子目录，并将百度网盟 SDK 的 aar 包拷贝到 `libs` 目录下

对应的 build.gradle 文件里面添加如下配置

```
repositories{
    flatDir{
        dirs 'libs'
    }
}
dependencies {
    compile(name:'Baidu_MobAds_SDK-release',ext:'aar')
}
```

步骤 2：配置 AndroidManifest 文件

(1) 在 AndroidManifest.xml 的 Application 标签下配置 APPSID

```
<META-DATA android:name="BaiduMobAd_APP_ID" android:value="e866cfb0" />
或者代码 AdView.setAppSid(context, "e866cfb0");
```

将申请到的 APPSID 替换 e866cfb0 位置，两种方式选择其一，配置文件和代码配置等价。
配置文件 META-DATA 标签必须处于 application 标签下才生效

(2) 在 AndroidManifest.xml 添加以下权限声明

必备权限

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

推荐权限

```
<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.permissio.WRITE_EXTERNAL_STORAGE" />
```

如果应用的 targetSdkVersion >= 26：

推荐增加一条权限声明

```
  <uses-permission android:name="android.permission.REQUEST_INSTALL_PACKAGES" />
```

(3) 在 AndroidManifest.xml 声明打开落地页的 Activity

```
<activity
    android:name="com.baidu.mobads.AppActivity"
    android:configChanges="screenSize|keyboard|keyboardHidden|orientation"
    android:theme="@android:style/Theme.Translucent.NoTitleBar"/>
```

(4)声明 BdFileProvider

如果应用的 targetSdkVersion >= 24

为了让 SDK 能够正常下载、安装 App 类广告，必须按照下面的步骤做兼容性处理。

(1) 首先在 AndroidManifest.xml 中的 Application 标签中添加 provider 标签

```
<provider
    android:name="com.baidu.mobads.openad.BdFileProvider"
    android:authorities="${packageName}.bd.provider"
    android:exported="false"
    android:grantUriPermissions="true">
    <meta-data
        android:name="android.support.FILE_PROVIDER_PATHS"
        android:resource="@xml/bd_file_path" />
</provider>
```

需要注意的是 provider 的 authorities 值为 ${packageName}对于每一个开发者而言，这个值都是不同的，${ packageName} 在代码中

和 Context.getPackageName() 值相等（即 build.gradle 文件中的 applicationId 字段），是应用的唯一 id。

例如 Demo 示例工程中的 packageName 为 "com.baidu.mobads.demo.main"。

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

```
<paths xmlns:android="http://schemas.android.com/apk/res/android
">
    <external-files-path name="bdpath" path="bddownload/" />
    <external-path name="bdpathsd " path="bddownload/" />
</paths>
```

步骤 3 代码混淆

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-rules.pro 文件(或其他混淆文件)尾部添加如下配置：

```
-keepclassmembers class * extends android.app.Activity {
   public void *(android.view.View);
}

-keepclassmembers enum * {
```

```
    public static **[] values();
    public static ** valueOf(java.lang.String);
}
-keep class com.baidu.mobads.** { *; }
-keep class com.baidu.mobad.** { *; }
```

敏感权限 API

隐私敏感权限 API

目前 SDK 已支持工信部隐私敏感权限要求。
敏感权限，用户同意则使用，若用户拒绝则不再获取。默认不获取。
申请权限界面由应用依据各自情况制作，并依据用户真实选择，使用如下 api 更新用户权限开关状态。建议授权“红色字体 API”， 可以有效提升广告的 ECPM。

MobadsPermissionSettings

方法名	方法介绍
setPermissionReadDeviceID(boolean permissionGranted)	读取设备 ID 的权限（建议授权）
setPermissionAppList(boolean permissionGranted)	读取已安装应用列表权限（建议授权）
setPermissionLocation(boolean permissionGranted)	读取粗略地理位置权限
setPermissionStorage(boolean permissionGranted)	读写外部存储权限
hasPermissionGranted (String permission)	SDK 授权权限查询

代码如下：

```
// SP的key，读取设备信息权限
private static final String KEY_PHONE_STATE = "key_phone_state";
// SP的key，读取定位权限
private static final String KEY_LOCATION = "key_location";
// SP的key，读写外部存储权限（SD卡）
private static final String KEY_STORAGE = "key_storage";
```

```
// SP的key，读取应用列表权限
private static final String KEY_APP_LIST = "key_app_list";

// 设置SDK可以使用的权限，包含：设备信息、定位、存储、APP LIST
initMobadsPermissions();

public boolean onMenuItemSelected(int featureId, @NonNull
MenuItem item) {
    int id = item.getItemId();
    if (id == R.id.permissions_list) {
        return true;
    }
    // 明文提示用户申请权限
    switch (id) {
        case R.id.phone_state:
            // 申请权限：android.permission.READ_PHONE_STATE
            // 获取IMEI，有助于转化
            showRequestPhoneStateDialog();
            break;
        case R.id.location:
            // 申请权限：android.permission.ACCESS_COARSE_LOCATION
            // 获取定位，有助于精准投放
            showRequestLocationDialog();
            break;
        case R.id.storage:
            // 申请权限：android.permission.WRITE_EXTERNAL_STORAGE
            // 获取外部存储权限，用于广告的下载和缓存
            showRequestExternalStorageDialog();
            break;
        case R.id.app_list:     // 轮播图+文字模版
            showRequestAppListDialog();
            break;
        default:
            // nop
    }
    return super.onOptionsItemSelected(item);
}

public void onRequestPermissionsResult(int requestCode, String[]
permissions, int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
    switch (requestCode) {
        case 1000:
```

```
            updatePermissions(KEY_PHONE_STATE, grantResults[0] ==
PackageManager.PERMISSION_GRANTED);
            break;
        case 1001:
            updatePermissions(KEY_LOCATION, grantResults[0] ==
PackageManager.PERMISSION_GRANTED);
            break;
        case 1002:
            updatePermissions(KEY_STORAGE, grantResults[0] ==
PackageManager.PERMISSION_GRANTED);
            break;
        default:
            // nop
    }
}

/**
 * 初始化设置广告 SDK 的权限，从 SharedPreference 中读取存储的权限状态
 */
private void initMobadsPermissions() {
    MobadsPermissionSettings
            .setPermissionReadDeviceID(DemoSPUtils.getBoolean(thi
s, KEY_PHONE_STATE, false));
    MobadsPermissionSettings
            .setPermissionLocation(DemoSPUtils.getBoolean(this,
KEY_LOCATION, false));
    MobadsPermissionSettings
            .setPermissionStorage(DemoSPUtils.getBoolean(this,
KEY_STORAGE, false));
    MobadsPermissionSettings
            .setPermissionAppList(DemoSPUtils.getBoolean(this,
KEY_APP_LIST, false));
}

private void showRequestPhoneStateDialog() {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setMessage("请求广告需要设备信息权限，是否允许?");
    builder.setPositiveButton("允许", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            if (Build.VERSION.SDK_INT >= 23) {
                if (checkSelfPermission(BaiduSDKDemo.this,
Manifest.permission.READ_PHONE_STATE)) {
```

```
                    updatePermissions(KEY_PHONE_STATE, true);
                } else {
                    requestPermissions(new String[]
{Manifest.permission.READ_PHONE_STATE}, 1000);
                }
            } else {
                updatePermissions(KEY_PHONE_STATE, true);
            }
        }
    });
    builder.setNegativeButton("禁止", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            updatePermissions(KEY_PHONE_STATE, false);
        }
    });
    builder.show();
}

private void showRequestLocationDialog() {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setMessage("请求广告需要定位权限，是否允许?");
    builder.setPositiveButton("允许", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            if (Build.VERSION.SDK_INT >= 23) {
                if (checkSelfPermission(BaiduSDKDemo.this,
Manifest.permission.ACCESS_COARSE_LOCATION)) {
                    updatePermissions(KEY_LOCATION, true);
                } else {
                    requestPermissions(new String[]
{Manifest.permission.ACCESS_COARSE_LOCATION}, 1001);
                }
            } else {
                updatePermissions(KEY_LOCATION, true);
            }
        }
    });
    builder.setNegativeButton("禁止", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
```

```
            updatePermissions(KEY_LOCATION, false);
        }
    });
    builder.show();
}

private void showRequestExternalStorageDialog() {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setMessage("请求广告需要外部存储权限，是否允许?");
    builder.setPositiveButton("允许", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            // Android Q（API 29）支持分区存储，无需单独申请外部存储权限
            if (Build.VERSION.SDK_INT >= 23 &&
Build.VERSION.SDK_INT < 29) {
                if (checkSelfPermission(BaiduSDKDemo.this,
Manifest.permission.WRITE_EXTERNAL_STORAGE)) {
                    updatePermissions(KEY_STORAGE, true);
                } else {
                    requestPermissions(new String[]
{Manifest.permission.WRITE_EXTERNAL_STORAGE}, 1002);
                }
            } else {
                updatePermissions(KEY_STORAGE, true);
            }
        }
    });
    builder.setNegativeButton("禁止", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            updatePermissions(KEY_STORAGE, false);
        }
    });
    builder.show();
}

private void showRequestAppListDialog() {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setMessage("请求广告需要读取应用列表，是否允许?");
    builder.setPositiveButton("允许", new
DialogInterface.OnClickListener() {
        @Override
```

```
        public void onClick(DialogInterface dialog, int which) {
            updatePermissions(KEY_APP_LIST, true);
        }
    });
    builder.setNegativeButton("禁止", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            updatePermissions(KEY_APP_LIST, false);
        }
    });
    builder.show();
}

private boolean checkSelfPermission(Context context, String
permission) {
    try {
        if (Build.VERSION.SDK_INT >= 23) {
            Method method =
Context.class.getMethod("checkSelfPermission",
                    String.class);
            return (Integer) method.invoke(context, permission)
== PackageManager.PERMISSION_GRANTED;
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return true;
}

private void updatePermissions(String permission, boolean
granted) {
    if (KEY_PHONE_STATE.equalsIgnoreCase(permission)) {

MobadsPermissionSettings.setPermissionReadDeviceID(granted);
    } else if (KEY_LOCATION.equalsIgnoreCase(permission)) {
        MobadsPermissionSettings.setPermissionLocation(granted);
    } else if (KEY_STORAGE.equalsIgnoreCase(permission)) {
        MobadsPermissionSettings.setPermissionStorage(granted);
    } else if (KEY_APP_LIST.equalsIgnoreCase(permission)) {
        MobadsPermissionSettings.setPermissionAppList(granted);
    }
    DemoSPUtils.setBoolean(this, permission, granted);
}
```

集成信通院 MSA

强烈建议集成信通院 SDK。详见：*http://msa-alliance.cn/*

接入步骤：
1. lib 引入 miit_mdid_1.0.10.aar，并在 gradle 文件内依赖该库，假如如下代码 `implementation files ('libs/miit_mdid_1.0.10.aar')`
2. 在默认 assets 文件夹内添加 supplierconfig.json 文件，如 demo 内所示。
3. 在 gradle 中，依据支持的设备类型，增加适配名单。如下

```
        ndk {
            abiFilters 'armeabi-v7a','x86','arm64-
v8a','x86_64','armeabi'
        }
        packagingOptions {

            doNotStrip "*/armeabi-v7a/*.so"
            doNotStrip "*/x86/*.so"
            doNotStrip "*/arm64-v8a/*.so"
            doNotStrip "*/x86_64/*.so"
            doNotStrip "armeabi.so"
        }
```

4. 代码混淆规则增加 **`-keep class com.bun.miitmdid.core.** {*;}`**
5. 初始化信通院 SDK，代码如下

```
        JLibrary.InitEntry(this);
```

6. 获取 OAID

```
int sdkState = MdidSdkHelper.InitSdk(getApplicationContext(),
true, new IIdentifierListener() {
    @Override
    public void OnSupport(boolean b, IdSupplier idSupplier) {
        if (idSupplier != null) {
            String oaid = idSupplier.getOAID();
        }
    }
});
```

代码接入

横幅（Banner）

简介

基本信息

横幅广告(banner 广告)位于一般位于 app 底部或顶部，横幅广告会停留在屏幕上，并可在一段时间后自动新。

适用场景：文章底部，app 底部/顶部，小说底部等。

样式


推荐 20:3 宽高比

也可选择 7：3，3：2 或 2：1

代码接入示例

```
（详细内容请参考压缩包中的代码示例）
public class BannerListActivity extends Activity {

    public static final String TAG = "BannerListActivity";

    private static final String AD_PLACE_ID_20_3 = "2015351";
    private static final String AD_PLACE_ID_7_3 = "3536891";
    private static final String AD_PLACE_ID_3_2 = "3536888";
```

```
    private static final String AD_PLACE_ID_2_1 = "3536896";

    private View mNoDataView;
    private RelativeLayout mRlAdContainer;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.banner_main);

        mNoDataView = getLayoutInflater().inflate(R.layout.no_ad_
view, null);

        mRlAdContainer = findViewById(R.id.ad_container);

        // 默认请求 http 广告，若需要请求 https 广告，请设置 AdSettings.se
tSupportHttps 为 true
        // AdSettings.setSupportHttps(true);

        // 代码设置 AppSid，此函数必须在 AdView 实例化前调用
        // AdView.setAppSid("debug");

        // 设置'广告着陆页'动作栏的颜色主题
        // 目前开放了七大主题：黑色、蓝色、咖啡色、绿色、藏青色、红色、白色
(默认) 主题
        AppActivity
                .setActionBarColorTheme(AppActivity.ActionBarColor
Theme.ACTION_BAR_BLUE_THEME);
        // 另外，也可设置动作栏中单个元素的颜色，颜色参数为四段制，0xFF(透
明度，一般填 FF)DE(红)DA(绿)DB(蓝)
        // AppActivity.getActionBarColorTheme().set[Background|Ti
tle|Progress|Close]Color(0xFFDEDADB);

        // 重要：请填上您的广告位 ID，代码位错误会导致无法请求到广告
        // 创建广告 View，并添加至接界面布局中
        bindBannerView(mRlAdContainer, AD_PLACE_ID_20_3, 20, 3);

    }

    public void onClick(View view) {
        // 计算横幅广告的长宽比，以及设置广告位 id
        int scaledWidth = 20;
        int scaledHeight = 3;
```

```
        String adPlaceId = AD_PLACE_ID_20_3;
        switch (view.getId()) {
            case R.id.btn_show1:
                break;
            case R.id.btn_show2:
                adPlaceId = AD_PLACE_ID_7_3;
                scaledWidth = 7;
                scaledHeight = 3;
                break;
            case R.id.btn_show3:
                adPlaceId = AD_PLACE_ID_3_2;
                scaledWidth = 3;
                scaledHeight = 2;
                break;
            case R.id.btn_show4:
                adPlaceId = AD_PLACE_ID_2_1;
                scaledWidth = 2;
                scaledHeight = 1;
                break;
            default:
                // nop
        }
        mRlAdContainer.removeAllViews();
        // 重要：请填上您的广告位 ID，代码位错误会导致无法请求到广告
        // 创建广告 View，并添加至接界面布局中
        bindBannerView(mRlAdContainer, adPlaceId, scaledWidth, sc
aledHeight);
    }

    /**
     * 创建横幅广告的 View，并添加至接界面布局中
     * 注意：只有将 AdView 添加到布局中后，才会有广告返回
      */
    private void bindBannerView(final RelativeLayout yourOriginal
Layout,
                                String adPlaceId, int scaleWidth, in
t scaleHeight) {
        AdView adView = new AdView(this, adPlaceId);
        // 设置监听器
        adView.setListener(new AdViewListener() {
            @Override
            public void onAdSwitch() {
                Log.w(TAG, "onAdSwitch");
            }
```

```
            @Override
            public void onAdShow(JSONObject info) {
                // 广告已经渲染出来
                Log.w(TAG, "onAdShow " + info.toString());
            }

            @Override
            public void onAdReady(AdView adView) {
                // 资源已经缓存完毕，还没有渲染出来
                Log.w(TAG, "onAdReady " + adView);
            }

            @Override
            public void onAdFailed(String reason) {
                Log.w("", "onAdFailed " + reason);
                RelativeLayout.LayoutParams rllp = new RelativeLay
out
                        .LayoutParams(ViewGroup.LayoutParams.MATCH_
PARENT, ViewGroup.LayoutParams.WRAP_CONTENT);
                rllp.addRule(RelativeLayout.CENTER_IN_PARENT);
                yourOriginalLayout.addView(mNoDataView, rllp);
            }

            @Override
            public void onAdClick(JSONObject info) {
                // Log.w(TAG, "onAdClick " + info.toString());
            }

            @Override
            public void onAdClose(JSONObject arg0) {
                Log.w(TAG, "onAdClose");
            }
        });

        DisplayMetrics dm = new DisplayMetrics();
        ((WindowManager) getSystemService(Context.WINDOW_SERVIC
E)).getDefaultDisplay().getMetrics(dm);
        int winW = dm.widthPixels;
        int winH = dm.heightPixels;
        int width = Math.min(winW, winH);
        int height = width * scaleHeight / scaleWidth;
        // 将 adView 添加到父控件中(注：该父控件不一定为您的根控件，只要该
控件能通过 addView 能添加广告视图即可)
```

```
        RelativeLayout.LayoutParams rllp = new RelativeLayout.Lay
outParams(width, height);
        rllp.addRule(RelativeLayout.ALIGN_PARENT_TOP);
        yourOriginalLayout.addView(adView, rllp);
    }

}
```

主要 API

AdView

方法名	方法介绍
public AdView(android.content.Context context, AdSize adSize, java.lang.String adPlaceId)	构造函数，参数: context - 视图运行的上下文环境，建议使用 Activity 的 context adSize - 枚举类型，用于指定横幅或方形 adPlaceId - 广告位 id
setListener(final AdViewListener listener)	设置回调监听
destroy()	当不需要 Banner 广告实例时用于主动释放资源,调用后实例销毁，不可以再调用

AdViewListener

方法名	方法介绍
onAdFailed(String reason)	广告加载失败，error 对象包含了错误码和错误信息
onAdReady(AdView adView)	广告加载成功回调，表示广告相关的资源已经加载完毕，Ready To Show
onAdShow(JSONObject info)	当广告展现时发起的回调
onAdClick(JSONObject info)	当广告点击时发起的回调
onAdClosed(JSONObject arg0)	当广告关闭时调用

接入注意事项

- 推荐您将 Banner 的宽高比固定为 20：3 以获得最佳的广告展示效果
- Banner 给开发者的回调（AdViewListener）全部执行在主线程中（异步回调）
- 尽量复用广告实例，不要实例化过多的广告实例，当广告实例不再使用时务必调用 destory 方法进行资源释放
- 百青藤平台配置广告位配置尺寸和前端渲染尺寸需要严格保持一致。

开屏广告

简介

开屏广告在 App 启动时展现。用户可以点击广告跳转到广告落地页；或者点击右上角的“跳过”按钮，跳转到 app 内容首页。

适用场景：开屏广告会在您的应用开启时加载，展示完毕后自动关闭并进入您的应用主界面。展示时间和跳过广告样式，可以通过百青藤平台配置，前端可以自定义修改。

分类：开屏广告分为半屏和全屏，其中半屏开屏广告支持开发者控制广告容器，设置开屏底部的界面，用以展示媒体 LOGO

半屏	全屏
	

接入代码示例

```
(详细内容请参考压缩包中的代码示例)
public class RSplashActivity extends Activity {
```

```
    /**
     * 当设置开屏可点击时，需要等待跳转页面关闭后，再切换至您的主窗口。故此时需要增加 canJumpImmediately 判断。 另外，点击开屏还需要在 onResume 中
     * 调用 jumpWhenCanClick 接口。
     */
    private boolean canJumpImmediately = false;
    private TextView mSplashHolder;
    // 控制开屏广告在落地页关闭后自动关闭，并进入到媒体的应用主页
    private boolean mExitAfterLp;

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

        Intent intent = getIntent();
        boolean needAppLogo = true;
        if (intent != null) {
            needAppLogo = intent.getBooleanExtra("need_app_logo", true);
            mExitAfterLp = intent.getBooleanExtra("exit_after_lp", false);
        }
        if (!needAppLogo) {
            findViewById(R.id.appLogo).setVisibility(View.GONE);
        }
        mSplashHolder = findViewById(R.id.splash_holder);

        PackageInfo info;
        int targetVersion = 0;
        try {
            info = getPackageManager().getPackageInfo(getPackageName(), 0);
            targetVersion = info.applicationInfo.targetSdkVersion;
        } catch (NameNotFoundException e) {
            e.printStackTrace();
        }
        // 如果 targetSDKVersion >= 23，并且运行环境在 Android6.0 版本之上，就要申请好权限。
        // 否则，只需要在这里直接调用 fetchSplashAD 接口
        if (Build.VERSION.SDK_INT >= 23 && targetVersion >= 23) {
            checkAndRequestPermission();
        } else {
```

```
            // 如果是 Android6.0 以下的机器，或者 targetSDKVersion < 23，默认在安装时获得了所有权限，可以直接调用 SDK
            fetchSplashAD();
        }
    }

    /**
     *
     * Android6.0 以上的权限适配简单示例：
     *
     * 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。
     */
    @TargetApi(Build.VERSION_CODES.M)
    private void checkAndRequestPermission() {
        List<String> lackedPermission = new ArrayList<String>();
        if (!(checkSelfPermission(this, Manifest.permission.READ_PHONE_STATE))) {
            lackedPermission.add(Manifest.permission.READ_PHONE_STATE);
        }

        if (!(checkSelfPermission(this, Manifest.permission.WRITE_EXTERNAL_STORAGE))) {
            lackedPermission.add(Manifest.permission.WRITE_EXTERNAL_STORAGE);
        }

        if (!(checkSelfPermission(this, Manifest.permission.ACCESS_COARSE_LOCATION))) {
            lackedPermission.add(Manifest.permission.ACCESS_COARSE_LOCATION);
        }

        if (lackedPermission.size() == 0) {
            // 权限都已经有了，那么直接调用 SDK
            fetchSplashAD();
        } else {
            // 请求所缺少的权限，在 onRequestPermissionsResult 中再看是否获得权限，如果获得权限就可以调用 SDK，否则不要调用 SDK。
            String[] requestPermissions = new String[lackedPermission.size()];
            lackedPermission.toArray(requestPermissions);
            requestPermissions(requestPermissions, 1000);
```

```
        }
    }

    public static boolean checkSelfPermission(Context context, St
ring permission) {
        try {
            if (Build.VERSION.SDK_INT >= 23) {
                Method method = Context.class.getMethod("checkSelf
Permission",
                        String.class);
                return (Integer) method.invoke(context, permissio
n) == PackageManager.PERMISSION_GRANTED;
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return true;
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, Strin
g[] permissions, int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permission
s, grantResults);
        if (requestCode == 1000 && hasAllPermissionsGranted(grantR
esults)) {
            fetchSplashAD();
        } 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();
        }
    }

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

```
                return false;
            }
        }
        return true;
    }

    private void fetchSplashAD() {
        // 默认请求 http 广告，若需要请求 https 广告，请设置 AdSettings.se
tSupportHttps 为 true
        // AdSettings.setSupportHttps(true);
        // 设置视频广告最大缓存占用空间(15MB~100MB),默认 30MB,单位 MB
        // SplashAd.setMaxVideoCacheCapacityMb(30);
        // adUnitContainer
        RelativeLayout adsParent = (RelativeLayout) this.findView
ById(R.id.adsRl);

        // 增加 lp 页面关闭回调，不需要该回调的继续使用原来接口就可以
        SplashLpCloseListener listener = new SplashLpCloseListene
r() {
            @Override
            public void onLpClosed() {
                Toast.makeText(RSplashActivity.this,"lp 页面关闭",To
ast.LENGTH_SHORT).show();
                // 落地页关闭后关闭广告，并跳转到应用的主页
                if (mExitAfterLp) {
                    jumpWhenCanClick();
                }
            }

            @Override
            public void onAdDismissed() {
                Log.i("RSplashActivity", "onAdDismissed");
                jumpWhenCanClick(); // 跳转至您的应用主界面
            }

            @Override
            public void onAdFailed(String arg0) {
                Log.i("RSplashActivity", arg0);
                jump();
            }

            @Override
            public void onAdPresent() {
                Log.i("RSplashActivity", "onAdPresent");
```

```
                mSplashHolder.setVisibility(View.GONE);
            }

            @Override
            public void onAdClick() {
                Log.i("RSplashActivity", "onAdClick");
                // 设置开屏可接受点击时，该回调可用
            }
        };
        // 不需要使用 lp 关闭之后回调方法的，可以继续使用该接口
//        SplashAdListener listener = new SplashAdListener() {
//            @Override
//            public void onAdDismissed() {
//                Log.i("RSplashActivity", "onAdDismissed");
//                jumpWhenCanClick(); // 跳转至您的应用主界面
//            }
//
//            @Override
//            public void onAdFailed(String arg0) {
//                Log.i("RSplashActivity", arg0);
//                jump();
//            }
//
//            @Override
//            public void onAdPresent() {
//                Log.i("RSplashActivity", "onAdPresent");
//            }
//
//            @Override
//            public void onAdClick() {
//                Log.i("RSplashActivity", "onAdClick");
//                // 设置开屏可接受点击时，该回调可用
//            }
//        };

        AdSettings.setSupportHttps(false);
        String adPlaceId = "2058622"; // 重要：请填上您的广告位 ID，代码位错误会导致无法请求到广告
        // 如果开屏需要支持 vr,needRequestVRAd(true)
//        SplashAd.needRequestVRAd(true);
        // 等比缩小放大，裁剪边缘部分
//        SplashAd.setBitmapDisplayMode(BitmapDisplayMode.DISPLAY_MODE_CENTER_CROP);
        new SplashAd(this, adsParent, listener, adPlaceId, true);
```

```
    }

    private void jumpWhenCanClick() {
        Log.d("RSplashActivity", "this.hasWindowFocus():" + this.
hasWindowFocus());
        if (canJumpImmediately) {
            this.startActivity(new Intent(RSplashActivity.this, Ba
iduSDKDemo.class));
            this.finish();
        } else {
            canJumpImmediately = true;
        }

    }

    /**
     * 不可点击的开屏，使用该 jump 方法，而不是用 jumpWhenCanClick
     */
    private void jump() {
        this.startActivity(new Intent(RSplashActivity.this, Baidu
SDKDemo.class));
        this.finish();
    }

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

    @Override
    protected void onResume() {
        super.onResume();
        if (canJumpImmediately) {
            jumpWhenCanClick();
        }
        canJumpImmediately = true;
    }

}
```

主要 API

SplashAD

方法名	方法介绍
SplashAd(android.content.Context context, android.view.ViewGroup viewParent, SplashAdListener rlistener, java.lang.String adPlaceId)	构造方法，广告相关状态会通过 adListener 通知开发者。参数说明： context - 上下文，建议使用 Activity 的 context viewParent - ad view 父组件 rlistener - 回调接口 adPlaceId - 代码位 ID，必须正确填写否则无法请求到广告
SplashAd(android.content.Context context, android.view.ViewGroup viewParent, SplashAdListener rlistener, java.lang.String adPlaceId, boolean canClick)	构造方法，参数: context - 上下文，建议使用 Activity 的 context viewParent - ad view 父组件 rlistener - 回调接口 adPlaceId - 代码位 ID，必须正确填写否则无法请求到广告 canClick - 广告是否支持点击

SplashADListener

开屏广告回调接口

方法名	方法介绍
onAdDismissed()	广告关闭
onAdPresent()	广告成功展示
onAdClick()	广告被点击
onAdFailed(String reason)	广告加载失败
onLpClose()	广告落地页关闭或点击返回键

接入注意事项

- 1.开屏广告需嵌入应用启动页 Activity 中。
- 2.开屏广告支持自定义跳过按钮，需在百青藤平台配置广告位设置。
- 3. SplashAd 中 canClick 参数强烈建议设置为 true，否则影响填充。
- 4.开屏广告物料为图片或视频。

插屏广告

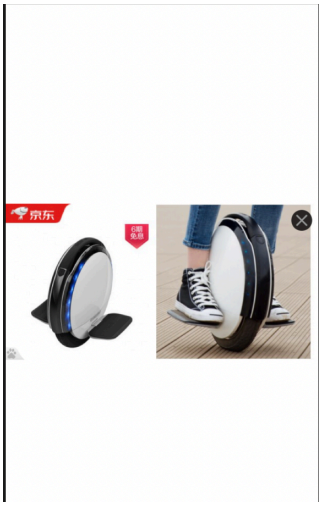
简介

基本信息

插屏广告是弹出式广告，用户点击广告前往广告落地页。可以选择点击关闭/倒计时关闭等多种形式

适用场景：在应用停顿点，适合投放这类广告

分类：插屏广告分为全屏插屏，前贴插屏和暂停插屏，具体种类可以在 SDK 进行选择：

前贴（弹窗）	暂停（弹窗）	全屏（竖屏）
		

代码接入示例

```
(详细内容请参考压缩包中的代码示例)
public class InterstitialAdActivity extends Activity implements I
nterstitialAdListener {
    // 重要：请填上您的广告位 ID，代码位错误会导致无法请求到广告
    private static final String YOUR_AD_PLACE_ID = "2403633";
    private static final String YOUR_VIDEO_AD_PLACE_ID = "2058626
";

    private InterstitialAd mInterAd;            // 插屏广告实例，支持
单例模式
    private String mAdType = "interAd";         // 插屏广告的类型，De
mo 使用，避免重复创建广告实例
    private EditText mAdPlaceIdView;            // 广告位 id
```

```
    private boolean isAdForVideo = false;       // 视频插屏广告
    private RelativeLayout mVideoAdLayout;      // 展示视频插屏的布局
    private RelativeLayout.LayoutParams reLayoutParams;
    private boolean isQianTiePianAd = true;     // 前贴片广告 or 暂停页广告

    private Button mButton;

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

        // 默认请求 http 广告，若需要请求 https 广告，请设置 AdSettings.setSupportHttps 为 true
        // AdSettings.setSupportHttps(true);

        // 设置 appsid
        mAdPlaceIdView = findViewById(R.id.edit_apid);
        mAdPlaceIdView.setText(YOUR_AD_PLACE_ID);

        initAdControlView();

        // 视频插屏广告：初始化展示布局
        final RelativeLayout parentLayout = findViewById(R.id.parent_interstitial);
        mVideoAdLayout = new RelativeLayout(this);
        reLayoutParams = new RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.WRAP_CONTENT,
                RelativeLayout.LayoutParams.WRAP_CONTENT);
        reLayoutParams.addRule(RelativeLayout.CENTER_IN_PARENT);
        parentLayout.addView(mVideoAdLayout, reLayoutParams);

        // 创建插屏广告实例
        createInterstitialAd();

        mButton = this.findViewById(R.id.btn_interstitial);
        mButton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View arg0) {
                if (mInterAd.isAdReady()) {
                    showAd();
                } else {
```

```
                    createInterstitialAd();
                    loadAd();
                }
            }
        });

        // 提前加载插屏广告
        loadAd();
    }

    /**
     * 创建广告实例，支持：插屏、前贴片、暂停页
     */
    private void createInterstitialAd() {
        String adPlaceId = mAdPlaceIdView.getText().toString();
        if (isAdForVideo) {
            if (isQianTiePianAd) {
                if (!"qianTiePian".equals(mAdType)) {
                    // 创建前贴片广告
                    mInterAd = new InterstitialAd(this, AdSize.Inte
rstitialForVideoBeforePlay, adPlaceId);
                    mInterAd.setListener(this);
                    mAdType = "qianTiePian";
                }
            } else {
                // 创建暂停页广告
                if (!"zanTingYe".equals(mAdType)) {
                    mInterAd = new InterstitialAd(this, AdSize.Inte
rstitialForVideoPausePlay, adPlaceId);
                    mInterAd.setListener(this);
                    mAdType = "zanTingYe";
                }
            }
        } else {
            // 创建插屏广告
            if (mInterAd == null || !"interAd".equals(mAdType)) {
                mInterAd = new InterstitialAd(this, adPlaceId);
                mInterAd.setListener(this);
                mAdType = "interAd";
            }
        }
    }

    /**
```

```
     * 加载广告
     */
    private void loadAd() {
        if (isAdForVideo) {
            int width = getValueById(R.id.edit_width);
            int height = getValueById(R.id.edit_height);
            if (width <= 0 || height <= 0) {
                width = 600;
                height = 500;
            }
            reLayoutParams.width = width;
            reLayoutParams.height = height;
            mInterAd.loadAdForVideoApp(width, height);
        } else {
            mInterAd.loadAd();
        }
    }

    /**
     * 展现广告
     */
    private void showAd() {
        if (isAdForVideo) {
            mInterAd.showAdInParentForVideoApp(this, mVideoAdLayou
t);
        } else {
            mInterAd.showAd(this);
        }
    }

    @Override
    public void onAdClick(InterstitialAd arg0) {
        Log.i("InterstitialAd", "onAdClick");
    }

    @Override
    public void onAdDismissed() {
        Log.i("InterstitialAd", "onAdDismissed");
        mButton.setText("加载插屏广告");
        if (!isAdForVideo) {
            loadAd();
        }
    }
```

```
    @Override
    public void onAdFailed(String arg0) {
        Log.i("InterstitialAd", "onAdFailed");
    }

    @Override
    public void onAdPresent() {
        Log.i("InterstitialAd", "onAdPresent");
    }

    @Override
    public void onAdReady() {
        Log.i("InterstitialAd", "onAdReady");
        mButton.setText("展现插屏广告");
    }

    private void initAdControlView() {
        CheckBox mCheckVideo = findViewById(R.id.check_video_ad);
        RadioGroup videoType = findViewById(R.id.video_ad_type);
        videoType.setOnCheckedChangeListener(new RadioGroup.OnChe
ckedChangeListener() {
            @Override
            public void onCheckedChanged(RadioGroup group, int che
ckedId) {
                isQianTiePianAd = (checkedId == R.id.qian_tie_pia
n);
            }
        });

        final LinearLayout adSizeEditView = findViewById(R.id.ad_
xy_size);

        final RadioButton btn1 = findViewById(R.id.qian_tie_pia
n);
        final RadioButton btn2 = findViewById(R.id.zan_ting_ye);
        mCheckVideo.setOnCheckedChangeListener(new CompoundButto
n.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonVie
w, boolean isChecked) {
                isAdForVideo = isChecked;
                mAdPlaceIdView.setText(isAdForVideo ? YOUR_VIDEO_A
D_PLACE_ID : YOUR_AD_PLACE_ID);
```

```
                btn1.setVisibility(isChecked ? View.VISIBLE : Vie
w.INVISIBLE);
                btn2.setVisibility(isChecked ? View.VISIBLE : Vie
w.INVISIBLE);
                adSizeEditView.setVisibility(isAdForVideo ? View.V
ISIBLE : View.GONE);
            }
        });
    }

    private int getValueById(int viewId) {
        EditText editText = findViewById(viewId);
        String value = editText.getText().toString();
        if (value.length() > 0) {
            try {
                return Integer.valueOf(value);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return 0;
    }
}
```

主要 API

InterstitialAD

方法名	方法介绍
InterstitialAd(android.content.Context context, AdSize adSize, java.lang.String adPlaceId)	构造函数。参数: context - 上下文，建议使用 Activity 的 context adSize - 插屏类型，请填写以 Interstitial 开头的类型

方法名	方法介绍
	adPlaceId - 代码位 ID，如果为空则不能请求到广告
loadAd()	加载广告
showAd()	展示插屏广告
loadAdForVideoApp(int width, int height)	加载视频暂停插屏或视频前贴插屏
setListener	设置 listener
showAdInParentForVideoApp(android.app.Activity activity, android.widget.RelativeLayout parent)	在指定容器中显示视频前贴片插屏广告或者视频暂停插屏广告

InterstitialAdListener

方法名	方法介绍
onAdFailed(java.lang.String reason)	广告加载失败
onADReady ()	插屏广告加载完毕
onAdPresent ()	插屏广告展开时回调
onAdClick()	插屏广告点击时回调
onAdDismissed()	插屏广告关闭时回调

您需要注意以下几点：

1. 插屏广告同一条广告不能重复曝光（一次 load 只能 show 一次）
2. 插屏广告支持视频物料，但是需要在百青藤平台修改广告位设置，或新建广告位
3. 视频前贴插屏有倒计时，结束后自动关闭
4. 视频暂停插屏无倒计时，可以自定大小，width 和 height 以及传入的 parent 必须大小合理，不可太小

信息流广告

简介

基本信息

SDK API 提供特定物料组合的广告物料，开发者可以自由使用和渲染广告，在自定场景内使用。使用 BaiduNativeManager 和 BaiduNative 均可请求信息流广告，两者接口使用方式不同，推荐使用 BaiduNativeManager。注意信息流需要手动发送曝光和点击计费。BaiduNativeManager 和 BaiduNative 发送曝光的接口不同，注意区分。

示例

一条常规的广告包含如下字段，如有需要您可以创建以下字段相关视图：

1. Logo
2. 广告大图
3. 广告标题

4. 广告描述

除此之外，开发者还可以拿到三小图广告的图片集合。

代码接入示例

(详细内容请参考 Demo 中的代码示例)

```
public class FeedAdActivity extends Activity {
    private static final String TAG = FeedAdActivity.class.getSimpleName();
    private static final String BIG_PIC_AD_PLACE_ID = "2058628";
        // 大图+ICON+描述
    private static final String SANTU_AD_PLACE_ID = "5887568";
      // 三图
    private static final String H5_LISTVIEW_AD_PLACE_ID = "3143845";    // 三图模版
    private static final String H5_LUNBO_AD_PLACE_ID = "2403624";
        // 轮播图+文字
    private static final String FEED_SMART_OPT_AD_PLACE_ID = "6481012"; // 信息流智能优选

    private static final int NATIVE_BIG_PIC = 1;
    private static final int NATIVE_SANTU = 2;
    private static final int NATIVE_SMART_OPT = 5;

    List<NativeResponse> nrAdList = new ArrayList<NativeResponse>(); // 媒体自渲染广告使用
    private NativeAdAdapter mAdapter;
    private RefreshAndLoadMoreView mRefreshLoadView;
    private int mAdPatternType = NATIVE_BIG_PIC;
    private BaiduNativeManager mBaiduNativeManager;
    private boolean isRefreshing = false;

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

        initView();
```

```
        // 默认请求大图广+ICON 样式
//        fetchNativeAd(BIG_PIC_AD_PLACE_ID);

        /**
         * Step 1. 创建 BaiduNative 对象，参数分别为： 上下文 context，广告位 ID
         * 注意：请将 adPlaceId 替换为自己的广告位 ID
         * 注意：信息流广告对象，与广告位 id 一一对应，同一个对象可以多次发起请求
         */
        mBaiduNativeManager = new BaiduNativeManager(this, BIG_PIC_AD_PLACE_ID);

        mRefreshLoadView.setRefreshing(true);
        loadFeedAd();

    }

    private void initView() {

        mRefreshLoadView = findViewById(R.id.refresh_container);
        mRefreshLoadView.setLoadAndRefreshListener(new RefreshAndLoadMoreView.LoadAndRefreshListener() {
            @Override
            public void onRefresh() {
                loadFeedAd();
                // fetchNativeAd(getNativeAdPlaceId());
            }

            public void onLoadMore() {
                loadFeedAd();
                // fetchNativeAd(getNativeAdPlaceId());
            }
        });

        ListView list = mRefreshLoadView.getListView();
        mAdapter = new NativeAdAdapter(this);
        list.setAdapter(mAdapter);

        list.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
```

```
                if (position < nrAdList.size()) {
                    NativeResponse nrAd = nrAdList.get(position);
                    nrAd.handleClick(view);
                    // mRefreshLayout.setRefreshing(false); // 点击
取消刷新
                }
            }
        });
    }

    private void loadFeedAd() {
        // 若与百度进行相关合作，可使用如下接口上报广告的上下文
        RequestParameters requestParameters = new RequestParamete
rs.Builder()
                .downloadAppConfirmPolicy(RequestParameters.DOWNLO
AD_APP_CONFIRM_ONLY_MOBILE)
                // 用户维度：用户性别，取值：0-unknown，1-male，2-femal
e
                .addExtra(ArticleInfo.USER_SEX, "1")
                // 用户维度：收藏的小说 ID，最多五个 ID，且不同 ID 用'/分隔
'
                .addExtra(ArticleInfo.FAVORITE_BOOK, "这是小说的名称
1/这是小说的名称 2/这是小说的名称 3")
                // 内容维度：小说、文章的名称
                .addExtra(ArticleInfo.PAGE_TITLE, "测试书名")
                // 内容维度：小说、文章的 ID
                .addExtra(ArticleInfo.PAGE_ID, "10930484090")
                // 内容维度：小说分类，一级分类和二级分类用'/'分隔
                .addExtra(ArticleInfo.CONTENT_CATEGORY, "一级分类/
二级分类")
                // 内容维度：小说、文章的标签，最多 10 个，且不同标签用'/分
隔'
                .addExtra(ArticleInfo.CONTENT_LABEL, "标签 1/标签 2/
标签 3")
                .build();

        mBaiduNativeManager.loadFeedAd(requestParameters, new Bai
duNativeManager.FeedAdListener() {
            @Override
            public void onNativeLoad(List<NativeResponse> nativeRe
sponses) {
                Log.i(TAG, "onNativeLoad:" +
                                        (nativeResponses != null ? n
ativeResponses.size() : null));
```

```
                // 一个广告只允许展现一次，多次展现、点击只会计入一次
                if (nativeResponses != null && nativeResponses.siz
e() > 0) {
                    // 刷新时重制数据
                    if (mRefreshLoadView.isRefreshing()) {
                        nrAdList.clear();
                    }
                    nrAdList.addAll(nativeResponses);
                    mAdapter.notifyDataSetChanged();
                }
                mRefreshLoadView.onLoadFinish();
            }

            @Override
            public void onNativeFail(NativeErrorCode errorCode) {
                Log.w(TAG, "onNativeFail reason:" + errorCode.name
());
                mRefreshLoadView.onLoadFinish();
            }

            @Override
            public void onVideoDownloadSuccess() {

            }

            @Override
            public void onVideoDownloadFailed() {

            }

            @Override
            public void onLpClosed() {

            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.feed_menu, menu);
        return true;
    }

    @Override
```

```
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if (id == R.id.feed_patterns) {
            return true;
        }
        switch (id) {
            case R.id.big_pic:  // 大图+ICON+描述
                mAdPatternType = NATIVE_BIG_PIC;
                mRefreshLoadView.setRefreshing(true);
//                fetchNativeAd(BIG_PIC_AD_PLACE_ID);
                mBaiduNativeManager = new BaiduNativeManager(this,
 BIG_PIC_AD_PLACE_ID);
                loadFeedAd();
                break;
            case R.id.santu:    // 信息流三图
                mAdPatternType = NATIVE_SANTU;
                mRefreshLoadView.setRefreshing(true);
//                fetchNativeAd(SANTU_AD_PLACE_ID);
                mBaiduNativeManager = new BaiduNativeManager(this,
 SANTU_AD_PLACE_ID);
                loadFeedAd();
                break;
            case R.id.h5_listview:  // 三图模版
                Intent h5List = new Intent(FeedAdActivity.this, Fe
edH5ListViewActivity.class);
                h5List.putExtra("adPlaceId", H5_LISTVIEW_AD_PLACE_
ID);
                startActivity(h5List);
                break;
            case R.id.h5_lunbo:     // 轮播图+文字模版
                Intent h5Lunbo = new Intent(FeedAdActivity.this, F
eedH5LunBoActivity.class);
                h5Lunbo.putExtra("adPlaceId", H5_LUNBO_AD_PLACE_I
D);
                startActivity(h5Lunbo);
                break;
            case R.id.smart_opt:    // 信息流智能优选
                mAdPatternType = NATIVE_SMART_OPT;
                mRefreshLoadView.setRefreshing(true);
//                fetchNativeAd(FEED_SMART_OPT_AD_PLACE_ID);
                mBaiduNativeManager = new BaiduNativeManager(this,
 FEED_SMART_OPT_AD_PLACE_ID);
                loadFeedAd();
                break;
```

```
            default:
                // nop
        }
        return super.onOptionsItemSelected(item);
    }

    public String getNativeAdPlaceId() {
        String adPlaceID = "";
        switch (mAdPatternType) {
            case NATIVE_BIG_PIC:
                adPlaceID = BIG_PIC_AD_PLACE_ID;
                break;
            case NATIVE_SANTU:
                adPlaceID = SANTU_AD_PLACE_ID;
                break;
            case NATIVE_SMART_OPT:
                adPlaceID = FEED_SMART_OPT_AD_PLACE_ID;
                break;
            default:
                // nop
        }
        return adPlaceID;
    }

    class NativeAdAdapter extends BaseAdapter {
        LayoutInflater inflater;

        public NativeAdAdapter(Context context) {
            super();
            inflater = (LayoutInflater) context.getSystemService(C
ontext.LAYOUT_INFLATER_SERVICE);
        }

        @Override
        public int getCount() {
            return nrAdList.size();
        }

        @Override
        public NativeResponse getItem(int position) {
            return nrAdList.get(position);
        }

        @Override
```

```
        public long getItemId(int position) {
            return position;
        }

        @Override
        public View getView(int position, View convertView, ViewGr
oup parent) {
            final NativeResponse nrAd = getItem(position);
            switch (mAdPatternType) {
                case NATIVE_BIG_PIC:
                    // 大图广告样式
                    if (convertView == null || ((Integer) convertVi
ew.getTag()) != NATIVE_BIG_PIC) {
                        convertView = inflater.inflate(R.layout.fee
d_native_listview_ad_row, null);
                        convertView.setTag(NATIVE_BIG_PIC);
                    }
                    AQuery aq = new AQuery(convertView);
                    aq.id(R.id.native_icon_image).image(nrAd.getIc
onUrl(), false, true);
                    aq.id(R.id.native_main_image).image(nrAd.getIm
ageUrl(), false, true);
                    aq.id(R.id.native_text).text(nrAd.getDesc());
                    aq.id(R.id.native_title).text(nrAd.getTitle
());
                    aq.id(R.id.native_brand_name).text(nrAd.getBra
ndName());
                    aq.id(R.id.native_adlogo).image(nrAd.getAdLogo
Url(), false, true);
                    aq.id(R.id.native_baidulogo).image(nrAd.getBai
duLogoUrl(), false, true);
                    // nrAd.isDownloadApp() ? "下载" : "查看";
                    Log.i(TAG, "AD button [" + getBtnText(nrAd) +
"]: " + nrAd.getTitle());
                    break;
                case NATIVE_SANTU:
                    // 三图广告样式
                    if (convertView == null || ((Integer) convertVi
ew.getTag()) != NATIVE_SANTU) {
                        convertView = inflater.inflate(R.layout.fee
d_native_santu_item, null);
                        convertView.setTag(NATIVE_SANTU);
                    }
                    AQuery aq1 = new AQuery(convertView);
```

```
                    aq1.id(R.id.iv_title).text(nrAd.getTitle());
                    aq1.id(R.id.iv_icon).image(nrAd.getIconUrl());
                    List<String> picUrls = nrAd.getMultiPicUrls();
                    if (picUrls != null && picUrls.size() > 2) {
                        aq1.id(R.id.iv_main1).image(picUrls.get
(0));
                        aq1.id(R.id.iv_main2).image(picUrls.get
(1));
                        aq1.id(R.id.iv_main3).image(picUrls.get
(2));
                    }
                    aq1.id(R.id.iv_desc).text(nrAd.getDesc());
                    aq1.id(R.id.iv_baidulogo).image(nrAd.getBaiduL
ogoUrl());
                    aq1.id(R.id.iv_adlogo).image(nrAd.getAdLogoUrl
());
                    aq1.id(R.id.iv_brandname).text(nrAd.getBrandNa
me());
                    break;
                case NATIVE_SMART_OPT:
                    // 信息流智能优选
                    if (convertView == null || ((Integer) convertVi
ew.getTag()) != NATIVE_SMART_OPT) {
                        convertView = LayoutInflater.from(parent.ge
tContext())
                                .inflate(R.layout.feed_native_listv
iew_item, null);
                        convertView.setTag(NATIVE_SMART_OPT);
                    } else {
                        ((ViewGroup) convertView).removeAllViews();
                    }
                    final FeedNativeView newAdView = new FeedNative
View(FeedAdActivity.this);
                    if (newAdView.getParent() != null) {
                        ((ViewGroup) newAdView.getParent()).removeV
iew(newAdView);
                    }
                    final NativeResponse ad = (NativeResponse) nrAd
List.get(position);
                    newAdView.setAdData((XAdNativeResponse) ad);
                    if (position == 2) {
                        // 自定义参数，调整广告效果，需要在 setAdData()后
调用
```

```
                        StyleParams params = new StyleParams.Builde
r()
                                .setTitleFontColor(getResources().g
etColor(R.color.blue))
                                .setTitleFontSizeSp(16)
                                .setTitleFontTypeFace(Typeface.crea
te(Typeface.MONOSPACE, Typeface.BOLD_ITALIC))
                                .setImageBackground(getResources().
getDrawable(R.drawable.no_ad_icon))
                                .build();
                        newAdView.changeViewLayoutParams(params);
                    }
                    ((ViewGroup) convertView).addView(newAdView);
                default:
                    // nop
            }

            // 警告：调用该函数来发送展现，勿漏！
            // recordImpression() 与 BaiduNative 配套使用
            // nrAd.recordImpression(convertView);

            /**
             * registerViewForInteraction()与 BaiduNativeManager 配
套使用
             * 警告：调用该函数来发送展现，勿漏！
             */
            nrAd.registerViewForInteraction(convertView, new Nativ
eResponse.AdInteractionListener() {
                @Override
                public void onAdClick() {
                    Log.i(TAG, "onAdClick:" + nrAd.getTitle());
                }

                @Override
                public void onADExposed() {
                    Log.i(TAG, "onADExposed:" + nrAd.getTitle());
                }

                @Override
                public void onADStatusChanged() {
                    Log.i(TAG, "onADStatusChanged:" + getBtnText(n
rAd));
                }
            });
```

```
            return convertView;
        }
    }

    // 下载状态及下载的进度
    private String getBtnText(NativeResponse nrAd) {
        if (nrAd == null) {
            return "";
        }
        if (nrAd.isDownloadApp()) {
            int status = nrAd.getDownloadStatus();
            if (status >= 0 && status <= 100) {
                return "下载中: " + status + "%";
            } else if (status == 101) {
                return "点击安装";
            } else if (status == 102) {
                return "继续下载";
            } else if (status == 103) {
                return "点击启动";
            } else if (status == 104) {
                return "重新下载";
            } else {
                return "点击下载";
            }
        }
        return "查看详情";
    }
}
```

主要 API

BaiduNativeManager（推荐使用）

方法名	方法介绍
BaiduNativeManager(Context context, String adPlaceId)	构造函数，参数: context：上下文，建议使用 Activity 的 context adPlaceId：广告位 ID

方法名	方法介绍
BaiduNativeManager(Context context, String adPlaceId, boolean isCacheVideo, int timeoutMillis)	构造函数，参数: context：上下文，建议使用 Activity 的 context adPlaceId：广告位 ID baiduNativeNetworkListener 接口回调 isCacheVideo：是否缓存视频，默认 true timeoutMillis：请求超时时间
void setCacheVideoOnlyWifi(boolean cacheVideoOnlyWifi)	是否仅在 wifi 环境下缓存视频
void loadFeedAd(RequestParameters requestParameters, FeedAdListener feedAdListener)	请求广告 **requestParameters**：请求参数 **feedAdListener**：请求回调

BaiduNative

方法名	方法介绍
BaiduNative(Context context, String adPlaceId, BaiduNativeNetworkListener baiduNativeNetworkListener)	构造函数，参数: context：上下文，建议使用 Activity 的 context adPlaceId：广告位 ID baiduNativeNetworkListener 接口回调

方法名	方法介绍
public BaiduNative(Context context, String adPlaceId, BaiduNativeNetworkListener baiduNativeNetworkListener, boolean isCacheVideo, int timeout)	构造函数，参数: context： 上下文，建议使用 Activity 的 context adPlaceId: 广告位 ID baiduNativeNetworkListener: 接口回调 isCacheVideo: 是否缓存视频，默认 true timeout: 请求超时时间，ms
makeRequest()	请求广告

NativeResponse

单条原生/信息流广告，包含广告有物料信息

方法名	方法介绍
registerViewForInteraction(View view, AdInteractionListener interactionListener)	发送展现，计费必传（BaiduNativeManager 调用）
recordImpression()	发送展现，计费必传（BaiduNative 调用）
handleClick()	发送点击，计费必传
getTitle()	标题
getDesc()	描述
getIconUrl()	Icon URL

方法名	方法介绍
getImageUrl();	大图 URL
getMainPicWidth()	大图宽
getMainPicHeight()	大图高
getBrandName()	品牌名/应用名
getAdLogoUrl()	广告 字样图片 URL
getBaiduLogoUrl()	百度 图片 URL（百香果）
isDownloadApp ()	是否为下载广告
isAdAvailable(Context context)	广告是否有效
getAppSize()	应用大小
isAutoPlay()	是否自动播放
getAppPackage()	应用包名
getMultiPicUrls()	三图 URL
onStart()	视频开始播放
onError()	视频错误
onComplete()	视频完成
onClose()	视频关闭

方法名	方法介绍
onClickAd()	视频点击
onFullScreen()	视频全屏
getVideoUrl()	视频 URL
getDuration()	视频时长
getMaterialType()	物料类型
getHtmlSnippet()	模板 html 代码
getWebView();	模板 webview
getStyleType()	智能优选类型
getContainerWidth()	智能优选容器宽
getContainerHeight()	智能优选容器高
getContainerSizeType()	智能优选尺寸单位（像素/比例）
getECPMLevel()	获取价格标签，若广告位未配置则返回空
pauseAppDownload()	若为下载广告，则可以暂停相应的下载任务

方法名	方法介绍
resumeAppDownload()	若为下载广告，则可以恢复相应的下载任务
isNonWifiAutoPlay()	非 wifi 下是否为自动播放
getDownloadStatus()	下载状态和下载进度

BaiduNativeNetworkListener

原生广告回调

方法名	方法介绍
onNativeLoad(final List<NativeResponse> nativeResponses)	广告请求成功
onNativeFail(final NativeErrorCode errorCode)	广告请求失败

BaiduNativeEventListener

广告事件回调，~~已废弃~~

方法名	方法介绍
onImpressionSended()	广告曝光
onClicked()	广告被点击

NativeADEventListener

广告事件回调

方法名	方法介绍
onADExposed(NativeResponse response)	广告曝光回调

NativeDownloadListener

下载状态回调

方法名	方法介绍
void onADStatusChanged(NativeResponse response)	下载状态回调

FeedLpCloseListener

点击行为回调

方法名	方法介绍
onLpClosed()	落地页关闭
onAdClick(NativeResponse response)	广告被点击

BaiduNativeH5AdViewManager

模板组件生成器

方法名	方法介绍
getBaiduNativeH5AdView(Context context, BaiduNativeAdPlacement placement, int resid)	生成 html 模板 View

BaiduNativeH5AdView

模板组件

方法名	方法介绍
BaiduNativeH5AdView(Context context, int resid)	初始化 View
BaiduNativeH5AdView(Context context, AttributeSet attrs)	初始化 View
BaiduNativeH5AdView(Context context, AttributeSet attrs, int defStyle)	初始化 View

方法名	方法介绍
isAdDataLoaded()	广告数据已加载
getAdPlacement()	获取广告位ID
setAdPlacement(BaiduNativeAdPlacement placement)	设置广告位ID
makeRequest(RequestParameters requestParameters)	请求
recordImpression()	检查曝光
setAdPlacementData(Object placement)	配置广告数据

BaiduNativeH5EventListner

模板组件回调

方法名	方法介绍
onAdClick()	模板广告被点击
onAdFail(String code)	模板广告加载失败
onAdShow()	模板广告已展现
onAdDataLoaded()	模板广告已加载

说明：

1. 广告从拉取到使用超过 30 分钟，将作为无效广告。利用 isAdAvailable 方法验证。
2. 广告数据渲染完毕，即将展示时需调用 recordImpression 方法，否则将不会产生曝光记录，也无法进行计费。
3. 广告位的返回物料组合配置/返回模板，当 APP 广告上线后，切勿随意变更，确保前端不会发生崩溃。且前端开发需要对物料进行判空校验，以确定物料是否满足渲染条件，若有不符需要抛弃广告，防止 crash。
4. 需要开发者手动调用 recordImpression 检查渲染以及 handleClick 触发点击计费。

信息流广告用户参数回传

为提升信息流广告的转化效率的效率，开发者可以申请（请联系相关的 PM 进行申请）与百度进行更加深入的合作，在请求广告时上报广告的上下文信息即广告所处页面的相关信息，由此返回更加符合预期的广告提升广告的转化率。

维度	字段名称	描述
用户维度	sex	用户性别
	fav_book	收藏图书 ID，取最近的 5 个，用"/"分隔
内容维度	page_title	页面标题：小说书名&文章标题
	page_content_id	内容 ID：小说 ID&文章 ID
	page_content_category	内容分类：小说一级、二级分类&文章所属 tab 页等，用"/"分割不同分类
	page_content_label	内容标签：小说&文章关键词标签等，最多 10 个，用"/"分割不同分类

智能优选

简介

原生广告/信息流接入新模式，SDK 支持直接渲染，媒体 app 可以直接挂载，对接成本更低。

需在百青藤平台，信息流样式控制选择【智能优选】，从 8 种样式中挑选。

且 SDK 支持直接渲染，开发者直接挂载就可以使用。

目前 SDK 支持全部样式，包含视频。样式和 NativeResponse 中 getStyleType 一致。

getStyleType	样式
28	大图（上图下文）
29	大图（上文下图）
30	大图 logo
33	左图右文
34	右图左文
35	三图图文
36	三图图文+logo
37	视频

不强制使用 SDK 组件渲染智能优选。开发者仍可以拿到物料 NativeResponse，自主渲染全部效果，使用 recordImpression 展现以及 handleClick 触发点击，和普通信息流使用方式一样。

注意：只有智能优选的广告（NativeResponse），才可以正常使用和渲染智能优选组件。使用时需要和百青藤平台信息流广告位对齐。

非智能优选信息流广告，getStyleType()无法获取如上类型值。

代码接入示例

```
(详细内容请参考压缩包中的代码示例)
case NATIVE_SMART_OPT:
                    // 信息流智能优选
                    if (convertView == null) {
                        convertView =
LayoutInflater.from(parent.getContext())
                                .inflate(R.layout.feed_native_listv
iew_item, null);
                    } else {
                        ((ViewGroup) convertView).removeAllViews();
                    }
                    final FeedNativeView newAdView = new
FeedNativeView(FeedAdActivity.this);
                    if (newAdView.getParent() != null) {
                        ((ViewGroup)
newAdView.getParent()).removeView(newAdView);
                    }
                    final NativeResponse ad = (NativeResponse)
nrAdList.get(position);
                    newAdView.setAdData((XAdNativeResponse) ad);
                    if (position == 2) {
                        // 自定义参数，调整广告效果，需要在 setAdData()后
调用
                        StyleParams params = new
StyleParams.Builder()
                                .setTitleFontColor(getResources().g
etColor(R.color.blue))
                                .setTitleFontSizeSp(16)
                                .setTitleFontTypeFace(Typeface.crea
te(Typeface.MONOSPACE, Typeface.BOLD_ITALIC))
                                .setImageBackground(getResources().
getDrawable(R.drawable.no_ad_icon))
                                .build();
                        newAdView.changeViewLayoutParams(params);
                    }
                    ((ViewGroup) convertView).addView(newAdView);
```

主要 API

FeedNativeView

智能优选组件

方法名	方法介绍
FeedNativeView(Context context)	构造函数，建议使用 Activity 的 context
FeedNativeView(Context context, AttributeSet attrs)	构造函数，建议使用 Activity 的 context
FeedNativeView(Context context, AttributeSet attrs, int defStyleAttr)	构造函数，建议使用 Activity 的 context
setAdData(XAdNativeResponse adResponse)	设置广告数据
changeViewLayoutParams(Object parameters)	改变组件内元素的 layoutParams
getAdContainerWidth()	真实宽度
getAdContainerHeight()	真实高度
getContainerView()	容器

尺寸设置参数如下

```
    /** icon 容器的宽度 单位 dp */
    public int mIconWidthDp;
    /** icon 容器的高度 单位 dp */
    public int mIconHeightDp;
    /** icon 距离顶部的间距 单位 dp */
```

```
    public int mIconTopDp;
    /** icon距离底部的间距 单位dp */
    public int mIconBottomDp;
    /** icon距离左边的间距 单位dp */
    public int mIconLeftDp;
    /** icon距离右边的间距 单位dp */
    public int mIconRightDp;

    /** title距离左边的间距 单位dp */
    public int mTitleLeftDp;
    /** title距离右边的间距 单位dp */
    public int mTitleRightDp;
    /** title距离顶部的间距 单位dp */
    public int mTitleTopDp;
    /** title距离底部的间距 单位dp */
    public int mTitleBottomDp;
    /** title字体的大小 单位sp */
    public int mTitleFontSizeSp;
    /** title字体的颜色 */
    public int mTitleFontColor;
    /** title字体样式 包含字体、粗体、斜体等设置 */
    public Typeface mTitleFontTypeFace;

    // (第一张图组件)
    /** 图片容器的宽度 单位dp */
    public int mFirstPicWidthDp;
    /** 图片容器的高度 单位dp */
    public int mFirstPicHeightDp;
    /** 图片容器距离顶部的间距 单位dp */
    public int mFirstPicTopDp;
    /** 图片容器距离底部的间距 单位dp */
    public int mFirstPicBottomDp;
    /** 图片容器距离左边的间距 单位dp */
    public int mFirstPicLeftDp;
    /** 图片容器距离右边的间距 单位dp */
    public int mFirstPicRightDp;

    // (第二张图组件)
    /** 图片容器的宽度 单位dp */
    public int mTwoPicWidthDp;
    /** 图片容器的高度 单位dp */
    public int mTwoPicHeightDp;
    /** 图片容器距离顶部的间距 单位dp */
    public int mTwoPicTopDp;
```

```
    /** 图片容器距离底部的间距 单位 dp */
    public int mTwoPicBottomDp;
    /** 图片容器距离左边的间距 单位 dp */
    public int mTwoPicLeftDp;
    /** 图片容器距离右边的间距 单位 dp */
    public int mTwoPicRightDp;

    // (第三张图组件)
    /** 图片容器的宽度 单位 dp */
    public int mThreePicWidthDp;
    /** 图片容器的高度 单位 dp */
    public int mThreePicHeightDp;
    /** 图片容器距离顶部的间距 单位 dp */
    public int mThreePicTopDp;
    /** 图片容器距离底部的间距 单位 dp */
    public int mThreePicBottomDp;
    /** 图片容器距离左边的间距 单位 dp */
    public int mThreePicLeftDp;
    /** 图片容器距离右边的间距 单位 dp */
    public int mThreePicRightDp;

    /** 图片容器的默认背景颜色 */
    public int mImageBackgroundColor;
    /** 图片容器的默认背景 */
    public Drawable mImageBackground;

    ** 底部品牌字样的左边距 单位 dp */
    public int mBrandLeftDp;
    /** 底部品牌字样的底部边距 单位 dp */
    public int mBrandBottomDp;
    /** 底部品牌字样的字体大小 单位 sp */
    public int mBrandFontSizeSp;
    /** 底部品牌字样体的颜色 */
    public int mBrandFontColor;
    /** 底部品牌字样字体样式 包含字体、粗体、斜体等设置 */
    public Typeface mBrandFontTypeFace;
```

说明：

1. 初始化组件，以屏幕高度的宽度重新计算高度宽度，所以需要调用

   getAdContainerWidth 和 getAdContainerHeight 获取真实宽高。

2. 如果 NativeResponse 不是智能优选的广告，可能无法渲染。

3. 修改过参数后，组件高度有可能改变，建议重新获取组件高度

4. 目前增加字体和字体颜色可供修改。且可以配置图片背景色或默认背景图

信息流视频

简介

基本信息

特定的信息流元素，需要在百青藤平台新建或修改广告位的物料组合。使用信息流 NativeRespone 初始化和播放。提供基本的渲染能力和视频播放能力，带有默认图和尾帧效果。

代码接入示例

```
(详细内容请参考压缩包中的代码示例)
public class FeedNativeVideoActivity extends Activity implements
AdapterView.OnItemClickListener {

    private static final String TAG = "FeedNativeVideoActivity";
    /** 广告位 id */
    private String mAdPlaceId = "2362913";
    private NativeResponse mNativeAd;
    private ListView mListView;
    private VideoAdapter mVideoAdapter;
    private XNativeView mFirstNativeView;
    private List<Object> mList = new ArrayList<Object>();
    private float density;

    @TargetApi(Build.VERSION_CODES.JELLY_BEAN)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP
_SCREEN_ON);
        setContentView(R.layout.feed_native_video_list);
        DisplayMetrics metric = new DisplayMetrics();
        getWindowManager().getDefaultDisplay().getMetrics(metri
c);
        density = metric.density;
        mListView = (ListView) findViewById(R.id.feed_native_vide
o_list);
        mVideoAdapter = new VideoAdapter();
        mListView.setAdapter(mVideoAdapter);
        mListView.setOnItemClickListener(this);
        fetchAd(this);
    }

    private void fetchAd(Activity activity) {
        /**
         * 创建广告对象，参数分别为：
         * （1）activity 上下文
         * （2）mAdPlaceId 广告位 id
         * （3）mAdListener 广告回调监听
         */
        BaiduNative baidu = new BaiduNative(activity, mAdPlaceId,
mAdListener);

        // 创建 requestParameters 对象，传递请求参数
        RequestParameters requestParameters = new RequestParamete
rs.Builder()
                .setWidth((int) (640 * density))
                .setHeight((int) (360 * density))
                .downloadAppConfirmPolicy(
                        RequestParameters.DOWNLOAD_APP_CONFIRM_ONLY
_MOBILE) // 用户点击下载类广告时，是否弹出提示框让用户选择下载与否
                .build();
        // 请求广告
        baidu.makeRequest(requestParameters);
    }

    @Override
    public void onItemClick(AdapterView<?> parent, View view, int
 position, long id) {
        if (mNativeAd != null) {
            // 调用 sdk 提供的点击 api
```

```
            mNativeAd.handleClick(view);
        }
    }

    /** 广告回调监听 */
    private BaiduNative.FeedLpCloseListener mAdListener = new Bai
duNative.FeedLpCloseListener() {

        @Override
        public void onLpClosed() {
            Log.i(TAG, "onLpClosed: ");
        }

        @Override
        public void onAdClick() {
            Log.i(TAG, "onAdClick: ");
        }

        @Override
        public void onNativeFail(NativeErrorCode arg0) {
            Toast.makeText(FeedNativeVideoActivity.this, "没有收到
视频广告，请检查", Toast.LENGTH_LONG).show();
        }

        @Override
        public void onNativeLoad(List<NativeResponse> arg0) {
            Log.i(TAG, "广告加载成功: ");
            if (arg0 != null && arg0.size() > 0) {
                // 这里每次都取第一条广告来做展示,模拟多条广告;实际开发过
程中需要开发者自己处理
                mNativeAd = arg0.get(0);
                mList.addAll(arg0);
                mVideoAdapter.notifyDataListChanged(mList);
            }
        }

    };

    class VideoAdapter extends BaseAdapter {

        private List<Object> mDataList = new ArrayList<Object>();

        public void notifyDataListChanged(List<Object> dataList)
{
```

```
            mDataList.clear();
            mDataList.addAll(dataList);
            notifyDataSetChanged();
        }

        @Override
        public int getCount() {
            return mDataList.size();
        }

        @Override
        public Object getItem(int position) {
            return mDataList.get(position);
        }

        @Override
        public long getItemId(int position) {
            return position;
        }

        @Override
        public View getView(int position, View convertView, ViewGr
oup parent) {
            Log.e(TAG, "getView=" + position);
            ViewHolder viewHolder = null;
            // 视频广告类型，媒体可以自定义，SDK 提供了渲染视频的控件，该控
件同时区分是大图还是视频，开发者只需要传入广告对象即可
            if (convertView == null) {
                convertView = LayoutInflater.from(parent.getContex
t())
                        .inflate(R.layout.feed_native_video, null);
                viewHolder = new ViewHolder(convertView);
                convertView.setTag(viewHolder);
            } else {
                viewHolder = (ViewHolder) convertView.getTag();
            }
            final AQuery aq = new AQuery(convertView);
            final NativeResponse ad = (NativeResponse) mDataList.g
et(position);
            aq.id(viewHolder.mIcon).image(ad.getIconUrl(),
                    false, true);
            aq.id(viewHolder.mDescription).text(ad.getDesc());
            aq.id(viewHolder.mTitle).text(ad.getTitle());
```

```
            aq.id(viewHolder.mBaiduLog).image(ad.getBaiduLogoUrl
()); 
            aq.id(viewHolder.mAdLog).image(ad.getAdLogoUrl());
            // 设置广告数据给视频播放组件
            viewHolder.mNativeView.setNativeItem(ad);
            // 设置该监听，在点击播放按钮之后，用户知道当前播放的是哪个视频
组件，拿到当前播放的视频组件，可以主动控制视频的播放和暂停（根据媒体自己的业
务场景）
            viewHolder.mNativeView.setNativeViewClickListener(new
 XNativeView.INativeViewClickListener() {
                @Override
                public void onNativeViewClick(XNativeView nativeVi
ew) {
                    Log.e(TAG, "当前播放的视频组件是=" + nativeView);
                }
            });
            // 发送展现，保证每条广告都有被展现的机会
            ad.recordImpression(convertView);
            // 让第一个视频尝试自动播放（如果 mssp 上配置为自动播放，那么调
用该方法会直接播放该视频）
            if (mFirstNativeView == null && position == 0) {
                mFirstNativeView = viewHolder.mNativeView;
                // 自动播放 api
                mFirstNativeView.render();
            }
            return convertView;
        }

        private class ViewHolder {
            TextView mTitle;
            TextView mDescription;
            ImageView mAdLog;
            ImageView mBaiduLog;
            ImageView mIcon;
            XNativeView mNativeView;

            public ViewHolder(View convertView) {
                mTitle = (TextView) convertView.findViewById(R.id.
native_title);
                mDescription = (TextView) convertView.findViewById
(R.id.native_text);
                mNativeView = (XNativeView) convertView.findViewBy
Id(R.id.videoview);
```

```
                mBaiduLog = (ImageView) convertView.findViewById
(R.id.native_baidulogo);
                mAdLog = (ImageView) convertView.findViewById(R.i
d.native_adlogo);
                mIcon = (ImageView) convertView.findViewById(R.id.
native_icon_image);
            }
        }
    }

    /**
     * 可见性区域检测
     *
     * @param view                  需要检测的 View
     * @param minPercentageViewed 最小百分比 50 百分之 50
     *
     * @return 是否满足可见性检测
     */
    private static boolean isVisible(final View view, final int m
inPercentageViewed) {
        if (view == null || view.getVisibility() != View.VISIBLE
                || view.getParent() == null) {
            return false;
        }
        Rect mClipRect = new Rect();

        if (!view.getGlobalVisibleRect(mClipRect)) {
            // Not visible
            return false;
        }

        final long visibleViewArea = (long) mClipRect.height()
                * mClipRect.width();
        final long totalViewArea = (long) view.getHeight() * view.
getWidth();

        if (totalViewArea <= 0) {
            return false;
        }

        return 100 * visibleViewArea >= minPercentageViewed * tota
lViewArea;
    }
```

```
}
```

主要 API

XNativeView

信息流视频组件

方法名	方法介绍
XNativeView(Context context)	构造函数，建议使用 Activity 的 context
XNativeView(Context context, AttributeSet attrs)	构造函数，建议使用 Activity 的 context
XNativeView(Context context, AttributeSet attrs, int defStyleAttr)	构造函数，建议使用 Activity 的 context
setNativeItem(NativeResponse currentNativeItem)	设置原生广告
setNativeViewClickListener(INativeViewClickListener listener)	设置回调
play()	初始化播放
render()	自动播放
resume()	恢复播放
pause()	暂停

方法名	方法介绍
stop()	销毁
handleCover()	展示浮层

VideoCacheListener

视频缓存回调

方法名	方法介绍
onVideoDownloadSuccess()	视频缓存成功
onVideoDownloadFailed()	视频缓存失败

说明：

前后台切换，自动暂停。

小视频

简介

基本信息

当开发者，需要为全屏小视频增加全屏视频广告/全屏图文广告时，可以选择接入小视频。小视频广告也是原生/信息流广告的一种定制场景扩展

代码接入示例

```
(详细内容请参考压缩包中的代码示例)
public class FeedPortraitVideoActivity extends Activity {

    private static final String TAG = "FeedPortraitVideo";
    // 代码位 id
    private String mAdPlaceId = "6164562";
    /**
     * 提供的视频组件，整个视频区域不可以点击
     */
    private RelativeLayout mVideoRl;
    private RelativeLayout.LayoutParams videoLp;
    private FeedPortraitVideoView mFeedVideoView;
    /**
     * 媒体接入的时候，可以把 title,icon,desc,adlogol 等等设置为可点击的
(按自己产品的设计去开发)
     * 这儿只是示例设置点击之后，如何调用相关的 api
     */
    private TextView mTitle;
    // 控制音量
    private ImageView mVolume;
    private TextView mBrandName;
    private SwipeRefreshLayout mRefreshLayout;
    // 提供的视频组件默认是有声音播放，媒体可以调用设置静音与否的 api;
    private Boolean mIsMute = false;
    private NativeResponse mNativeAd;

    @TargetApi(Build.VERSION_CODES.JELLY_BEAN)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // 需要在视频播放过程中保持屏幕常亮，所以需要设置这个
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP
_SCREEN_ON);
        setContentView(R.layout.feed_portrait_video);
        mVideoRl = findViewById(R.id.feed_portrait_video);
        mFeedVideoView = new FeedPortraitVideoView(this);
        videoLp = new RelativeLayout.LayoutParams(RelativeLayout.
LayoutParams.MATCH_PARENT, RelativeLayout
                .LayoutParams.MATCH_PARENT);
        mVideoRl.addView(mFeedVideoView, videoLp);
```

```
        mFeedVideoView.setFeedPortraitListener(new IFeedPortraitL
istener() {
            @Override
            public void playCompletion() {
                // 视频播放完成
                Log.i(TAG, "playCompletion==");
            }

            @Override
            public void playError() {
                // 播放错误
            }

            @Override
            public void playRenderingStart() {
                // 视频开始播放第一帧
                Log.i(TAG, "playRenderingStart==");
            }
        });
        mTitle = findViewById(R.id.iv_title);
        mBrandName = findViewById(R.id.iv_brandname);
        mVolume = findViewById(R.id.test_volume);
        mVolume.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mIsMute = !mIsMute;
                // 调用设置静音的 api
                mFeedVideoView.setVideoMute(mIsMute);
                mVolume.setImageResource(mIsMute ? R.mipmap.volume
_close : R.mipmap.volume_open);
            }
        });

        /**
         * 说明：媒体划走 view 两种处理方式：
         * （1）划走时候调用 mFeedVideoView.stop()，释放播放器资源，切换
到前台之后可以重新设置数据进行播放播放，可以这样调用 api：
         if (mFeedVideoView != null) {
         // (XAdNativeResponse) mNativeAd 这个广告数据可以是已经播放过
的，也可以是已经新请求 OK 的广告数据
         mFeedVideoView.setAdData((XAdNativeResponse) mNativeAd);
         mFeedVideoView.play();
         }
         */
```

```
        // 媒体可以复用同一个组件 view，这个组件 view 接收广告数据（(XAdNativeResponse) mNativeAd）进行重新播放，
        Button stopAndRestart = findViewById(R.id.stop_Restart);
        stopAndRestart.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // 释放播放器资源
                mFeedVideoView.stop();
                if (mFeedVideoView != null) {
                    // 传入当前想播放的广告数据,如果是视频就开始准备播放器
                    mFeedVideoView.setAdData((XAdNativeResponse) mNativeAd);
                    // 提供的视频组件默认是有声音播放，调用设置静音的 api
                    mFeedVideoView.setVideoMute(mIsMute);
                    // 调用 play 方法，开始播放
                    mFeedVideoView.play();
                }
            }
        });

        mRefreshLayout = findViewById(R.id.refresh_container);
        mRefreshLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {
            @Override
            public void onRefresh() {
                fetchAd();
            }
        });
        /**
         * 请求广告数据，
         * 备注：如果需要的话，可以一次请求返回多条广告数据（mssp 后台可配置）
         */
        fetchAd();

        ImageView backBtn = findViewById(R.id.btn_back);
        backBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                onBackPressed();
            }
        });
    }
```

```
    @Override
    protected void onPause() {
        super.onPause();
        // 暂停播放的时候需要调用 pause
        if (mFeedVideoView != null) {
            mFeedVideoView.pause();
        }
    }

    @Override
    protected void onResume() {
        super.onResume();
        // 回到前台时需要调用 resume
        if (mFeedVideoView != null) {
            mFeedVideoView.resume();
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        // 调用 stop,释放播放器资源
        if (mFeedVideoView != null) {
            mFeedVideoView.stop();
        }
    }

    private void fetchAd() {
        /**
         * @param context 上下文
         * @param mAdPlaceId 广告位 id,说明：媒体需要写自己申请好的广告
位 id
         * @param BaiduNativeNetworkListener 接收广告请求成功和失败的
回调
         */
        BaiduNative baidu = new BaiduNative(this, mAdPlaceId,
                new BaiduNative.VideoCacheListener() {

                    @Override
                    public void onLpClosed() {
                        // 落地页关闭回调
                        Toast.makeText(FeedPortraitVideoActivity.th
is, "lp 页面关闭", Toast.LENGTH_SHORT).show();
```

```
                    }

                    @Override
                    public void onAdClick() {
                        Log.i(TAG, "onAdClick: ");
                    }

                    @Override
                    public void onVideoDownloadSuccess() {
                        // 视频缓存成功
                        Log.i(TAG, "onVideoDownloadSuccess: ");
                        Toast.makeText(FeedPortraitVideoActivity.th
is, "视频缓存成功", Toast.LENGTH_SHORT).show();
                    }

                    @Override
                    public void onVideoDownloadFailed() {
                        // 视频缓存失败
                        Log.i(TAG, "onVideoDownloadFailed: ");
                        Toast.makeText(FeedPortraitVideoActivity.th
is, "视频缓存失败", Toast.LENGTH_SHORT).show();
                    }

                    @Override
                    public void onNativeFail(NativeErrorCode arg0)
 {
                        mRefreshLayout.setRefreshing(false);
                        Toast.makeText(FeedPortraitVideoActivity.th
is, "没有收到视频广告，请检查", Toast.LENGTH_SHORT).show();
                    }

                    @Override
                    public void onNativeLoad(List<NativeResponse>
arg0) {
                        Log.i(TAG, "onNativeLoad: ");
                        mRefreshLayout.setRefreshing(false);
                        if (arg0 != null && arg0.size() > 0) {
                            // 这里每次都取第一条广告来做展示,模拟多条广
告;实际开发过程中需要开发者自己处理
                            mNativeAd = arg0.get(0);
                            runOnUiThread(new Runnable() {
                                @Override
                                public void run() {
                                    if (mFeedVideoView != null) {
```

```
                                                    // 传入当前想播放的广告数据,如果
是视频就开始准备播放器
                                                    mFeedVideoView.setAdData((XA
dNativeResponse) mNativeAd);
                                                    // 调用 play 方法，开始播放，
                                                    mFeedVideoView.play();
                                                    // 重置静音按钮的状态，保持与播放
器的状态一致
                                                    mIsMute = false;
                                                    mVolume.setImageResource(mIs
Mute
                                                            ? R.mipmap.volume_clo
se : R.mipmap.volume_open);
                                                }
                                                mBrandName.setText(mNativeAd.get
BrandName());
                                                mTitle.setText(mNativeAd.getTitl
e());
                                                mTitle.setOnClickListener(new Vi
ew.OnClickListener() {
                                                    @Override
                                                    public void onClick(View v)
{
                                                        // 触发点击操作时，需要调用该
api
                                                        mNativeAd.handleClick(mT
itle);
                                                    }
                                                });
                                            }
                                        });
                                    }
                                }

                            }, true);

        DisplayMetrics dm = new DisplayMetrics();
        ((WindowManager) getSystemService(Context.WINDOW_SERVIC
E)).getDefaultDisplay().getMetrics(dm);
        int winW = dm.widthPixels;
        int winH = dm.heightPixels;
        /**
         * 配置请求广告的参数
         * @param winW 宽度
```

```
         * @param winH 高度
         * @param policy  用户点击下载类广告时，是否弹出提示框让用户选择下载与否
         */
        RequestParameters requestParameters = new RequestParameters.Builder()
                .setWidth(winW)
                .setHeight(winH)
                .downloadAppConfirmPolicy(
                        RequestParameters.DOWNLOAD_APP_CONFIRM_ONLY_MOBILE) // 用户点击下载类广告时，是否弹出提示框让用户选择下载与否
                .build();
        // 开始请求广告
        baidu.makeRequest(requestParameters);
    }
}
```

主要 API

FeedPortraitVideoView

小视频组件

方法名	方法介绍
FeedPortraitVideoView(Context context)	构造函数，建议使用 Activity 的 context
FeedPortraitVideoView(Context context, AttributeSet attrs)	构造函数，建议使用 Activity 的 context
FeedPortraitVideoView(Context context, AttributeSet attrs, int defStyleAttr)	构造函数，建议使用 Activity 的 context
setFeedPortraitListener(IFeedPortraitListener listener)	设置回调

方法名	方法介绍
setAdData(XAdNativeResponse adResponse)	设置广告数据
showNormalPic(XAdNativeResponse response)	显示普通图片
play()	播放
pause()	暂停
resume()	恢复播放
stop()	停止
setVideoMute(boolean mute)	设置静音
isPlaying()	是否正在播放
getCurrentPosition()	当前进度
getDuration()	总时长
setCanClickVideo(boolean canClick)	视频是否可点击
isShowEndFrame()	是否正显示尾帧

IFeedPortraitListener

小视频组件回调

方法名	方法介绍
playCompletion()	播放完成
playError()	播放错误

方法名	方法介绍
playRenderingStart()	开始播放

说明：

开发者可以在小视频组件内自定义广告展现样式。

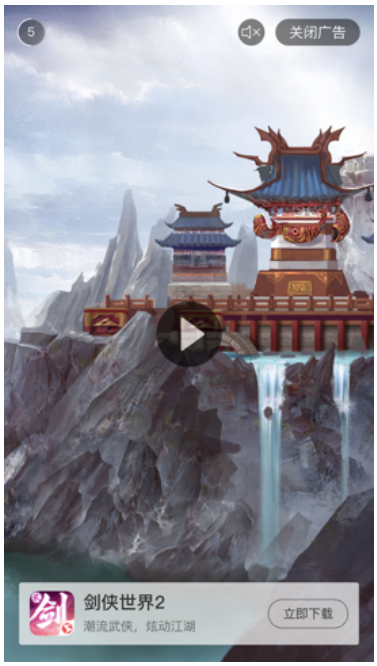
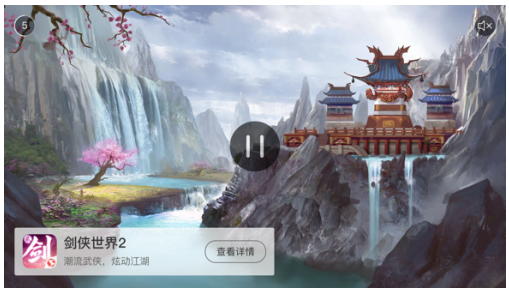
激励视频

简介

基本信息

激励视频广告是用户通过观看短视频，获取应用内奖励（免广告，任务奖励，游戏金币）等。开发者定义触发点。

分类：目前的视频包括横版及竖版 2 种样式，您可以根据需要创建对应的广告位。样式图如下：

竖屏广告	横屏广告
	

代码接入示例

```
(详细内容请参考压缩包中的代码示例)
public class RewardVideoActivity extends Activity implements Rewa
rdVideoAd.RewardVideoAdListener {

    public static final String TAG = "RewardVideoActivity";
    // 线上广告位 id
    private static final String AD_PLACE_ID = "5925490";
    public RewardVideoAd mRewardVideoAd;
    private EditText mAdPlaceIdView;
    // 测试环境的广告位 id
    //    private String mAdPlaceId = "2411590";

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.reward_video);
        initView();
    }
```

```
    @Override
    protected void onPause() {
        super.onPause();
    }

    @Override
    protected void onResume() {
        super.onResume();
    }

    @Override
    public void onVideoDownloadSuccess() {
        // 视频缓存成功
        // 说明：如果想一定走本地播放，那么收到该回调之后，可以调用 show
        Log.i(TAG, "onVideoDownloadSuccess,isReady=" + mRewardVid
eoAd.isReady());
    }

    @Override
    public void onVideoDownloadFailed() {
        // 视频缓存失败，如果想走本地播放，可以在这儿重新 load 下一条广告，
最好限制 load 次数（4-5 次即可）。
        Log.i(TAG, "onVideoDownloadFailed");
    }

    @Override
    public void playCompletion() {
        // 播放完成回调，媒体可以在这儿给用户奖励
        Log.i(TAG, "playCompletion");
    }

    @Override
    public void onAdShow() {
        // 视频开始播放时候的回调
        Log.i(TAG, "onAdShow");
    }

    @Override
    public void onAdClick() {
        // 广告被点击的回调
        Log.i(TAG, "onAdClick");
    }

    @Override
```

```
    public void onAdClose(float playScale) {
        // 用户关闭了广告
        // 说明：关闭按钮在 mssp 上可以动态配置，媒体通过 mssp 配置，可以选择广告一开始就展示关闭按钮，还是播放结束展示关闭按钮
        // 建议：收到该回调之后，可以重新 load 下一条广告,最好限制 load 次数（4-5 次即可）
        // playScale[0.0-1.0],1.0 表示播放完成，媒体可以按照自己的设计给予奖励
        Log.i(TAG, "onAdClose" + playScale);
    }

    @Override
    public void onAdFailed(String arg0) {
        // 广告失败回调 原因：广告内容填充为空；网络原因请求广告超时
        // 建议：收到该回调之后，可以重新 load 下一条广告，最好限制 load 次数（4-5 次即可）
        Log.i(TAG, "onAdFailed");

    }

    private void initView() {
        Button btn1 = this.findViewById(R.id.btn_change_orientation);
        btn1.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                int currentOrientation = getResources().getConfiguration().orientation;
                if (currentOrientation == ORIENTATION_PORTRAIT) {
                    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
                } else if (currentOrientation == ORIENTATION_LANDSCAPE) {
                    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
                }
            }
        });

        Button btn2 = this.findViewById(R.id.btn_load);
        btn2.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
                mRewardVideoAd = new RewardVideoAd(RewardVideoActi
vity.this,
                        mAdPlaceIdView.getText().toString(), Reward
VideoActivity.this, false);
                mRewardVideoAd.load();
            }

        });

        Button btn3 = this.findViewById(R.id.btn_show);
        btn3.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                if (mRewardVideoAd != null && mRewardVideoAd.isRea
dy()) {
                    mRewardVideoAd.show();
                } else {
                    Toast.makeText(RewardVideoActivity.this,
                            "请成功加载后在进行广告展示！", Toast.LENGT
H_LONG).show();
                }
            }

        });

        Button btn4 = this.findViewById(R.id.is_ready);
        btn4.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                boolean isReady = mRewardVideoAd != null && mRewar
dVideoAd.isReady();
                Toast.makeText(RewardVideoActivity.this, "可用广告:
" + isReady, Toast.LENGTH_SHORT).show();
            }
        });

        mAdPlaceIdView = findViewById(R.id.edit_apid);
        mAdPlaceIdView.setText(AD_PLACE_ID);
        mAdPlaceIdView.clearFocus();
    }
}
```

主要 API

RewardVideoAd

方法名	方法介绍
RewardVideoAd(Activity activity, String adPlaceId, RewardVideoAdListener listener)	构造函数 参数: context 上下文，建议使用 Activity 的 context adUnitId 广告位 ID listener 回调监听
RewardVideoAd(Activity activity, String adPlaceId, RewardVideoAdListener listener, boolean useSurfaceView)	构造函数 参数: context 上下文，建议使用 Activity 的 context adUnitId 广告位 ID listener 回调监听 useSurfaceView 是否使用 surfaceView 渲染，默认使用 TextureView
load()	请求一条广告

方法名	方法介绍
show()	展现激励视频
pause()	暂停
resume()	恢复
isReady()	广告是否未过期

RewardVideoAdListener

方法名	方法介绍
onAdFailed(String reason)	广告加载失败
onVideoDownloadSuccess()	视频物料缓存成功
onAdShow()	播放第一帧时回调
onAdClick()	点击时回调
onVideoDownloadFailed()	视频物料缓存失败
onAdClose(float playScale)	关闭回调，附带播放进度
playCompletion()	播放完成回调

您需要注意以下几点：

1. isReady 表示本地有一条未过期的且视频内容已经缓存完成的广告。媒体在调用 show 的时候，如果一定保证走本地播放，可以使用 isReady 进行判断

2. 如果媒体想保证都是本地播放激励视频，那么初始化参数之后就调用 load 加载广告，加载广告成功---开始缓存视频—缓存成功回调 onVideoDownloadSuccess。广告存在有效期，需要一定时间（大概 2 小时）内展现。如果广告超时未展现，调用 show()的时候会重新请求广告并在线播放
3. 单次检索的广告不支持多次展现，show()过后则认为广告过期。下次展现前建议重新预加载视频。不做预加载处理则在线请求并播放

贴片广告

简介

基本信息

贴片广告是常用于视频组件，用于给视频贴片，在视频预加载，暂停或结束时使用。

适用场景：视频组件的自然停顿点，适合投放这类广告

代码接入示例

```
(详细内容请参考压缩包中的代码示例)
public class VideoPatchAdActivity extends Activity implements Vie
w.OnClickListener {

    private static final String TAG = "VideoPatchAdActivity";
    // 竖版小视频代码位 id
    private String mAdPlaceId = "2362913";
    // 贴片视频代码位 id
```

```
    //    private String mAdPlaceId = "2058634";
    private PatchVideoNative mPrerollView;
    /** 父容器，添加视频 view */
    private RelativeLayout mParentVideoRl;
    /** 倒计时 view */
    private TextView mCountDownView;
    /** 倒计时的刷新间隔，单位 ms */
    private static final int INTERVAL = 500;
    private final Handler mCountdownHandler = new Handler();
    private View mCloseAdView;
    private View mIntervalView;
    private boolean mNeedHandlePlayer = true;
    private String mMaterialType;
    private Timer mTimer = new Timer();
    private static final int PIC_MATERIAL_DURATION = 5 * 1000;
    private static final int PIC_INTERVAL = 1000;
    private View mNoAdView;

    @TargetApi(Build.VERSION_CODES.JELLY_BEAN)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // 需要在视频播放过程中保持屏幕常亮，所以需要设置这个
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP
_SCREEN_ON);
        setContentView(R.layout.patch_ad);
        mParentVideoRl = findViewById(R.id.video_parent_view);
        mNoAdView = getLayoutInflater().inflate(R.layout.no_ad_vi
ew, null);
        DisplayMetrics dm = new DisplayMetrics();
        ((WindowManager) getSystemService(Context.WINDOW_SERVIC
E)).getDefaultDisplay().getMetrics(dm);
        int winW = dm.widthPixels;
        int height = winW * 9 / 16;
        FrameLayout.LayoutParams rllp = new FrameLayout.LayoutPar
ams(winW, height);
        mParentVideoRl.setLayoutParams(rllp);
        fetchAd();
    }

    private void initView() {
        if (mCloseAdView != null) {
            mParentVideoRl.removeView(mCloseAdView);
            mCloseAdView = null;
```

```
        }

        mCloseAdView = getLayoutInflater().inflate(R.layout.close
_skip_ad, null);
        mCountDownView = mCloseAdView.findViewById(R.id.timer);
        mIntervalView = mCloseAdView.findViewById(R.id.interval_v
iew);
        TextView skipAd = mCloseAdView.findViewById(R.id.skip_a
d);
        skipAd.setOnClickListener(this);
        ImageView closeAd = mCloseAdView.findViewById(R.id.close_
ad);
        closeAd.setOnClickListener(this);
        RelativeLayout.LayoutParams closeLp =
                new RelativeLayout.LayoutParams(RelativeLayout.Lay
outParams.WRAP_CONTENT,
                        RelativeLayout.LayoutParams.WRAP_CONTENT);
        closeLp.addRule(RelativeLayout.ALIGN_PARENT_RIGHT);
        closeLp.addRule(RelativeLayout.ALIGN_PARENT_TOP);
        int top = dp2px(26);
        int right = dp2px(15);
        closeLp.setMargins(0, top, right, 0);
        mParentVideoRl.addView(mCloseAdView, closeLp);
    }

    private void hideCountDownView() {
        if (mCountDownView != null) {
            mCountDownView.setVisibility(View.GONE);
        }
        if (mIntervalView != null) {
            mIntervalView.setVisibility(View.GONE);
        }
    }

    /** 请求广告 */
    private void fetchAd() {
        mPrerollView = new PatchVideoNative(this, mAdPlaceId, mPar
entVideoRl,
                new PatchVideoNative.IPatchVideoNativeListener() {
                    @Override
                    public void onAdLoad(String type) {
                        // sdk默认播放非静音，如果想默认静音播放，可以这
么设置
                        // mPrerollView.setVideoMute(true);
```

```
                        // 广告请求成功 type 表示物料类型,目前支持图片（"
normal"),视频物料（"video"）;
                        Log.i(TAG, "onAdLoad,广告请求成功");
                        mMaterialType = type;
                        mParentVideoRl.removeView(mNoAdView);
                    }

                    @Override
                    public void onAdFailed(NativeErrorCode errorCo
de) {
                        // 广告请求失败
                        mNeedHandlePlayer = false;
                        Log.i(TAG, "onAdFailed,广告请求失败");
                        RelativeLayout.LayoutParams rllp = new Rela
tiveLayout
                                .LayoutParams(ViewGroup.LayoutParam
s.MATCH_PARENT, ViewGroup.LayoutParams.WRAP_CONTENT);
                        rllp.addRule(RelativeLayout.CENTER_IN_PAREN
T);
                        mParentVideoRl.addView(mNoAdView, rllp);
                    }

                    @Override
                    public void playCompletion() {
                        // 视频播放完成
                        mNeedHandlePlayer = false;
                        hideCountDownView();
                        stopVideoTimer();
                        Log.i(TAG, "playCompletion,视频播放完成");
                    }

                    @Override
                    public void playError() {
                        // 视频播放错误
                        mNeedHandlePlayer = false;
                        stopVideoTimer();
                        hideCountDownView();
                        Log.i(TAG, "playError,视频播放错误");
                    }

                    @Override
                    public void onAdShow() {
                        // 视频播放第一帧或者图片渲染成功
```

```
                        Log.i(TAG, "onAdShow,视频第一帧展示，或者图片渲
染成功展示");
                        initView();
                        if ("video".equals(mMaterialType)) {
                            // 视频物料
                            startVideoTimer();
                        } else if ("normal".equals(mMaterialType))
{
                            // 图片物料，倒计时时间需要媒体自己定义，demo
默认是 5 秒
                            startPicTimer();
                        } else if ("gif".equals(mMaterialType)) {
                            // 不支持该物料格式
                        }
                    }

                    @Override
                    public void onAdClick() {
                        Log.i(TAG, "onAdClick,被点击");
                    }
                });
        DisplayMetrics dm = new DisplayMetrics();
        ((WindowManager) getSystemService(Context.WINDOW_SERVIC
E)).getDefaultDisplay().getMetrics(dm);
        int winW = dm.widthPixels;
        int winH = dm.heightPixels;
        /**
         * 配置请求广告的参数
         * @param winW 宽度
         * @param winH 高度
         * @param policy  用户点击下载类广告时，是否弹出提示框让用户选择
下载与否
         */
        RequestParameters requestParameters = new RequestParamete
rs.Builder()
                .setWidth(winW)
                .setHeight(winH)
                .downloadAppConfirmPolicy(
                        RequestParameters.DOWNLOAD_APP_CONFIRM_ONLY
_MOBILE) // 用户点击下载类广告时，是否弹出提示框让用户选择下载与否
                .build();
        // 开始请求广告
        mPrerollView.requestAd(requestParameters);
    }
```

```
    @Override
    public void onClick(View v) {
        if (mParentVideoRl != null) {
            mParentVideoRl.removeAllViews();
        }
    }

    @Override
    protected void onPause() {
        super.onPause();
        stopVideoTimer();
        stopPicTimer();
    }

    @Override
    protected void onResume() {
        super.onResume();
        startVideoTimer();
        startPicTimer();
    }

    /** 计时器开始 */
    private void startVideoTimer() {
        if (mCountDownView != null) {
            mCountdownHandler.removeCallbacksAndMessages(null);
            mCountdownHandler.postDelayed(updateVideoTimerRunnabl
e, 0);
        }
    }

    /** 计时器停止 */
    private void stopVideoTimer() {
        if (mCountDownView != null) {
            mCountdownHandler.removeCallbacksAndMessages(null);
        }
    }

    private void startPicTimer() {
        if (mPrerollView == null || mCountDownView == null || !"no
rmal".equals(mMaterialType)) {
            return;
        }
        if (mTimer == null) {
```

```
            mTimer = new Timer();
        }
        mTimer.schedule(new TimerTask()

        {
            @Override
            public void run() {
                updateProTask();
            }

        }, 0, PIC_INTERVAL);
    }

    int adTime = PIC_MATERIAL_DURATION / PIC_INTERVAL;

    private void updateProTask() {
        if (adTime >= 1 && adTime <= 5) {
            mCountdownHandler.post(new Runnable() {
                @Override
                public void run() {
                    mCountDownView.setText(String.valueOf(adTime)
+ "s");
                    // 倒计时继续
                    adTime -= 1;
                }
            });
        } else {
            mCountdownHandler.post(new Runnable() {
                @Override
                public void run() {
                    hideCountDownView();
                }
            });
            stopPicTimer();
        }
    }

    public void stopPicTimer() {
        if (mTimer != null) {
            mTimer.cancel();
            mTimer = null;
        }
    }
```

```
    /** 更新倒计时进度 */
    private Runnable updateVideoTimerRunnable = new Runnable() {
        public void run() {
            if (mPrerollView == null || mCountDownView == null ||
 !mNeedHandlePlayer) {
                return;
            }
            int position = (int) mPrerollView.getCurrentPosition
();
            int mDuration = (int) mPrerollView.getDuration();
            int adTime = 0;
            if (mDuration > 0 && position <= mDuration && position
 >= 0) {
                adTime = (int) Math.round((mDuration - position) /
 1000.0);
            }
            position = Math.min(position + 1000, mDuration);
            mCountDownView.setText(String.valueOf(adTime) + "s");
            if (position < mDuration) {
                mCountdownHandler.postDelayed(updateVideoTimerRunn
able, INTERVAL);
            }
        }
    };

    private int dp2px(float dp) {
        final float scale = getResources().getDisplayMetrics().de
nsity;
        return (int) (dp * scale + 0.5f);
    }
}
```

主要 API

PatchVideoNative

方法名	方法介绍
PatchVideoNative(Context context, String adUnitId, RelativeLayout parent, IPatchVideoNativeListener listener)	构造函数 参数: context 上下文，建议使用 Activity 的 context adUnitId 广告位 ID parent 父容器 listener 回调监听
requestAd()	请求广告
setVideoMute()	设置贴片静音
getCurrentPosition()	当前播放进度
getDuration()	总时长

IPatchAdListener

方法名	方法介绍
onAdFailed(NativeErrorCode errorCode)	广告加载失败

方法名	方法介绍
onAdLoad(String type)	插屏广告加载完毕
onAdShow()	插屏广告展现或播放第一帧时回调
onAdClick()	插屏广告点击时回调
playError()	播放错误
playCompletion()	播放完成

说明：

PatchVideoNative 开发者可自定义元素添加到组件内渲染

JSSDK

简介

基本信息

对 JS 广告的端能力支持，接入方式详见 JS 广告文档，SDK 此处仅进行功能补充。不涉及广告渲染

代码接入示例

```
(详细内容请参考压缩包中的代码示例)
public class HybridInventoryActivity extends Activity {
```

```
    public static final String AD_NUM = "AD_NUM";

    private static final String TAG = "HybridInventoryActivity";

    @SuppressLint("SetJavaScriptEnabled")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.hybrid_app_main);

        //
        final WebView webView = (WebView)
this.findViewById(R.id.webView);

webView.getSettings().setCacheMode(WebSettings.LOAD_NO_CACHE);
        webView.getSettings().setJavaScriptEnabled(true);

        // 创建百度 BaiduHybridAdManager, BaiduHybridAdManager 对象和
WebView 对象是一一对应的关系
        final BaiduHybridAdManager hybridAdManager = new
BaiduHybridAdManager();
        hybridAdManager.setBaiduHybridAdViewListener(new
BaiduHybridAdViewListener() {

            @Override
            public void onAdShow(int adSequence, String adId) {

            }

            @Override
            public void onAdFailed(int adSequence, String adId,
String reason) {

            }

            @Override
            public void onAdClick(int adSequence, String adId) {

            }
        });

        //
        //
        webView.setWebViewClient(new WebViewClient() {
```

```
            @Override
            public boolean shouldOverrideUrlLoading(final WebView
view, String url) {
                // 之前百度 SDK 将 JavaScript 代码注入到 App 的 webView
内，
                // 这段 JavaScript 代码会和 Java 代码交互，此处是
BaiduSDK 处理 JavaScript 交互
                boolean b =
hybridAdManager.shouldOverrideUrlLoading(view, url);
                if (b) {
                    return true;
                } else {
                    // App 原来的处理代码放在这里
                    ;
                }
                return false;
            }

            @Override
            public void onPageStarted(WebView view, String url,
Bitmap favicon) {
                // 一定要在 onPageStarted 调用百度广告 SDK 的
onPageStarted

hybridAdManager.onPageStarted(webView,url,favicon);

                super.onPageStarted(view, url, favicon);
            }

            @Override
            public void onPageFinished(WebView view, String url)
{
                Log.i(TAG, "onPageFinished");

                super.onPageFinished(view, url);

                hybridAdManager.injectJavaScriptBridge(webView);
            }

            @Override
            public void onReceivedError(WebView view, int
errorCode, String description, String failingUrl) {
                Log.i(TAG, "onReceivedError");
```

```
                // !!!!!!!!!!!!!!!!!!!!!!!!! 非常非常重
要 !!!!!!!!!!!!!!!!!!!!!!!!!
                // 一定要在 onReceivedError 调用百度广告 SDK 的
onReceivedError
                hybridAdManager.onReceivedError(view, errorCode,
description, failingUrl);

                super.onReceivedError(view, errorCode,
description, failingUrl);
            }

            @Override
            public void onLoadResource(WebView view, String url)
{
                Log.i(TAG, "onLoadResource");
            }

            @Override
            public void onReceivedSslError(WebView view, final
SslErrorHandler handler, SslError error) {
                Log.i(TAG, "onReceivedSslError");

            }
        });

        //设置响应 js 的 Alert()函数
        webView.setWebChromeClient(new WebChromeClient() {
            @Override
            public boolean onJsAlert(WebView view, String url,
String message, final JsResult result) {
                AlertDialog.Builder b = new
AlertDialog.Builder(HybridInventoryActivity.this);
                b.setTitle("Alert");
                b.setMessage(message);
                b.setPositiveButton(android.R.string.ok, new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog,
int which) {
                        result.confirm();
                    }
                });
                b.setCancelable(false);
                b.create().show();
```

```
                return true;
            }
            //设置响应 js 的 Confirm()函数
            @Override
            public boolean onJsConfirm(WebView view, String url,
String message, final JsResult result) {
                AlertDialog.Builder b = new
AlertDialog.Builder(HybridInventoryActivity.this);
                b.setTitle("Confirm");
                b.setMessage(message);
                b.setPositiveButton(android.R.string.ok, new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog,
int which) {
                        result.confirm();
                    }
                });
                b.setNegativeButton(android.R.string.cancel, new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog,
int which) {
                        result.cancel();
                    }
                });
                b.create().show();
                return true;
            }
        });


webView.loadUrl("http://mobads.baidu.com/ads/indexlp.html");
        // san tu
//        webView.loadUrl("
https://appin.baidu.com/page/route/game?source=1&jssdk1=6143024&
jssdk2=6143019&sdkid=5887568");
        // da tu
//        webView.loadUrl("
https://appin.baidu.com/page/route/game?source=1&jssdk1=6143024&
jssdk2=6143019&sdkid=2058628");
    }

    @Override
```

```
    protected void onPause() {
        super.onPause();
    }

    @Override
    protected void onResume() {
        super.onResume();
    }

}
```

主要 API

BaiduHybridAdManager

JSSDK 支持实例

方法名	方法介绍
injectJavaScriptBridge(WebView view)	WebviewJS 注入
setBaiduHybridAdViewListener(BaiduHybridAdViewListener listener)	设置回调
onPageStarted(WebView view, String url, Bitmap favicon)	页面启动
shouldOverrideUrlLoading(WebView view, String url)	URL 加载
onReceivedError(WebView view, int errorCode, String description, String failingUrl)	发生错误

BaiduHybridAdViewListener

JSSDK 回调

方法名	方法介绍
onAdShow(final int adSequence, final String adId)	展现成功
onAdClick(final int adSequence, final String adId)	点击
onAdFailed(final int adSequence, final String adId, final String reason)	广告失败

内容联盟

简介

当应用需要资讯内容和广告混合展示时，可以使用此产品。提供 URL，使用 Webview 渲染。

代码接入示例

```
(详细内容请参考压缩包中的代码示例)
public class CpuAdActivity extends Activity {
    // 测试 id
    private static String DEFAULT_APPSID = "d77e414";
    private WebView mWebView;
    private RelativeLayout mRootRelativeLayout;

    public enum CpuChannel {
```

```
        /**
         * 娱乐频道
         */
        CHANNEL_ENTERTAINMENT(1001),
        /**
         * 体育频道
         */
        CHANNEL_SPORT(1002),
        /**
         * 图片频道
         */
        CHANNEL_PICTURE(1003),
        /**
         * 手机频道
         */
        CHANNEL_MOBILE(1005),
        /**
         * 财经频道
         */
        CHANNEL_FINANCE(1006),
        /**
         * 汽车频道
         */
        CHANNEL_AUTOMOTIVE(1007),
        /**
         * 房产频道
         */
        CHANNEL_HOUSE(1008),
        /**
         * 热点频道
         */
        CHANNEL_HOTSPOT(1021);

        private int value;

        private CpuChannel(int value) {
            this.value = value;
        }

        public int getValue() {
            return value;
        }

    }
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.cpu);
        initSpinner();
        Button button = (Button) this.findViewById(R.id.button);
        button.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                showSelectedCpuWebPage();
            }
        });

        Button buttonblock = (Button)
this.findViewById(R.id.button_block);
        buttonblock.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                showSelectedCpuWebPageBlock();
            }
        });

    }

    /**
     * 调用 SDK 接口，获取内容联盟页面 Block 样式 URL
     */
    private void showSelectedCpuWebPageBlock() {
        CpuAdView adView = new CpuAdView(this, getAppsid(),
getBlockId(), null);
        final RelativeLayout parentLayout = (RelativeLayout)
this.findViewById(R.id.parent_block);
        RelativeLayout.LayoutParams reLayoutParams = new
RelativeLayout.LayoutParams(
                LayoutParams.WRAP_CONTENT,
LayoutParams.WRAP_CONTENT);
        reLayoutParams.addRule(RelativeLayout.CENTER_IN_PARENT);
        parentLayout.addView(adView, reLayoutParams);
    }

    /**
```

```
     * 调用 SDK 接口，获取内容联盟页面 URL
     */
    private void showSelectedCpuWebPage() {
        // 内容联盟 url 获取后只能展示一次，多次展示需要每次通过以下接口重新
获取
        // 媒体伙伴必须在 MSSP 业务端选择接入内容联盟的应用与频道类型，以便
在接入内容页中生成广告，从而获得广告收益。
        // 不进行相关操作，将无法获得内容联盟页面的广告收益。
        CpuInfoManager.getCpuInfoUrl(this, getAppsid(),
getChannel().getValue(), new UrlListener() {

            @Override
            public void onUrl(String url) {
                handleWebViewLayout(url);
            }
        });
    }

    /**
     * 初始化下拉框
     */
    private void initSpinner() {
        Spinner channelSpinner = (Spinner)
this.findViewById(R.id.channel);
        channelSpinner.setOnItemSelectedListener(new
OnItemSelectedListener() {

            @Override
            public void onItemSelected(AdapterView<?> parent,
View view, int position, long id) {
//                showSelectedCpuWebPage();
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent)
{
            }

        });
        List<SpinnerItem> list = new ArrayList<SpinnerItem>();
        list.add(new SpinnerItem("娱乐频道",
CpuChannel.CHANNEL_ENTERTAINMENT));
        list.add(new SpinnerItem("体育频道",
CpuChannel.CHANNEL_SPORT));
```

```
        list.add(new SpinnerItem("图片频道",
CpuChannel.CHANNEL_PICTURE));
        list.add(new SpinnerItem("手机频道",
CpuChannel.CHANNEL_MOBILE));
        list.add(new SpinnerItem("财经频道",
CpuChannel.CHANNEL_FINANCE));
        list.add(new SpinnerItem("汽车频道",
CpuChannel.CHANNEL_AUTOMOTIVE));
        list.add(new SpinnerItem("房产频道",
CpuChannel.CHANNEL_HOUSE));
        list.add(new SpinnerItem("热点频道",
CpuChannel.CHANNEL_HOTSPOT));
        ArrayAdapter<SpinnerItem> dataAdapter = new
ArrayAdapter<SpinnerItem>(this,
                android.R.layout.simple_spinner_item, list);

dataAdapter.setDropDownViewResource(android.R.layout.simple_spin
ner_dropdown_item);
        channelSpinner.setAdapter(dataAdapter);

    }

    /**
     * 根据内容联盟 url，渲染页面
     *
     * @param url
     */
    private void handleWebViewLayout(String url) {
        initWebView();
        final ViewGroup totalView = (ViewGroup)
this.getWindow().getDecorView();
        mRootRelativeLayout = new RelativeLayout(this);
        totalView.addView(mRootRelativeLayout, new
ViewGroup.LayoutParams(-1, -1));
        RelativeLayout.LayoutParams webViewLayoutParams = new
RelativeLayout.LayoutParams(-1, -1);
        webViewLayoutParams.topMargin = getStatusBarHeight();
        mRootRelativeLayout.addView(mWebView,
webViewLayoutParams);
        mWebView.loadUrl(url);
        Button closeButton = new Button(this);
        closeButton.setBackgroundResource(R.drawable.close_icon);
        int side = (int) (30 * getScreenDensity(this));
```

```
        RelativeLayout.LayoutParams buttonLayoutParams = new
RelativeLayout.LayoutParams(side, side);

buttonLayoutParams.addRule(RelativeLayout.ALIGN_PARENT_RIGHT);

buttonLayoutParams.addRule(RelativeLayout.ALIGN_PARENT_TOP);
        buttonLayoutParams.topMargin = getStatusBarHeight();
        closeButton.setLayoutParams(buttonLayoutParams);
        mRootRelativeLayout.addView(closeButton);
        closeButton.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                close();
            }
        });
    }

    /**
     * 关闭内容联盟页面
     */
    private void close() {
        final ViewGroup totalView = (ViewGroup)
this.getWindow().getDecorView();
        totalView.removeView(mRootRelativeLayout);
        mRootRelativeLayout.removeAllViews();
    }

    /**
     * 初始化展示内容联盟页面的 webview
     */
    private void initWebView() {
        mWebView = new WebView(this);
        mWebView.setVerticalScrollBarEnabled(false);
        mWebView.setHorizontalScrollBarEnabled(false);
        WebSettings webSettings = mWebView.getSettings();
        webSettings.setJavaScriptEnabled(true);
        // 如果是图片频道，则必须设置该接口为 true，否则页面无法展现
        webSettings.setDomStorageEnabled(true);
        mWebView.setWebViewClient(new WebViewClient() {
            @Override
            public boolean shouldOverrideUrlLoading(WebView view,
String url) {
                view.loadUrl(url);
```

```
                return true;
            }
        });

    }

    /**
     * 获取屏幕密度
     */
    public float getScreenDensity(Context context) {
        DisplayMetrics dm = new DisplayMetrics();
        ((WindowManager)
context.getSystemService(Context.WINDOW_SERVICE)).getDefaultDisp
lay().getMetrics(dm);
        return dm.density;
    }

    /**
     * 获取状态栏高度
     */
    private int getStatusBarHeight() {
        Rect frame = new Rect();

this.getWindow().getDecorView().getWindowVisibleDisplayFrame(fra
me);
        return frame.top;
    }

    /**
     * 获取 appsid
     *
     * @return
     */
    private String getAppsid() {
        return DEFAULT_APPSID;
    }

    /**
     * 获取 blockId
     *
     * @return
     */
    private String getBlockId() {
        return "2365";
```

```
    }

    /**
     * 获取频道
     *
     * @return
     */
    private CpuChannel getChannel() {
        Spinner channelSpinner = (Spinner)
this.findViewById(R.id.channel);
        SpinnerItem selectedItem = (SpinnerItem)
channelSpinner.getSelectedItem();
        return selectedItem.getChannel();
    }

    private String getValueFromEditText(int id) {
        EditText appsidEditText = (EditText)
this.findViewById(id);
        String value = appsidEditText.getText().toString();
        return value;
    }

    /*
     * 监听返回键，浏览器回退
     */
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        if (mRootRelativeLayout == null ||
mRootRelativeLayout.getParent() == null) {
            return super.onKeyDown(keyCode, event);
        }
        try {
            if (keyCode == KeyEvent.KEYCODE_BACK) {
                if (mRootRelativeLayout != null &&
mRootRelativeLayout.getParent() != null
                        && mWebView != null &&
mWebView.canGoBack()) {
                    mWebView.goBack();
                } else {
                    close();
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
        return true;
    }

    class SpinnerItem extends Object {
        /**
         * 频道名称
         */
        String text;
        /**
         * 频道 id
         */
        CpuChannel channel;

        public SpinnerItem(String text, CpuChannel cpuChannel) {
            this.text = text;
            this.channel = cpuChannel;
        }

        @Override
        public String toString() {
            return text;
        }

        CpuChannel getChannel() {
            return channel;
        }

    }
}
```

主要 API

CpuInfoManager

生成内容联盟 URL

方法名	方法介绍
getCpuInfoUrl(final Context context, final String appsid, final int channel,final UrlListener listener)	生成内容联盟 URL

UrlListener

内容联盟回调

方法名	方法介绍
onUrl(String url)	回调 URL

CpuAdView

内容联盟 webview

方法名	方法介绍
CpuAdView(Context context)	构造函数，建议使用 Activity 的 context
CpuAdView(final Context cxt, final String appsid, final String blockId, final UrlListener listener)	构造函数，建议使用 Activity 的 context

内容联盟（原生渲染）

简介

百度联盟推出的一种信息流合作产品，此产品是由资讯内容和广告（原生样式）组合而成，以“信息流”的形式在页面上呈现。目前，资讯渲染、广告渲染，都由开发者自行渲染。您如果想要接入信息流资讯广告，请联系百度联盟商务申请，目前只支持配置资讯列表页的广告样式。

代码接入示例

```
//初始化并加载广告
public class NativeCPUAdActivity extends Activity {
    private static final String TAG =
NativeCPUAdActivity.class.getSimpleName();

    private View cpuDataContainer;
    private ListView listView;
    private MyAdapter adapter;
    private Button showAd;
    private Button loadAd;
    private int mContentType = 0;  // 默认新闻类型
    private int mChannelId = 1001; // 默认娱乐频道
    private int mPageIndex = 1;
    private List<IBasicCPUData> nrAdList = new
ArrayList<IBasicCPUData>();
    private static String YOUR_AD_PLACE_ID = "10000004"; // 双引号
中填写自己的广告位 ID

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.cpu_native_list);
        initAdListView();
        initSpinner();
        loadAd = findViewById(R.id.btn_load);
        loadAd.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                loadAd(NativeCPUAdActivity.this, mPageIndex);
            }
        });
        showAd = findViewById(R.id.btn_show);
        showAd.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showAdList();
            }
        });
        showAd.setEnabled(false);
    }
```

```
    private void initAdListView() {
        cpuDataContainer = findViewById(R.id.cpuDataContainer);
        listView = (ListView)
this.findViewById(R.id.native_list_view);
        listView.setCacheColorHint(Color.WHITE);
        adapter = new MyAdapter(this);
        listView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {

            @Override
            public void onItemClick(AdapterView<?> parent, View
view, int position, long id) {
                Log.i(TAG, "NativeCPUAdActivity.onItemClick");
                IBasicCPUData nrAd = nrAdList.get(position);
                nrAd.handleClick(view);
            }
        });
        cpuDataContainer.setVisibility(View.GONE);
    }

    private void initSpinner() {
        // 内容类型
        Spinner typeSpinner = (Spinner)
this.findViewById(R.id.type);
        typeSpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {

            @Override
            public void onItemSelected(AdapterView<?> parent,
View view, int position, long id) {
                mContentType = ((SpinnerItem)
parent.getItemAtPosition(position)).getId();
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent)
{
            }

        });
        List<SpinnerItem> list1 = new ArrayList<SpinnerItem>();
        list1.add(new SpinnerItem("新闻", 0));
        list1.add(new SpinnerItem("图片", 1));
        list1.add(new SpinnerItem("视频", 2));
```

```
        ArrayAdapter<SpinnerItem> dataAdapter1 = new
ArrayAdapter<SpinnerItem>(this,
                android.R.layout.simple_spinner_item, list1);

dataAdapter1.setDropDownViewResource(android.R.layout.simple_spi
nner_dropdown_item);
        typeSpinner.setAdapter(dataAdapter1);

        // 频道类目
        Spinner channelSpinner = (Spinner)
this.findViewById(R.id.channel);
        channelSpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {

            @Override
            public void onItemSelected(AdapterView<?> parent,
View view, int position, long id) {
                mChannelId = ((SpinnerItem)
parent.getItemAtPosition(position)).getId();
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent)
{
            }

        });
        List<SpinnerItem> list2 = new ArrayList<SpinnerItem>();
        list2.add(new SpinnerItem("娱乐频道", 1001));
        list2.add(new SpinnerItem("体育频道", 1002));
        list2.add(new SpinnerItem("财经频道", 1006));
        list2.add(new SpinnerItem("汽车频道", 1007));
        list2.add(new SpinnerItem("时尚频道", 1009));
        list2.add(new SpinnerItem("文化频道", 1011));
        list2.add(new SpinnerItem("科技频道", 1013));
        ArrayAdapter<SpinnerItem> dataAdapter2 = new
ArrayAdapter<SpinnerItem>(this,
                android.R.layout.simple_spinner_item, list2);

dataAdapter2.setDropDownViewResource(android.R.layout.simple_spi
nner_dropdown_item);
        channelSpinner.setAdapter(dataAdapter2);
    }
```

```
    public void showAdList() {
        cpuDataContainer.setVisibility(View.VISIBLE);
        listView.setAdapter(adapter);
    }

    public void loadAd(Activity activity, int pageIndex) {
        /**
         * Step 1. NativeCPUManager，参数分别为： 上下文 context（必须为 Activity），广告位 ID，认证 token， CPUAdListener（监听广告请求的成功与失败）
         * 注意：请将 YOUR_AD_PLACE_ID，YOUR_AD_TOKEN 替换为自己的 ID 和 TOKEN
         */
        NativeCPUManager manager = new NativeCPUManager(activity, YOUR_AD_PLACE_ID,
                new NativeCPUManager.CPUAdListener() {
                    /**
                     * 请求广告成功，返回广告列表
                     * @param list 广告+内容数据
                     */
                    @Override
                    public void onAdLoaded(List<IBasicCPUData> list) {
                        if (list != null && list.size() > 0) {
                            nrAdList = list;
                            showAd.setEnabled(true);
                            makeToast("Load ad success!");
                        }
                    }

                    @Override
                    public void onAdError(String msg, int errorCode) {
                        Log.w(TAG, "onAdError reason:" + msg);
                        makeToast("onAdError reason:" + msg);
                    }

                    @Override
                    public void onNoAd(String msg, int errorCode) {
                        Log.w(TAG, "onNoAd reason:" + msg);
                        makeToast("onNoAd reason:" + msg);
                    }
```

```
                    @Override
                    public void onAdClick() {
                        Log.i(TAG, "onAdClick");
                    }

                    @Override
                    public void onVideoDownloadSuccess() {
                        // 预留接口
                    }

                    @Override
                    public void onVideoDownloadFailed() {
                        // 预留接口
                    }

                    @Override
                    public void onAdStatusChanged(final String
appPackageName) {
                        if (!TextUtils.isEmpty(appPackageName) &&
nrAdList != null) {
                            int size = nrAdList.size();
                            for (int pos = 0; pos < size; pos++) {
                                IBasicCPUData nrAd =
nrAdList.get(pos);
                                if (nrAd != null &&
nrAd.isDownloadApp()) {
                                    if
(appPackageName.equals(nrAd.getAppPackageName())) {
                                        adapter.updateBtnText(nrAd,
pos);
                                    }
                                }
                            }
                        }
                    }
                });

        /**
         * Step2: 构建请求参数
         */
        CPUAdRequest.Builder builder = new
CPUAdRequest.Builder();
        builder.setContentType(mContentType)
```

```
                .setDownloadAppConfirmPolicy(RequestParameters.DOW
NLOAD_APP_CONFIRM_ONLY_MOBILE);
        int[] channels = new int[] {mChannelId};
        manager.setRequestParameter(builder.build());
        manager.setRequestTimeoutMillis(10 * 1000); // 如果不设置，
则默认 5s 请求超时
        /**
         * Step2: 调用请求接口，请求广告
         */
        makeToast("Start loadAd!");
        manager.loadAd(pageIndex, channels, true);

        showAd.setEnabled(false);
    }

    class MyAdapter extends BaseAdapter {
        LayoutInflater inflater;
        AQuery aq;

        public MyAdapter(Context context) {
            super();
            inflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            aq = new AQuery(context);
        }

        @Override
        public int getCount() {
            return nrAdList.size();
        }

        @Override
        public IBasicCPUData getItem(int position) {
            return nrAdList.get(position);
        }

        @Override
        public long getItemId(int position) {
            return position;
        }

        @Override
        public View getView(int position, View convertView,
ViewGroup parent) {
```

```
            android.util.Log.i(TAG, "position is " + position);
            IBasicCPUData nrAd = getItem(position);

            ViewHolder viewHolder = null;
            if (convertView == null || convertView.getTag() ==
null) {
                viewHolder = new ViewHolder();
                convertView =
inflater.inflate(R.layout.cpu_native_list_item, null);
                viewHolder.imageAvatar =
convertView.findViewById(R.id.native_icon_image);
                viewHolder.textTitle =
convertView.findViewById(R.id.native_title);
                viewHolder.textAuthor =
convertView.findViewById(R.id.native_author);
                viewHolder.textDesc =
convertView.findViewById(R.id.native_text);
                viewHolder.imageLeft =
convertView.findViewById(R.id.image_left);
                viewHolder.imageMid =
convertView.findViewById(R.id.image_mid);
                viewHolder.imageRight =
convertView.findViewById(R.id.image_right);
                viewHolder.imageBigPic =
convertView.findViewById(R.id.image_big_pic);
                viewHolder.textCommentNum =
convertView.findViewById(R.id.comment_num);
                viewHolder.textUpdateTime =
convertView.findViewById(R.id.update_time);
                viewHolder.imageAdLogo =
convertView.findViewById(R.id.native_adlogo);
                viewHolder.imageBaiduLogo =
convertView.findViewById(R.id.native_baidulogo);
                viewHolder.textButton =
convertView.findViewById(R.id.native_button);
                convertView.setTag(viewHolder);
            } else {
                viewHolder = (ViewHolder) convertView.getTag();
            }

            String urlAvatar = nrAd.getIconUrl();
            String title = nrAd.getTitle();
            String author = nrAd.getAuthor();
            String content = nrAd.getDesc();
```

```
            int commentCounts = nrAd.getCommentCounts();
            String updateTime = nrAd.getUpdateTime();
            String bigPic = "";
            String imageLeft = "";
            String imageMid = "";
            String imageRight = "";


            List<String> imageList = nrAd.getImageUrls();
            List<String> smallImageList =
nrAd.getSmallImageUrls();
            if (smallImageList != null && smallImageList.size() >
2) {
                imageLeft = smallImageList.get(0);
                imageMid = smallImageList.get(1);
                imageRight = smallImageList.get(2);
            } else if (imageList != null && imageList.size() > 0)
{
                bigPic = imageList.get(0);
            }

            bindData2View(viewHolder.imageAvatar, urlAvatar, 2);
            bindData2View(viewHolder.textTitle, title, 1);
            bindData2View(viewHolder.textAuthor, author, 1);
            bindData2View(viewHolder.textDesc, content, 1);
            bindData2View(viewHolder.imageBigPic, bigPic, 2);
            bindData2View(viewHolder.imageLeft, imageLeft, 2);
            bindData2View(viewHolder.imageMid, imageMid, 2);
            bindData2View(viewHolder.imageRight, imageRight, 2);
            if ("ad".equals(nrAd.getType())) {
                bindData2View(viewHolder.textCommentNum, "", 1);
                bindData2View(viewHolder.textUpdateTime, "", 1);
            } else {
                bindData2View(viewHolder.textCommentNum, "评论数："
+ commentCounts, 1);
                bindData2View(viewHolder.textUpdateTime,
updateTime, 1);
            }
            bindData2View(viewHolder.imageAdLogo,
nrAd.getAdLogoUrl(), 2);
            bindData2View(viewHolder.imageBaiduLogo,
nrAd.getBaiduLogoUrl(), 2);

            String text = getBtnText(nrAd);
```

```
            aq.id(viewHolder.textButton).text(text);
            // 展现时需要调用 onImpression 上报展现
            nrAd.onImpression(convertView);
            return convertView;
        }

        private void bindData2View(View view, String data, int
dataType) {
            if (TextUtils.isEmpty(data)) {
                view.setVisibility(View.GONE);
            } else {
                view.setVisibility(View.VISIBLE);
                if (dataType == 1) {
                    aq.id(view).text(data);
                } else if (dataType == 2) {
                    // 通过 callback 的方式渲染 ImageView，避免 AQuery 直
接渲染后将 View.GONE 的控件显示出来
                    aq.id(view).image(data, false, true, 0, 0,
                            new BitmapAjaxCallback() {
                                @Override
                                protected void callback(String url,
ImageView iv, Bitmap bm, AjaxStatus status) {
                                    if (iv.getVisibility() ==
View.VISIBLE) {
                                        iv.setImageBitmap(bm);
                                    }
                                }
                            });
                }
            }
        }

        public void updateBtnText(IBasicCPUData data, int
position) {
            int firstVisiblePosition =
listView.getFirstVisiblePosition();
            int lastVisiblePosition =
listView.getLastVisiblePosition();
            if (position >= firstVisiblePosition && position <=
lastVisiblePosition) {
                View view = listView.getChildAt(position -
firstVisiblePosition);
                TextView btn =
view.findViewById(R.id.native_button);
```

```
                btn.setText(getBtnText(data));
            }
        }

        private String getBtnText(IBasicCPUData data) {
            if (data == null) {
                return "";
            }
            if (data.isDownloadApp()) {
                int status = data.getDownloadStatus();
                if (status >= 0 && status <= 100) {
                    return "下载中: " + status + "%";
                } else if (status == 101) {
                    return "点击安装";
                } else if (status == 102) {
                    return "继续下载";
                } else if (status == 103) {
                    return "点击启动";
                } else if (status == 104) {
                    return "重新下载";
                } else {
                    return "点击下载";
                }
            }
            return "查看详情";
        }
    }

    public final class ViewHolder {
        ImageView imageAvatar;
        TextView textTitle;
        TextView textAuthor;
        TextView textDesc;
        ImageView imageLeft;
        ImageView imageMid;
        ImageView imageRight;
        ImageView imageBigPic;
        TextView textCommentNum;
        TextView textUpdateTime;
        ImageView imageBaiduLogo;
        ImageView imageAdLogo;
        TextView textButton;
    }
```

```
    private void makeToast(String str) {
        Toast.makeText(this, str, Toast.LENGTH_SHORT).show();
    }

    class SpinnerItem extends Object {
        /**
         * 名称
         */
        String mName;
        /**
         * id
         */
        int mId;

        public SpinnerItem(String name, int id) {
            mName = name;
            mId = id;
        }

        @Override
        public String toString() {
            return mName;
        }

        int getId() {
            return mId;
        }

    }
}
```

主要 API

NativeCPUManager

信息流资讯广告入口，用于初始化并加载信息流资讯广告

方法名	方法介绍
NativeCPUManager(Context context, String appsid, CPUAdListener listener)	内容联盟构造函数，建议使用 Activity 的 context
loadAd(int page, int[] channels, boolean isShowAd)	请求内容（包含广告）
setRequestTimeoutMillis(int millis)	设置请求超时时间
setRequestParameter(CPUAdRequest requestParameter)	设置请求参数
setPageSize(int size)	设置单页数量

CPUAdListener

信息流回调

方法名	方法介绍
onAdLoaded(List<IBasicCPUData> list)	加载成功
onAdError(String msg, int errorCode)	加载错误
onNoAd(String msg, int errorCode)	无广告

方法名	方法介绍
onAdStatusChanged(final String appPackageName)	广告下载状态

- **IBasicCPUData**

- 加载内容联盟返回的**内容数据**，包含展示资讯所需的所有元素，以及触发曝光、点击的接口

方法名	方法介绍
getType()	获取资讯类型
onImpression(View view)	检查展现
handleClick(View view)	广告点击
getAuthor()	作者，来源
getIconUrl()	图标 url
getUpdateTime()	更新时间
getTitle()	标题
getDesc()	描述
isTop()	是否置顶
getCommentCounts()	评论数
getSmallImageUrls()	三图/多图 URL

方法名	方法介绍
getImageUrls()	大图 URL
getAdLogoUrl()	LogoURL
getBaiduLogoUrl()	Baidu 图标 URL
getThumbUrl()	缩略图 URL
getThumbWidth()	缩略图宽
getThumbHeight()	缩略图高
getDuration()	视频时长
getDownloadStatus()	下载状态
isDownloadApp()	是否为下载广告

频道列表如下：

频道 id	频道名称	备注
1022	推荐	全类目资讯/视频/图集 综合推荐
1001	娱乐	
1057	视频	全类目视频推荐
1081	热讯	

1043	健康	
1012	军事	
1042	母婴	
1035	生活	
1040	游戏	
1007	汽车	
1006	财经	
1013	科技	
1021	热点	
1068	图集	全类目图集推荐
1025	搞笑	
1002	体育	
1009	时尚	
1034	女人	
1080	本地	
1065	萌萌哒-视频	仅视频

1047	看点	
1055	动漫	
1062	小品-视频	仅视频
1036	文化	
1005	手机	
1008	房产	
1058	音乐-视频	仅视频
1059	搞笑-视频	仅视频
1060	影视-视频	仅视频
1067	游戏-视频	仅视频
1066	生活-视频	仅视频
1064	观天下-视频	仅视频
1061	娱乐-视频	仅视频
1063	社会-视频	仅视频

其他 API

SDK 默认请求 http 广告，若期望请求 https 的广告，直接调用如下接口：

```
AdSettings.setSupportHttps(true);
```

如果需要在锁屏场景下展示广告落地页，通过以下接口可以在广告展现前，全局设置落地页的属性支持锁屏下展示

```
AppActivity.canLpShowWhenLocked(boolean canShow);  // 是否锁屏下展示广告，默认为 false，广告展现前全局设置即可
```

您可以根据需要设置广告落地页的 ActionBar 颜色，目前开放了七大主题:黑色、蓝色、咖啡色、绿色、藏青色、红色、白色（默认）主题

```
AppActivity.setActionBarColorTheme(ActionBarColorTheme colorTheme);
//  广告展现前全局设置即可
```

SDK 相关问题排查

SDK 错误码

错误码	解释
0	无广告返回
0103011	应用 ID 信息错误，MSSP 未收录
0103060	应用包名信息错误，请保证注册包名和实际请求包名一致
0107003	广告位 ID 与 APPSID 不匹配
1020001	网络连接失败
1040003	请求超时
1040001	请求时使用了错误的参数，比如使用错误的广告位 ID
1040003	请求超时
3030002	缓存物料失败
3040001	广告展现标准不达标