目录

Android SDK 使用说明

基础信息

- 开发者：新义互联（北京）科技有限公司
- 开发者后台说明
- 隐私声明 / 合规使用说明
- 将XINYI SDK 添加到GroMore中介平台
- 将XINYI SDK 添加到Taku中介平台
- 将XINYI SDK 添加到ToBid中介平台
- SDK更新日志

SDK用法

注意

- 最低支持版本: API 21, Android 5.0

建议

AndroidManifest.xml 建议设置

- 在 `application` 中，加入 `android:usesCleartextTraffic="true"` ，以允许可以通过 http 接收广告资源。
- 在 `activity` 中，加入 `android:configChanges="orientation|screenSize|keyboard|keyboardHidden"` ，避免因手机旋转导致 `onCreate` 反复执行而造成广告被多次重复加载，影响用户体验。

初始化

```
public class SampleApplication extends Application {
    public void onCreate() {
        super.onCreate();
```

```
        SDK.Configuration configuration = new SDK.Configuration("媒体 Token", "应用 Token")
          .setLogEnabled(isDebug)
          .setAcquireGeoInfo(true)//建议开启
          .setAcquireFetchInstallApps(true)//建议开启
          //方式一
          .setOAID/GAID/IMEI("xxx") //必选，任意一个或多个
          .setUserAgent("xxx")//必选
          .setAndroidID("xxx")//建议设置

          //方式二
          .setAcquireOAID/GAID/IMEI(true)//必选，任意一个或多个
          .setAcquireUserAgent(true)//必选
          .setAcquireAndroidID(true)//建议开启

          //方式三
          .allowAllDeviceInfo();

        SDK.init(
            getApplicationContext(),
            configuration,
            new SDK.InitListener() {
              @Override
              public void onInitSuccess() {
                Log.d(TAG, "onInitSuccess: ");
              }

              @Override
              public void onInitFailed(Exception e) {
                Log.e(TAG, "onInitFailed: " + e);
              }
            });

        //方式四(请求广告之前设置)
        SDK.setOAID/GAID/IMEI("xxx");//必选，任意一个或多个
        SDK.setUserAgent("xxx");//必选
        SDK.setAndroidID("xxx");//建议设置

        //自定义User信息（非必须）
        SDK.setUser(User.newBuilder()
                .setId("test_id")
                .setAge(18)
                .setGender(Gender.MALE)
                .setKeywords("sdk", "test")
                .build());
    }
}
```

::: tip 注意 注意，其中的 `<媒体 Token>` 对应开发者平台 -> 账户 —> 个人信息里面的 token。 `<应用 Token>` 对应开发者平台 -> 应用信息 -> 应用列表中的您所注册应用的 token。 :::

开屏广告

```
// 测试广告位: A3B4C31EF6C3D4AA302A9748A812E88C
SplashAd splashAd = new SplashAd(activity, "广告位 TOKEN");
splashAd.setListener(new Listener() {
  @Override
  public void onRendered() {
    // 显示广告
    splashAd.show();
  }
```

```
});
splashAd.load();

...

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

插屏广告

```
// 测试广告位: B8680764759E4B31920EB98A2074EF5C
InterstitialAd interstitialAd = new InterstitialAd(activity, "广告位 TOKEN");
interstitialAd.setListener(new Listener() {
  @Override
  public void onRendered() {
    // 显示广告
    interstitialAd.show();
  }
});
interstitialAd.load();

...

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

激励视频广告

```
// 测试广告位: 5907B631E4854B53885EF6D515656E4E
RewardVideoAd rewardVideoAd = new RewardVideoAd(activity, "广告位 TOKEN");
rewardVideoAd.setListener(new RewardVideoListener() {
  @Override
  public void onRendered() {
    // 显示广告
    rewardVideoAd.show();
  }

  public void onRewarded(String data) {
    // 处理激励事件
  }
});
rewardVideoAd.load();

...

@Override
protected void onDestroy() {
  if (rewardVideoAd != null) {
    rewardVideoAd.destroy();
  }
```

```
  super.onDestroy();
}
```

其他方法说明

```
/**
 * 广告是否已加载且有效
 * @param requirePreloads, 是否要求附加资源已预加载
 */
public boolean isLoaded(boolean requirePreloads = true);

/**
 * 设置重试次数
 * @param times, 小于等于 0 时禁用自动重试
 */
public void autoRetry(int times = 5);

/**
 * 沉浸模式
 */
public void setImmersiveMode(boolean immersiveModeEnabled);
```

横幅广告

```
// 测试广告位 209A03F87BA3B4EB82BEC9E5F8B41383
BannerAd bannerAd = new BannerAd(activity, "广告位 TOKEN");
// 设置动画开启
bannerAd.setAnimationEnabled(true);
// 设置事件
bannerAd.setListener(new Listener() {
  @Override
  public void onRendered() {
    // 显示广告
    View view = bannerAd.getView();
    if (view != null) {
      FrameLayout container = findViewById(R.id.ad_container);
      container.removeAllViews();
      container.addView(view, MATCH_PARENT, 175);
    }
  }
});
bannerAd.load();

...

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

@Override
protected void onPause() {
  if (bannerAd != null) {
    bannerAd.pause();
  }
  super.onPause();
}
```

```
@Override
protected void onResume() {
  if (bannerAd != null) {
    bannerAd.resume();
  }
  super.onResume();
}
```

::: tip 注意 注意， `getView()` 在 `onLoaded` 事件就可以调用，但是在插入到 `Container` 中显示时，如果高度是 `WRAP_CONTENT` 的话，最好在 `onRendered` 事件之后再插入，否则会有一定概率显示不出来。 :::

信息流广告

```
// 测试广告位 F659DD00E21E44A7A309B3EF6BF9592C
FeedAd feedAd = new FeedAd(activity, "广告位 TOKEN");
// 设置动画开启
feedAd.setAnimationEnabled(true);
// 设置事件
feedAd.setListener(new Listener() {
  @Override
  public void onRendered() {
    // 显示广告
    View view = feedAd.getView();
    if (view != null) {
      FrameLayout container = findViewById(R.id.ad_container);
      container.removeAllViews();
      container.addView(view, MATCH_PARENT, WRAP_CONTENT);
    }
  }
});
feedAd.load();

...

@Override
protected void onPause() {
  if (feedAd != null) {
    feedAd.pause();
  }
  super.onPause();
}

@Override
protected void onResume() {
  if (feedAd != null) {
    feedAd.resume();
  }
  super.onResume();
}

@Override
protected void onDestroy() {
  if (feedAd != null) {
    //开发者主动销毁广告
    feedAd.destroy();
  }
  super.onDestroy();
}
```

原生自渲染广告

```
// 测试广告位: B93388D5C5F583EFEB795CAB8265A2CF
NativeAd nativeAd = new NativeAd(activity, "广告位 TOKEN");
nativeAd.setListener(new Listener() {
  @Override
  public void onLoaded() {
    NativeResponse response = nativeAd.getResponse();
    //自渲染
    renderNativeAdView(activity, response);
  }

  @Override
  public void onRendered() {
    NativeResponse response = nativeAd.getResponse();
    if (response == null || response.getNativeView() == null) return;
    // 开发者自定义容器
    nativeSelfRenderContainer.addView(response.getNativeView(), MATCH_PARENT, WRAP_CONTENT
  }
});
nativeAd.setVideoListener(new VideoListener() {
  @Override
  public void onVideoLoad(VideoController.Metadata metadata) {}

  @Override
  public void onVideoStart() {}

  @Override
  public void onVideoEnd() {}

  @Override
  public void onVideoError() {}

  @Override
  public void onVideoBreak() {}
});
nativeAd.load();

@Override
protected void onDestroy() {
  if (nativeAd != null) {
    //开发者主动销毁广告
    nativeAd.destroy();
  }
}
public static void renderNativeAdView(Context context, NativeResponse response) {
  if (response == null) return;
  //开发者自定义布局
  ViewGroup nativeAdView = (ViewGroup)
 ↪  LayoutInflater.from(context).inflate(R.layout.item_native_ad_view, null);
  RelativeLayout rlContainer = nativeAdView.findViewById(R.id.rl_container);
  FrameLayout adVideo = nativeAdView.findViewById(R.id.ad_video);
  ImageView adImage = nativeAdView.findViewById(R.id.ad_image);
  ImageView adIcon = nativeAdView.findViewById(R.id.ad_icon);
  TextView adTitle = nativeAdView.findViewById(R.id.ad_title);
  TextView adDesc = nativeAdView.findViewById(R.id.ad_desc);
  Button adButton = nativeAdView.findViewById(R.id.ad_button);
  ImageView adLogo = nativeAdView.findViewById(R.id.ad_logo);
  ImageView adClose = nativeAdView.findViewById(R.id.ad_close);
```

```
String title = response.getTitle();
if (!TextUtils.isEmpty(title)) {
  adTitle.setVisibility(View.VISIBLE);
  adTitle.setText(response.getTitle());
}
String description = response.getDescription();
if (!TextUtils.isEmpty(description)) {
  adDesc.setVisibility(View.VISIBLE);
  adDesc.setText(description);
}
String iconUrl = response.getIconUrl();
if (!TextUtils.isEmpty(iconUrl)) {
  Glide.with(context).load(iconUrl).into(adIcon);
  adIcon.setVisibility(View.VISIBLE);
}
if (response.hasVideo()) {//视频
  View view = response.getVideoView();
  if (view != null) {
    adVideo.setBackgroundColor(Color.BLACK);
    FrameLayout.LayoutParams layoutParams = new
↪ FrameLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
↪ ViewGroup.LayoutParams.MATCH_PARENT, Gravity.CENTER);
    adVideo.addView(view, layoutParams);
    adImage.setVisibility(GONE);
    nativeAdView.findViewById(R.id.video_container).setVisibility(View.VISIBLE);
    nativeAdView.findViewById(R.id.btn_play).setOnClickListener(v -> response.videoPlay());
    nativeAdView.findViewById(R.id.btn_pause).setOnClickListener(v -> response.videoPause());
    nativeAdView.findViewById(R.id.btn_mute).setOnClickListener(v -> response.videoMute(true));
  }
} else {//图片
  String imageUrl = response.getImageUrl();
  if (!TextUtils.isEmpty(imageUrl)) {
    Glide.with(context).load(imageUrl).into(adImage);
    adVideo.setVisibility(GONE);
  }
}
String buttonText = response.getButtonText();
if (!TextUtils.isEmpty(buttonText)) {
  adButton.setVisibility(View.VISIBLE);
  adButton.setText(buttonText);
}
String logoUrl = response.getLogoUrl();
if (!TextUtils.isEmpty(logoUrl)) {
  Glide.with(context).load(logoUrl).into(adLogo);
}

//摇一摇
View shakeView = response.getShakeView(100, 10, null);
RelativeLayout.LayoutParams tvParams =new
↪ RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.WRAP_CONTENT,
↪ RelativeLayout.LayoutParams.WRAP_CONTENT);
tvParams.addRule(RelativeLayout.CENTER_IN_PARENT);
rlContainer.addView(shakeView, tvParams);

//点击视图
ArrayList<View> clickViews = new ArrayList<>();
clickViews.add(adImage);
clickViews.add(adButton);
clickViews.add(adIcon);
clickViews.add(adTitle);
clickViews.add(adDesc);
```

```
  clickViews.add(adVideo);

  //关闭视图
  ArrayList<View> closeViews = new ArrayList<>();
  closeViews.add(adClose);
  ViewGroup nv = response.getNativeView();
  nv.addView(nativeAdView);

  //重要：注册点击和关闭视图
  response.registerViews(nv, clickViews, closeViews);
}
```

事件

标准事件 Listener

```
public interface Listener {
    // 请求广告前触发。
    void onBeforeRequest();
    // 当广告加载完毕后触发。
    void onLoaded();
    // 当广告加载失败后触发。
    void onFailedToLoad(Exception e);
    // 当广告渲染完毕后触发。
    void onRendered();
    // 当广告展示完毕后触发。
    void onShown();
    // 当广告中的有效连接被点击时触发。
    void onClicked();
    // 当用户点击打开其他应用（如外部浏览器、Deeplink、快应用等），从而当前应用在后台运行时触发。
    void onLeftApplication();
    // 当广告关闭时触发。
    void onClosed();
}
```

激励视频事件 RewardVideoListener

```
public interface RewardVideoListener extends Listener {
    // 当激励发生时触发。
    void onRewarded(String data);
}
```

视频事件 VideoListener

```
public interface VideoListener {
  void onVideoLoad(VideoController.Metadata metadata) {}

  void onVideoStart() {}

  void onVideoPlay() {}

  void onVideoPause() {}

  void onVideoEnd() {}

  void onVideoVolumeChange(double volume, boolean muted) {}

  void onVideoTimeUpdate(double currentTime, double duration) {}

  void onVideoError() {}
  // 当视频播放中断时触发。
```

```
  void onVideoBreak() {}
}
```

所有广告类型都可通过 `setListener` / `getListener` 来设置和获取。

竞价

::: tip 注意 注意，竞价失败后，广告对象废弃不可用，需及时 `xxxAd.destroy()` :::

```
XxxAd xxxAd = new XxxAd(activity, "广告位 TOKEN");
xxxAd.setListener(new Listener() {
  @Override
  public void onRendered() {
    // 竞胜 or 竞败（无竞价逻辑可不调用）
    xxxAd.sendWinNotice(price);//or xxxAd.sendLossNotice(lossReason);

    // 显示广告
    xxxAd.show();
  }
});
xxxAd.load();
```

```
/**
 * 获取广告返回价格，广告加载成功后才会有返回，否则返回 0
 */
public int getPrice()

/**
 * 发送竞价获胜上报
 * @param secondPrice，成交价
 * @return 是否成功发送
 */
public boolean sendWinNotice(int secondPrice = getPrice())

/**
 * 竞价失败上报
 * @param lossReason，失败原因
 * @return 是否成功发送
 */
public boolean sendLossNotice(int lossReason)
```

```
/** 竞价失败原因 */
public final class LossReason {
    /** 底价过滤 */
    public static final int BID_PRICE_FILTER = 1001;
    /** 竞价出价低于最高价 */
    public static final int PRICE_LOW_FILTER = 1002;
    /** 素材黑名单过滤 */
    public static final int ADM_BLACKLIST_FILTER = 1003;
    /** 竞品过滤 */
    public static final int COMPETE_FILTER = 1004;
    /** 超时过滤 */
    public static final int TIMEOUT_FILTER = 1005;
    /** 其它过滤 */
    public static final int OTHER_FILTER = 1006;
}
```

应用内浏览器设置

```
class LandingPageActivity {
    /**
```

```
     * 是否显示导航工具栏，默认是不显示的
     */
    public static void setDisplayActionBarEnabled(Boolean enabled);

    /**
     * 是否开启换页动画效果
     */
    public static void setAnimationEnabled(Boolean enabled);

    /**
     * 是否全屏显示，默认为全屏显示
     */
    public static void setFullScreenEnabled(Boolean enabled);

    /**
     * 显示方向，默认随设备旋转而旋转
     */
    public static void setOrientation(int orientation);
}
```

暂停、恢复、销毁

每种广告都有一个 `destroy` 方法，用于销毁对象，防止内存泄漏。如果可能的话，尽量在销毁对象调用一下该方法。

另外，对于横幅和信息流广告还有 `pause` 和 `resume` 两个方法，用于在 Activity 的 `onPause` 和 `onResume` 中调用，用于暂停和恢复广告的播放。

FAQ

androidx.core:core 版本冲突如何解决? 答：SDK 中目前使用的是 1.5.0 版本的 `androidx.core:core`，如果您项目中使用了更高版本的 `androidx.core:core` 库。可以将 gradle 配置中的：

```
dependencies {
    ...
    implementation 'com.xy.sdk:adtalos-sdk:6.0.2'
}
```

修改为：

```
dependencies {
    ...
    implementation('com.xy.sdk:adtalos-sdk:6.0.2') {
        exclude group: 'androidx.core', module: 'core'
    }
}
```

平台接入文档

- Gromore
- Taku
- ToBid