
小米 **Mediation SDK for Android** 接入使用文档

海量海外广告资源

2019 年 7 月

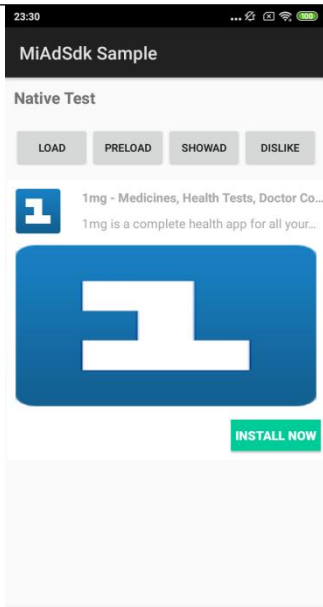
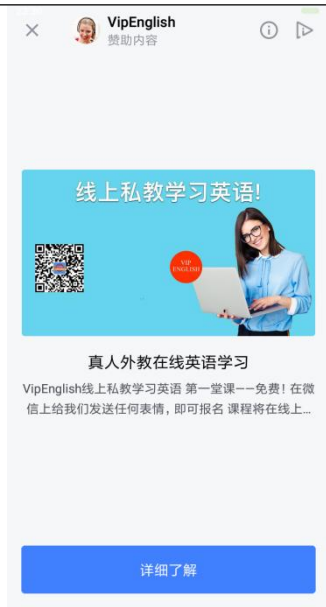

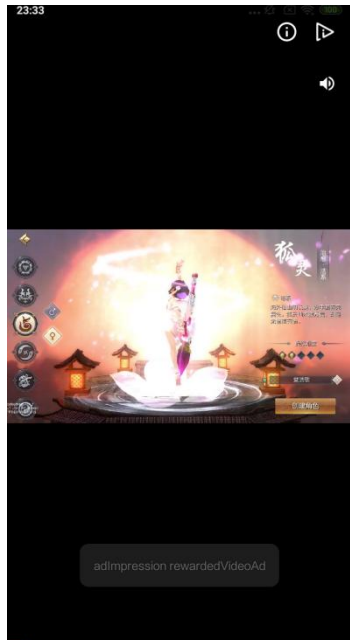
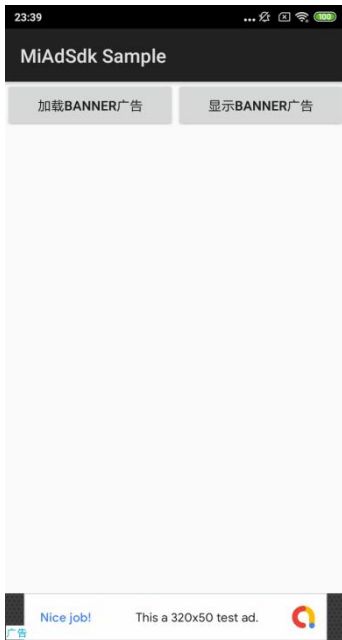
目录

1	Android Mediation SDK 功能简介.....	4
2	接入前的准备.....	5
2.1	申请账号.....	5
2.2	申请 APP ID.....	5
2.3	申请广告单元 ID.....	5
2.4	接入广告源 ID.....	5
3	初始化 SDK.....	6
3.1	导入 MediationSdk.jar 至项目.....	6
3.2	添加相应的权限.....	6
3.3	小米 Mediation SDK 初始化.....	6
3.4	代码混淆.....	7
3.5	本地默认云控配置（default-config）.....	7
4	原生广告接入.....	8
4.1	原生广告接入流程.....	8
4.1.1	参考初始化部分.....	8
4.1.2	注册 AdRenderer.....	8
4.1.3	请求广告.....	10
4.1.4	获取广告信息.....	10
4.1.5	广告解除绑定.....	10
4.1.6	增加信息流广告接口.....	10
4.1.7	针对 Facebook 和 Admob native 广告有一些特殊的设置.....	11
4.2	Native 视频广告测试.....	15
5	插屏广告接入.....	16
5.1	参考初始化部分.....	16
5.2	广告加载.....	16
5.3	在页面切换时展现广告.....	16
5.4	释放资源.....	16
6	贴片广告接入.....	18
6.1	参考初始化部分.....	18
6.2	广告加载.....	18
6.3	展现广告.....	18
6.4	释放资源.....	19

7 奖励视频广告接入.....	20
7.1 参考初始化部分.....	20
7.2 广告加载.....	20
7.3 展现广告.....	21
7.4 释放资源.....	21
8 横幅广告接入.....	22
8.1 参考初始化部分.....	22
8.2 广告加载.....	22
8.3 展现广告.....	23
8.4 释放资源.....	24
8.5 常见问题.....	24
9 X-out 接入.....	27
9.1 X-out 介绍.....	27
9.2 版本要求.....	27
9.3 API 接口及示例代码.....	27
10 API 接口.....	28
10.1 原生相关接口.....	28
10.2 公共接口.....	28
10.3 打点接口.....	29
11 错误码.....	30
12 媒体接入流程及注意事项.....	31
12.1 媒体接入流程.....	31
12.1.1 收到接入文件.....	31
12.1.2 收到相关参数.....	31
12.1.3 使用本地配置自行测试.....	32
12.1.4 使用线上测试环境进行测试.....	33
12.2 注意事项.....	33
12.2.1 AdReportHelper.reportPV 的正确调用.....	33
12.2.2 传入正确的参数.....	33
12.2.3 请求不到广告的常见原因.....	33
12.2.4 广告预加载与重试.....	33

1 Android Mediation SDK 功能简介

- (1) 提供高效、定制化的流量变现解决方案
- (2) 提供定制化的原生创意广告使 eCPM 显著提升，提供海量的广告资源确保广告填充率
- (3) 开发者可通过小米 Mediation SDK 接口请求广告数据并且在云端实时控制广告的优先级
- (4) 目前 Mediation SDK 支持的广告形式如下

<p>Native</p> <p>原生广告，广告中的素材资源可以通过自定义布局控制</p> 	<p>Interstitial</p> <p>插屏广告，全屏广告</p> 	<p>Intream</p> <p>贴片广告，可在视频频前或后播放的视频广告</p> 
<p>RewardVideo</p> <p>奖励视频，全屏广告</p> 	<p>Banner</p> <p>横幅广告，矩形图片或文字，只占用 app layout 的一部分</p> 	

2 接入前的准备

2.1 申请账号

开发者从小米 Mediation 运营人员处获取到账号、密码后，登录小米 Mediation 系统后台。

2.2 申请 APP ID

开发者使用产品提供的 MediaType 作为 APPID，作为应用的唯一标识。

2.3 申请广告单元 ID

TagID：开发者每创建一个广告位后，系统会自动生成广告单元 ID，由产品提供。

2.4 接入广告源 ID

PlacementID：开发者创建完一个广告位后，添加需要 Mediation 的广告网络（Facebook、AdMob 等），并且通过给定 eCPM 来控制该广告网络的优先级。

以上由产品申请后提供给开发者。接入前请先阅读 [12 媒体接入流程及注意事项](#)。

3 初始化 SDK

3.1 导入 MediationSdk.jar 至项目

将 [MediationSdk.jar](#) 复制到项目的 libs 文件夹中，然后按照下面修改 build.gradle:

```
dependencies {  
    ...  
    compile files('libs/MediationSDK.jar')  
    // 如果接入 facebook 必须是 5.1.0 以上版本  
    compile 'com.facebook.android:audience-network-sdk:5.3.1'  
    // 如果接入 admob 需要 15.0.1 以上版本  
    compile 'com.google.android.gms:play-services-ads:16.0.0'  
}
```

注：首次接入可能需要根据项目情况引入额外的包：analytics_api_1.9.3.jar（广告曝光点击等打点统计）、analytics_core.apk（打点统计，单发非 MIUI 系统应用需要，放在 assets 文件夹下，同时需要 analytics_api_1.9.3.jar）、[zeus-columbus-vx x x.jar](#)（mi 广告）、[AdJumper v1.0.1.aar](#)（mi 广告跳转）。请根据自身项目情况进行导入。

3.2 添加相应的权限

打开 AndroidManifest.xml，配置以下内容：

必要权限

```
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />  
<uses-permission android:name="com.miui.systemAdSolution.adSwitch.PROVIDER"/>
```

可选权限

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />  
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />  
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
```

3.3 小米 Mediation SDK 初始化

请在每个要用到广告的 Application 的 onCreate 中初始化 SDK。首先初始化第三方 SDK，例如：

```
// ColumbusSDK 需要  
AdGlobalSdk.initialize(getApplicationContext(), "AppKey", "AppSecret");  
// FacebookSDK 需要  
AudienceNetworkAds.initialize(this);  
// 开启开发者模式，Debug mode, for more log  
AdGlobalSdk.setDebugOn(true);  
// 开启测试环境，Staging mode, get mock ads, only for test!  
AdGlobalSdk.setStaging(true);  
// 调用参数中的 APPID 是小米 Mediation SSP 平台分配的应用 id  
MiAdManager.applicationInit(Context context, String appid, String configKey);
```

注：如果第三方 SDK 需要初始化，请在小米 Mediation 初始化之前初始化此 SDK。

3.4 代码混淆

如果需要代码混淆，请添加以下混淆脚本

```
-dontwarn com.xiaomi.**  
-keep class com.xiaomi.** { *;}  
-keep all adapter classes  
-keep class class com.miui.zeus.columbus.** { *;}
```

3.5 本地默认云控配置（default-config）

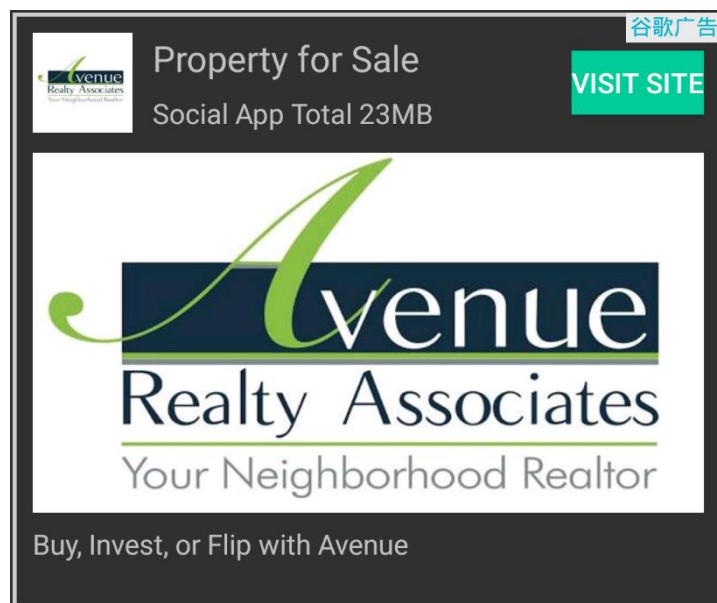
各媒体的广告 id 以及优先级拉取展示策略，均以服务端下发的配置云控。本地保存 default-config 需和线上配置一致，以保证最初在未拉取到的情况下有配置文件可供使用。

注：各媒体的默认配置需要联系产品负责人来确定，然后研发放在媒体 APP 的指定本地路径即可

```
String defaultConfigStr = FileUtils.readStringFromAsset(context, "default-  
config.txt");  
MiAdManager.setDefaultConfig(defaultConfigStr , false);
```

4 原生广告接入

集成后效果如下：



4.1 原生广告接入流程

广告位曝光方法说明：**AdReportHelper.reportPV(String positionId)**

请在广告位展现时/有机会展示广告时调用进行广告位曝光上报，此方法与广告是否曝光无关；方法调用次数与广告位曝光次数保持一致，**请勿多次调用**。

第 1 步：SDK 初始化

第 2 步：请求广告

第 3 步：获取广告信息

第 4 步：广告绑定

第 5 步：解除广告绑定

4.1.1 参考初始化部分

确保 SDK 已经在 Application 中初始化成功，见 [3 初始化 SDK](#)。

4.1.2 注册 AdRenderer

请务必在 load 前注册要接入的 DSP 的 renderer！

```
mNativeAdManager = new NativeAdManager(this, "小米 Mediation 广告位 id");

final FacebookAdRenderer facebookAdRenderer = new FacebookAdRenderer(new
FacebookAdRenderer.FacebookViewBinder.Builder(R.layout.facebook_native_ad_layout)

    .nativeAdLayoutId(R.id.native_ad_layout)

    .titleId(R.id.fb_native_title, false) // 是否可点击

    .textId(R.id.fb_native_text, false)

    .mediaId(R.id.fb_native_main_media)
```



```

        .iconId(R.id.fb_native_ad_icon)

        .callToActionId(R.id.fb_native_cta, false)

        .adChoicesContainerId(R.id.fb_ad_choices_container)

        .dislikeId(R.id.fb_native_dislike)

        .build()
    );

    final AdmobAdRenderer admobAdRenderer = new AdmobAdRenderer(
        new AdmobAdRenderer.AdmobViewBinder.Builder(R.layout.admob_native_ad_layout)
            .titleId(R.id.unified_title)
            .textId(R.id.unified_body)
            .mediaId(R.id.unified_media_view)
            .iconId(R.id.unified_icon)
            .callToActionId(R.id.unified_action_btn)
            .dislikeId(R.id.unified_dislike)
            .build());

    final ColumbusAdRenderer columbusAdRenderer = new ColumbusAdRenderer(
        new
ColumbusAdRenderer.ColumbusViewBinder.Builder(R.layout.columbus_native_ad_layout)
            .titleId(R.id.columbus_native_title)
            .textId(R.id.columbus_native_text)
            .imageId(R.id.columbus_native_main_image)
            .iconId(R.id.columbus_native_icon_image)
            .callToActionId(R.id.columbus_native_cta)
            .adChoicesContainerId(R.id.columbus_ad_choices_container)
            .dislikeId(R.id.columbus_native_dislike, false) // 不使用默认弹窗
            .build()
    );

    mNativeAdManager.registerAdRenderer(admobAdRenderer);

    mNativeAdManager.registerAdRenderer(facebookAdRenderer);

    mNativeAdManager.registerAdRenderer(columbusAdRenderer);

```

4.1.3 请求广告

请求广告之前请务必先注册 `renderer`!

```
NativeAdManager mNativeAdManager ;
private void requestNativeAd() {
    mNativeAdManager.setNativeAdManagerListener(new
        mNativeAdManager.NativeAdManagerListener() {
            @Override
            public void adLoaded() {
                // Code to be executed when an ad request success.
            }
            @Override
            public void adFailedToLoad(int code) {
                // Code to be executed when an ad request fails.
            }
            @Override
            public void adImpression(INativeAd nativeAd) {
                // Code to be executed when an ad impression
            }
            @Override
            public void adClicked(INativeAd nativeAd) {
                // Code to be executed when and ad clicked
            }
        });

    // Request an ad
    mNativeAdManager.loadAd();
}
```

4.1.4 获取广告信息

在 `adLoaded()` 中获取 `load` 结果

```
mNativeAd = mNativeAdManager.getAd();
View adView = mNativeAd.getAdView();
```

此 `adView` 可以添加到您需要展示广告的位置

4.1.5 广告解除绑定

在不需要该广告的时候, 将广告与之前绑定的 `View` 进行解绑。

跟据自己页面的生命周期, 在合适的时机反注册广告的 `view`:

```
ad.unregisterView();
```

注: 不调用可能会导致内存泄漏

4.1.6 增加信息流广告接口

如果希望将广告展示在信息流或者排行榜里面, 目前只有 **Columbus** 提供了一次 `load` 获取多个广告的接口, 步骤:

1. 使用 **V1.1.10.0** 及以上 **mediation** 版本
2. 使用支持信息流接口的 **ColumbusSdk**(全部版本都支持)

3.使用最新的 ColumbusNativeAdapter.java

4.在调用 NativeAdManager.loadAd() 方法之前,先调用

```
NativeAdManager.setLoadConfig(new LoadConfigBean.Builder()
    .setNativeAdSize(REQUEST_AD_SIZE)
    .setIsPalaceAd(true)
    .build());
```

方法	说明
setNativeAdSize(int num)	设置一次 load 后获取的广告数量
setIsPalaceAd(boolean flag)	设置获取广告的大图尺寸,宫格类广告需要设置为 true; 普通 native 广告可设置为 false 或不设置,大图尺寸为 1200*627(每家 dsp 图片比例可能有细微差别)

5.调用 loadAd 等待 adLoaded 方法回调后,使用 NativeAdManager.getAdList()获取广告

6.如果一个广告调用过 INative.registerViewForInteraction()展示广告,最后请使用 INative.unregisterView()方法销毁广告.

4.1.7 针对 Facebook 和 Admob native 广告有一些特殊的设置

为了支持 facebook 与 admob native 视频广告请使用各自平台的 MediaView 控件。

需要对以上自定义布局文件做些特殊修改。

(1) Facebook

举例说明 FacebookAdRenderer 中的 facebook_native_ad_layout 写的规范

facebook 需要将原来显示大图的 ImageView 替换成 facebook 的 MediaView(必须是 facebook 包里面的类)，需要将显示 icon 的 ImageView 替换成 facebook 的 AdIconView。

a. 在 xlm 中的修改（只修改 fb ad view 的 xml）（以下代码请根据实际业务需求进行相应调整）：

将原来的：

```
<ImageView
    android:id="@+id/native_main_image"
    android:layout_width="match_parent"
    android:layout_height="180dp"
```

```
————— android:layout_margin="5dp"  
————— android:scaleType="fitXY"/>
```

替换为：

```
<com.facebook.ads.MediaView  
    android:id="@+id/native_main_media"  
    android:layout_width="match_parent"  
    android:layout_height="180dp"  
    android:layout_margin="5dp"  
    android:scaleType="fitXY" />
```

将

```
<ImageView  
  
    android:id="@+id/native_icon_image"  
  
    android:layout_width="50dp"  
  
    android:layout_height="50dp"  
  
    android:background="@null"  
  
    android:layout_alignParentLeft="true"  
  
    android:layout_alignParentTop="true"  
  
    android:layout_marginTop="10dp"  
  
    android:layout_marginLeft="10dp"  
  
    android:scaleType="fitXY"  
  
/>
```

替换为

```
<com.facebook.ads.AdIconView  
    android:id="@+id/native_ad_icon"  
    android:layout_width="50dp"  
    android:layout_height="50dp"  
    android:layout_alignParentLeft="true"  
    android:layout_alignParentTop="true"  
    android:layout_marginLeft="10dp"  
    android:layout_marginTop="10dp" />
```

b. 对应的代码修改(以下代码请根据实际业务需求进行相应调整)：

将：

```
ImageView imageViewMain = (ImageView) adView.findViewById(R.id.native_main_image);
```

```
String mainImageUrl = ad.getAdCoverImageUrl();  
VolleyUtil.loadImage(imageViewMain, mainImageUrl);
```

替换为:

```
MediaView mediaViewMain = (MediaView) adView.findViewById(R.id.media_view)
```

将

```
String iconUrl = ad.getAdIconUrl();  
————— ImageView iconImageView = (ImageView) adView.findViewById  
(R.id.native_icon_image);  
————— if (iconUrl != null) {  
————— VolleyUtil.loadImage(iconImageView, iconUrl);
```

替换为

```
AdIconView adIconView = adView.findViewById(R.id.native_ad_icon);
```

Facebook 的注册必须使用 `registerViewForInteraction(adView, clickableViews);` 方法且 `clickableViews` 里面必须有 `mediaViewMain` 和 `adIconView`, 否则可能这两个控件不显示视图。

加入 facebook 的广告标识

```
AdOptionsView adOptionsView = new AdOptionsView(mContext, (NativeAdBase)  
adObject, nativeAdLayout);
```

最后增加了 `com.facebook.ads.NativeAdLayout` 类(它是 `FrameLayout` 的一个封装)用来给 `NativeAd` 添加一个容器。

```
nativeAdLayout = new NativeAdLayout(mContext);
```

```
nativeAdLayout.addView(adView);
```

(`adView` 是原来媒体方自己写的广告布局.只是将原来的布局使用 `NativeAdLayout` 包裹一次)

Facebook MediaView 内存泄露相关问题: 为了释放 `MediaView` 相关资源, 需要在使用完 `MediaView` 后调用 `MediaView` 的 `destory()` 方法, 或者在 `FacebookNativeAdapter` 的 `unregisterView()` 中同时调用 `Facebook.NativeAd` 的 `unregisterView()` 和 `destory()`。

需要注意的地方踩过的坑: 在调研 fb 的 `MediaView` 过程中发现在 xml 中直接将 `MediaView` 设置为 `gone` 然后在代码中根据需要调用 `mediaview.setVisibility(View.VISIABLE)` 显示的过程中, 如果要显示的是视频广告, 视频广告很可能显示不出来或者在 android5.1 手机上崩溃。如果有相关团队在接入过程中遇到类似问题请在 `mediaview` 的外层包裹一层 `GroupView`, 直接控制外层的 `visibility`。

如果此广告是原生横幅广告（没有 **MediaView** 的 native 广告），请调用以下方法

```
mNativeAdManager.setLoadConfig(new LoadConfigBean.Builder()
    .setNativeBannerAd(true)
    .build());
```

在 **AndroidManifest.xml** 文件的 **application** 节点增加 **android:hardwareAccelerated="true"** 为整个应用启用硬件加速功能(选加).

添加网络安全配置文件

1)在应用资源的 **xml** 文件夹下创建新的 **XML** 文件：**res/xml/network_security_config.xml**。

2)然后在应用程序清单 **AndroidManifest.xml** 中，按以下方法在您的应用程序属性中添加配置：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:networkSecurityConfig="@xml/network_security_config"
        ... >
        ...
    </application>
</manifest>
```

3)在允许的明文流量网域中添加 **localhost**,在 **res/xml/network_security_config.xml** 文件中，您可以通过添加以下代码在允许的明文流量网域中添加 **localhost**：

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    ...
    <domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">127.0.0.1</domain>
    </domain-config>
    ...
</network-security-config>
```

<https://developers.facebook.com/docs/audience-network/android-network-security-config>

(2) Admob

根据 **Admob** 官方文档，目前支持视频的 native 为 **install** 类型的广告。**注意：** admob 视频广告在播放过程中如果频繁的锁屏与解锁会出现视频只播放声音不播放画面的问题，属于 **admob sdk** 问题

a. 在 **xml** 中的修改，添加如下代码：

```
<com.google.android.gms.ads.formats.UnifiedNativeAdView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/admob_native_install_adview"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <!-- some view -->
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="180dp"
        android:visibility="gone"/>
    <com.google.android.gms.ads.formats.MediaView
```

```
        android:layout_width="match_parent"
        android:layout_height="180dp"
        android:visibility="gone"/>
    <!-- some view -->
</com.google.android.gms.ads.formats.UnifiedNativeAdView >
```

4.2 Native 视频广告测试

建议修改本地 `default-config` 文件来测试视频广告，比较方便。为了使用本地 `config` 请求广告，请在初始化 `sdk` 时调用。

```
String defaultConfigStr = xxx.readStringFromAsset(context, "default-
config.txt");
MiAdManager.setDefaultConfig(defaultConfigStr, true);
```

Facebook 视频广告测试：

在 `parameter` 前面添加（`VID_HD_9_16_39S_APP_INSTALL#`）这样请求时就只会拉取视频广告方便测试。

```
{
    "name": "fb",
    "parameter": "YOUR_PLACEMENT_ID",
    "parameter": "VID_HD_9_16_39S_APP_INSTALL#YOUR_PLACEMENT_ID",
    "weight": 8
}
```

5 插屏广告接入

广告位曝光方法说明：`AdReportHelper.reportPV(String positionId)`

请在广告位展现时/有机会展示广告时调用进行广告位曝光上报，此方法与广告是否曝光无关；方法调用次数与广告位曝光次数保持一致，请勿多次调用。

5.1 参考初始化部分

确保 SDK 已经在 Application 中初始化成功，见 [3 初始化 SDK](#)

5.2 广告加载

```
private InterstitialAdManager mInterstitialAdManager;
private void requestInterstitialAd() {
    mInterstitialAdManager = new InterstitialAdManager(getContext(), "小米
Mediation 广告位 id");
    mInterstitialAdManager.setInterstitialAdCallback(new
        InterstitialAdCallback() {
            @Override
            public void onAdLoaded() {
                // Ad is loaded and ready to be displayed!.
            }
            @Override
            public void onAdLoadedFailed(int errorCode) {
                // Code to be executed when an ad request fails.
            }
            @Override
            public void onAdDisplayed() {
                // Code to be executed when an ad displayed
            }
            @Override
            public void onAdDismissed() {
                // Code to be executed when an ad dismissed
                mInterstitialAdManager.destroyAd();
            }
            @Override
            public void onAdClicked() {
                // Code to be executed when and ad clicked
            }
        });

    // Request an ad
    mInterstitialAdManager.loadAd();
}
```

5.3 在页面切换时展现广告

收到 `onAdLoaded` 回调后展示广告：

```
if (mInterstitialAdManager.isReady()) {
    mInterstitialAdManager.showAd();
}
```

5.4 释放资源

最后在 Activity 的 `onDestory()` 方法中释放资源：

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

6 贴片广告接入

广告位曝光方法说明: `AdReportHelper.reportPV(String positionId)`

请在广告位展现时/有机会展示广告时调用进行广告位曝光上报, 此方法与广告是否曝光无关; 方法调用次数与广告位曝光次数保持一致, 请勿多次调用。

6.1 参考初始化部分

确保 SDK 已经在 Application 中初始化成功, 见 [3 初始化 SDK](#);

6.2 广告加载

```
InstreamVideoAdManager mManager = new InstreamVideoAdManager(mContext,
PLACE_ID);
mManager.setInstreamAdCallback(new InstreamVideoAdCallback() {
    @Override
    public void onAdVideoComplete(INativeAd ad) {
        // Instream Video Ad Complete - the ad has been played to the end
        // You can use this event to continue your video playing
    }
    @Override
    public void adLoaded() {
        // Ad is loaded and ready to be displayed!.
    }
    @Override
    public void adFailedToLoad(int code) {
        // Code to be executed when an ad request fails.
    }
    @Override
    public void adImpression(INativeAd nativeAd) {
        // Code to be executed when an ad impression.
    }
    @Override
    public void adClicked(INativeAd nativeAd) {
        // Code to be executed when and ad clicked
    }
});

// 这种 load 方式将以 container 的大小显示贴片广告
mManager.loadAd(mAdViewContainer);
// 这种 load 方式将以自定义大小显示贴片广告
mManager.loadAd(mAdViewContainer, width, height);
```

6.3 展现广告

在 `adLoaded()` 中展示广告

```
mManager.setInstreamAdCallback(new InstreamVideoAdCallback() {
    @Override
    public void adLoaded() {
        if (mManager.isReady()) {
            mManager.showAd();
        }
    }
    .....
});
```

或者特定时间展示广告

```
if (mManager.isReady()) {  
    mManager.showAd();  
}
```

请添加如下代码，**否则**点击 Learn More 返回后**谷歌贴片**广告会出现**不续播**的情况

```
@Override  
public void onResume() {  
    super.onResume();  
    mManager.onResume();  
}
```

6.4 释放资源

在 onAdVideoComplete() 释放资源

```
mManager.setInstreamAdCallback(new InstreamVideoAdCallback() {  
    .....  
    @Override  
    public void onAdVideoComplete(INativeAd ad) {  
        if (mManager != null) {  
            mManager.destroyAd();  
        }  
    }  
    .....  
});
```

或者 activity onDestroy 中释放资源:

```
@Override  
public void onDestroy() {  
    if (mManager != null) {  
        mManager.destroyAd();  
    }  
    super.onDestroy();  
}
```

7 奖励视频广告接入

广告位曝光方法说明: `AdReportHelper.reportPV(String positionId)`

请在广告位展现时/有机会展示广告时调用进行广告位曝光上报, 此方法与广告是否曝光无关; 方法调用次数与广告位曝光次数保持一致, 请勿多次调用。

7.1 参考初始化部分

确保 SDK 已经在 Application 中初始化成功, 见 [3 初始化 SDK](#);

7.2 广告加载

```
RewardedVideoAdManager mManager = new RewardedVideoAdManager(mContext,
PLACE_ID);
mManager.setRewardedVideoAdCallback(new RewardedVideoAdCallback() {
    @Override
    public void adLoaded() {
        // Ad is loaded and ready to be displayed!.
    }
    @Override
    public void adFailedToLoad(int errorCode) {
        // Ad failed to load.
    }
    @Override
    public void adImpression(INativeAd nativeAd) {
        // Video ad impression - the event will fire when the
        // video starts playing
    }
    @Override
    public void adClicked(INativeAd nativeAd) {
        // Rewarded Video is clicked
    }
    @Override
    public void onAdStarted() {
        // video ad starts playing
    }
    @Override
    public void onAdCompleted() {
        // Rewarded Video View Complete - video played to the end.
    }
    @Override
    public void onAdRewarded() {
        // Reward the user
    }
    @Override
    public void onAdDismissed() {
        // Reward Video is dismissed
    }
});

// Request an ad
if (mRewardedVideoAdManager != null) {
    mRewardedVideoAdManager.loadAd();
}
```

7.3 展现广告

在 `adLoaded()` 中展示广告，或者在特定的时间展示广告：

```
if (mManager.isReady()) {  
    mManager.showAd();  
}
```

7.4 释放资源

在 `onAdDismissed()`

```
mManager.setInstreamAdCallback(new InstreamVideoAdCallback() {  
    .....  
    @Override  
    public void onAdVideoComplete(INativeAd ad) {  
        if (mManager != null) {  
            mManager.destroyAd();  
        }  
    }  
    .....  
});
```

或者 activity `onDestory()` 中释放资源：

```
@Override  
public void onDestory() {  
    if (mManager != null) {  
        mManager.destroyAd();  
    }  
    super.onDestory();  
}
```

8 横幅广告接入

广告位曝光方法说明: `AdReportHelper.reportPV(String positionId)`

请在广告位展现时/有机会展示广告时调用进行广告位曝光上报, 此方法与广告是否曝光无关; 方法调用次数与广告位曝光次数保持一致, 请勿多次调用。

Tips: 所有媒体构造 `BannerAdSize` 时使用 `dp` 值

比如想申请 `1008px * 528 px` 的 banner, 请找产品同学换算成 `336dp * 176 dp`

最终以 `336*176` 请求广告

8.1 参考初始化部分

确保 SDK 已经在 Application 中初始化成功, 见 [3 初始化 SDK](#);

8.2 广告加载

```
BannerAdManager mBannerAdManager = new BannerAdManager(mContext,
POSITION_ID);
mBannerAdManager.setBannerAdCallback(new BannerAdCallback() {
    @Override
    public void adLoaded() {
        // Ad is loaded and ready to be displayed!
    }
    @Override
    public void adLoaded(int width, int height) {
        //浏览器专用
    }
    @Override
    public void adFailedToLoad(int errorCode) {
        // Ad failed to load.
    }
    @Override
    public void adImpression(INativeAd nativeAd) {
        // Ad is showed.
    }
    @Override
    public void adClicked(INativeAd nativeAd) {
        // Ad is clicked
    }
    @Override
    public void adDisliked(INativeAd nativeAd, int dislikeCode) {
        // Ad is disliked
    }
});

mManager.setAdSize(BannerSize bannerSize); or
// mManager.setAdSizeList(List<BannerSize> list)
// setIsWebViewBannerSupported(boolean isSupported) 仅 mib 支持, 如果设置为
false, 即使有 AdmobBannerAdapter 和 FacebookBannerAdapter 也不会请求广告
mManager.loadAd();
```

8.3 展现广告

A、如果**不需要**获取尺寸，在回调 `adLoaded()` 直接展示广告，或者在特定的时间展示广告：

```
@Override
public void adLoaded() {
    // Ad is loaded and ready to be displayed!
    if (mManager.isReady()) {
        mManager.showAd(mContainer);
    }
}
```

B、如果**需要**获取尺寸，然后再展示

```
@Override
public void adLoaded() {
    // Ad is loaded and ready to be displayed!
    isReady = true;
    if (isSizeGeted) {
        mManager.showAd(mContainer);
    }
}

@Override
public void adLoaded(int width, int height) {
    //浏览器专用
    isSizeGeted = true
    mWidth = width;
    mHeight = height;
    If (isReady) {
        mManager.showAd(mContainer);
    }
}
```

1、请求广告，`mContainer` 的尺寸必须大于等于横幅广告的大小，广告将会不展示并且会得到警告信息

标准 `admob` 横幅广告尺寸

Size in dp (WxH)	Availability	AdSize constant
320x50	Phones and Tablets	BANNER
320x100	Phones and Tablets	LARGE_BANNER
300x250	Phones and Tablets	MEDIUM_RECTANGLE
Screen width x 32 50 90	Phones and Tablets	SMART_BANNER

标准 `facebook` 横幅广告尺寸（宽度灵活最小 320px，**只需要定义高度**）

Size in dp (WxH)	Availability	AdSize constant
------------------	--------------	-----------------

320x50	Phones	BANNER_HEIGHT_50
320x90	tablets and larger devices	BANNER_HEIGHT_90
300x250	scrollable feeds or end-of-level screens	RECTANGLE_HEIGHT_250

标准 columbus 横幅广告尺寸 (支持各种自定义尺寸, columbus>= 1.2.2)

Size in dp (WxH)	Availability	AdSize constant
320x50	Phones	BANNER_HEIGHT_50
300x250	tablets and larger devices	BANNER_HEIGHT_90
WidthxHight	自定义宽高	

8.4 释放资源

在 activity onDestroy()中释放资源:

```
@Override
public void onDestroy() {
    if (mManager != null) {
        mManager.destroyAd();
    }
    super.onDestroy();
}
```

8.5 常见问题

1、如何请求到广告？

(1) 在研发初步接入阶段可以在以下链接添加 Mock 广告和 gaid 的方式 (可以找 SDK 客户端或者 SDK 服务端)

<http://tjl-miui-ad-stage06.kscn:9088/#/MockAdConfig?nsukey=N0ascDK0j8Q7qTf%2FeD02buQzqWaj0jid7jxeTIJ0nuw1jDwoyYmMBumS840Mz7SKzKLzSOSJFXdfJ7GiwJAvm6kkDP4apawe30PVomv7fX5ycW%2FTsRRCt%2BYTz0571tx8VqZ24m%2FaT6Ie03KVaxkqIOAT1HJQdlfc80Nu1WtDh8U%3D>

(2) 直客广告都是从 miads 投放, 在**上线前必须联调**方可上线

由于在 (1) 平台中 Mock 广告的优先级高于 miads, 所以在从 miads 拉取广告前, 从 (1) 的广告位中将测试手机 gaid 删除, 并将手机 gaid 添加到 miads 的广告白名单中, 且测试手机需要修改到和 miads 中广告一致的国家

2、Columbus 非 WebView Banner 如何自行拉伸？

如果需要完美适配界面，仅限只接 Columbus 非 WebView Banner，以下代码可供参考：

例：F11 手机 宽 1080px, 屏幕 density 为 2.7, 请求 320dp * 50dp 的横幅广告

广告最终展示大小为 864px * 135px, 如果想要更改 banner 的宽高, 请自行计算好 320: 50 的比例, 参考代码如下:

```
mBannerAdManager.showAd(mAdViewContainer); // 会将 BannerAdView add 到你的容器
View bannerView = mAdViewContainer.getChildAt(0); // 获取 BannerAdView
if (bannerView instanceof BannerAdView) {
    View view = ((BannerAdView) bannerView).getChildAt(0); // 获取内容
    ViewGroup.LayoutParams layoutParams = view.getLayoutParams();
    layoutParams.width = 1080; // 拉伸宽度
    layoutParams.height = 168; // 拉伸高度
    view.setLayoutParams(layoutParams); // 请保持原有宽高比例, 以确保展示效果
}
```

3、是否可以用一个全局 Holder 保存 BannerAdManager, 在不同 Fragment 复用 Manager?

答: 不能。建议在 Fragment 或者 Activity 的 onDestroy() 中调用

mBannerAdManager.destroyAd() 后, 就不要再有其他地方持有 Manager 的引用, 否则会导致内存泄露

4、BannerAdManager 在调用 destroyAd() 后是否能够复用请求广告和展示广告?

答: 可以, 除去 2 中情况, 可正常请求和展示广告

5、在接入 Columbus Banner 如果遇到图片不展示或者不跳转, 为了适配 android P 需要添加网络配置, 如果仍然存在问题请联系 sdk 研发小组。

媒体端在 manifest application 下添加配置

```
android:networkSecurityConfig="@xml/network_security_config"
```

network_security_config 文件内容

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
<base-config cleartextTrafficPermitted="true">
<trust-anchors>
<certificates src="system"/>
</trust-anchors>
</base-config>
<debug-overrides>
<trust-anchors>
<certificates src="user"/>
</trust-anchors>
</debug-overrides>
</network-security-config>
```

6、adLoaded() 与 adLoaded(int width, int height)的区别?

答: 只有回调 adLoaded() 才是广告真正请求成功。

7、如何接入 Columbus Banner 广告(包含 webview 和非 webview)?

答: Columbsv1.2.3 或更高版本 Mediation1.2.2.3 或更高版本

<https://wiki.n.miui.com/pages/viewpage.action?pageId=73317447>

<https://wiki.n.miui.com/pages/viewpage.action?pageId=73317103>

BannerAdManager.setIsWebViewBannerSupported(false)//仅 ColumbusBanner 会请求非 webview banner 广告

(1)在测试环境请求广告 *##672##*打开测试环境, 如果返回 10002, 则接入 sdk 成功

(2)Mock 测试广告, 找服务端同学, 并告知其广告位、gaid、banner 形式(非 webview banner 返回真落地页)、banner 尺寸(单位为 dp)

9 X-out 接入

9.1 X-out 介绍

X-out 作为海外用户体验计划的一部分，可以作为收集用户反馈的一个渠道。在此之前，在用户点击关闭海外广告后直接关闭广告，并没有收集原因及广告素材相关信息。X-out 可以在关闭广告时让用户选择原因，通过广告 SDK 上报原因及广告信息，根据用户的反馈及时对违规广告做屏蔽处理和对广告请求进行频次控制。

9.2 版本要求

Mediation 版本 \geq V1.1.10.3

9.3 API 接口及示例代码

(1) API 接口

序号	方法名	方法说明
1	<code>dislikeAd(INativeAd nativeAd)</code>	不喜欢此广告，参数：不喜欢的广告
2	<code>dislikeAndReport(Context context)</code>	使用默认弹窗时调用，用户选择原因后 SDK 内部会进行打点上报
3	<code>dislikeAndReport(Context context, int dislikeCode)</code>	使用自定义弹窗时调用，这里务必传入正确的反馈码 (int)

(2) 示例代码

A. 用户点击了关闭广告按钮后调用

```
private void dislikeAd(INativeAd nativeAd) {
    if (nativeAd == null) {
        Toast.makeText(mContext, "no native ad showing!",
            Toast.LENGTH_SHORT).show();
        return;
    }

    if (nativeAd != null) {
        // 使用默认弹窗时调用
        nativeAd.dislikeAndReport(mContext);

        /**
         * 如需自定义弹窗，请进行以下操作
         * (1) 自定义弹窗，反馈原因及反馈码请严格按照
            com.xiaomi.miglobaladsdk.FeedbackConst，禁止更改，如需增加请务必协商好再加入
            Your code;
         * (2) 用户选择原因后打点，广告反馈打点调用 ***必须传入正确的广告反馈码***
         *     nativeAd.dislikeAndReport(context, dislikeCode);
         */
    }
}
```

B. 用户选择原因后媒体方会收到回调

```
@Override
public void adDisliked(INativeAd nativeAd, int dislikeCode) {
    // 可以选择在这里关闭广告并 unregisterView
    nativeAd.unregisterView();
    nativeAdContainer.removeAllViews();
    mNativeAd = null;
    MLog.d(TAG, "AdDisliked nativeAd, dislikeCode: " + dislikeCode);
}
```

10 API 接口

10.1 原生相关接口

1. com.xiaomi.miglobaladsdk.nativead.api.NativeAdManager

方法名	方法说明
NativeAdManager(Context context, String posid)	构造方法
getAd()	获取广告内容
setNativeAdManagerListener(NativeAdManagerListener listener)	设置加载广告时的回调
loadAd()	并发请求广告
preloadAd()	串行请求广告

2. com.xiaomi.miglobaladsdk.nativead.api.NativeAdManager.NativeAdManagerListener

方法名	方法说明
onAdLoaded()	广告加载成功
adFailedToLoad(int error)	广告加载失败，error 为具体的错误码
adClicked(INativeAd ad)	广告点击的回调
adImpression(INativeAd nativeAd);	广告展示的回调

10.2 公共接口

com.xiaomi.miglobaladsdk.nativead.api.INativeAd

方法名	方法说明
getAdtitle()	广告 title
getAdBody()	广告描述
getAdIconUrl()	icon 的 url

getAdCoverImageUrl()	大图的 url
getAdCallToAction()	button 的文案
getAdStarRating()	广告的评分信息（可能为空）
getAdSocialContext()	广告的下载数或者是网站（可能为空）
hasExpired()	是否过期的判断 (true: 过期 false: 不过期)
isDownloadApp()	是否是下载类型广告 (true: 是 false: 不是)
setImpressionListener(ImpressionListener listener)	设置广告展现的回调接口
setAdOnClickListener(IAOnClickListener listener)	设置广告点击的回调接口
registerViewForInteraction(View view)	绑定广告内容和广告展现的 view
unregisterView()	解绑广告内容和广告展现的 view

10.3 打点接口

com.xiaomi.miglobaladsdk.report.AdReportHelper

方法名	方法说明
reportPV(String positionId)	广告位曝光打点, 传入广告位 id, 请媒体方正确调用
reportCustomPV(String positionId, String key, String value)	自定义广告位曝光打点, 目前浏览器使用分频道打点, positionId 广告位 id, key 需传入 channel, value 传入渠道名
reportDislike(String positionId, String key, String value)	不喜欢此广告打点, positionId 是广告位 id, key 和 value 可传 null。

11 错误码

错误码	错误码说明	解决方案
10001	Mediation 配置没有加载成功	检查 Mediation 的配置是否在 SSP 配置成功
10002	没有广告数据	检查各广告平台的 id 是否生效， 或抓包查看有无广告数据返回
10003	内部错误	向小米 Mediation 团队反馈
10004	请求超时	稍后再重新请求
10005	没有合法的 adaptor	检查自己设置的 adapter 是否合法
10009	参数错误	检查自己的参数

12 媒体接入流程及注意事项

12.1 媒体接入流程

12.1.1 收到接入文件

由 Mediation 产品或研发提供接入需要的文件：

- (1) 最新的 [MediationSdk-x.x.x.jar](#)、Demo、接入文档和提测邮件模板；
- (2) 其他可能需要的 jar/aar: [analytics_api_1.9.3.jar](#) (广告曝光点击等打点统计)、[zeus-columbus-vx_x_x.jar](#)(mi 广告)、[AdJumper_v1.0.1.aar](#) (mi 广告跳转)。

12.1.2 收到相关参数

(1) 请 Mediation 产品提供相关参数，包括广告 ID (TagID 和 PlacementID)、APP_ID、APP_KEY 和 APP_SECRET。以下为参数说明：

- **TagID:** 小米的广告位 ID，对应本地配置 (default-config.txt) 中的 placeid，形式为 1.***.***.***;
您需要：(1) 写入本地配置 (default-config.txt)；(2) 初始化**AdManager 时传入，例如：

```
NativeAdManager nativeAdManager = new NativeAdManager(Context context, String TagID);
```
- **PlacementID:** DSP 的广告位 ID，对应本地配置 (default-config.txt) 中的 parameter，形式是一长串字符串，mi 的 PlacementID 和 TagID 相同；
您需要：写入本地配置 (default-config.txt)。
- **APP_ID、APP_KEY、APP_SECRET:** APP_ID 和 APP_KEY 相同，媒体 APP 的唯一表示。
您需要：初始化 Mediation 时传入 APP_ID，configKey 传入 " miglobaladsdk_commonapp " :

```
MiAdManager.applicationInit(Context context, String APP_ID, String configKey);
```


您可能还需要：如果接入 mi 的广告，请在初始化 Columbus 传入 APP_KEY 和 APP_SECRET:

```
AdGlobalSdk.initialize(Context context, String APP_KEY, String APP_SECRET);
```

// 注意：
第三方 SDK 初始化请放在 Mediation 初始化之前

(2) 申请相关参数需要时间，且新申请的 **PlacementID** 经常出现无广告填充的情况，这属于正常现象。在参数申请下来前，媒体方可以先使用本地配置填写测试 **PlacementID** 进行接入&测试，测试 ID 如下。

Facebook 测试 ID:

YOUR_PLACEMENT_ID

AdMob 测试 ID:

Native: ca-app-pub-3940256099942544/2247696110

Banner: ca-app-pub-3940256099942544/6300978111

Interstitial: ca-app-pub-3940256099942544/1033173712

RewardedVideo: ca-app-pub-3940256099942544/5224354917

TagID:

形式为 1.***.***, 自定义即可。

(3) Facebook 广告 **PlacementID** 审核

需要@媒体方在登录 Facebook 开发者账号（可能需要@产品添加权限）的手机上，在对应应用（申请该 **PlacementID** 的包名）中使用本地配置（正式 **PlacementID**），发起一次广告请求并成功展示。@产品在 Facebook 后台看到请求展示情况后，向 Facebook 发起审核流程。

12.1.3 使用本地配置自行测试

按照接入文档成功接入广告 SDK 后，您需要先使用本地配置自测广告相关功能是否符合预期。

(1) 首先，请根据上述相关参数和 Demo 中的 default-config.txt 撰写本地配置文件，可在 <https://www.json.cn/> 校验无误后您再进行测试（注：本地配置可能存在的问题：格式错误、参数不一致、权重错误、TagID 未加引号等）。

(2) 然后，将配置文件放入 assets 文件夹，如下将 setDefaultConfig 方法第二个参数传入 true:

MiAdManager.setDefaultConfig(defaultConfigStr, true);// 上线前务必改为 false

(3) 最后，您可以测试了。测试内容参考需求文档，主要测试各 DSP 广告是否能够成功加载、展示和打点。

您可以逐个测试某 DSP 的广告情况：比如您的应用首页底部广告位需要请求 ab、fb 和 mi 三家 DSP 的广告，您现在想测试 ab 的广告情况，那么您可以这样做：在配置文件中将 ab 的 weight 设置一个正整数，而除 ab 之外的 DSP（fb 和 mi）的 weight 都设置为 0。

(4) 测试时问题排查-日志过滤

首先，应用启动同时发送以下广播以打开日志开关：

adb shell am broadcast -a "com.xiaomi.ad.intent.DEBUG_ON"// Mediation 日志

adb shell am broadcast -a "com.xiaomi.analytics.intent.DEBUG_ON" // 打点日志

然后，在拉取展示广告时过滤指定日志：

adb logcat | grep "Analytics-Core-TrackingServer\|Analytics-Core-Server\|response\|MEDIATION"// 打点和 MEDIATION 全部日志

其他日志：MediationConfig（配置拉取情况）、
LOAD_AD\DSP_LOAD\LOAD_SUCCESS\NO_FILL_REASON\load fail:\（广告拉取情况：
发起请求\DSP 请求完成\成功请求到广告\无填充原因\DSP 广告加载失败原因）
PAGE_VIEW\VIEW\CLICK\CLOSE\VIDEO_FINISH\REWARDED_CALL\DISLIKE\
（广告位曝光\广告曝光\广告点击\广告关闭\奖励视频结束\给予奖励\不喜欢广告上报）.....

12.1.4 使用线上测试环境进行测试

（1）请将使用本地配置更改为 **false**，然后进行测试，发现问题请及时更改。

```
MiAdManager.setDefaultConfig(defaultConfigStr, false);
```

测试内容主要包括：配置拉取、广告请求、权重和打点等，功能是否和需求一致。

（2）初测后无问题，请媒体方参照模板发送提测邮件给客户端对接人员进行 **check**，无误后转发提测。

注：测试环境/正式环境切换请通过拨号盘按*672***，进入广告调试，点击使用 staging 环境/关掉 staging 环境，点击开发者模式/staging 环境信息查询查看是否切换成功。**

12.2 注意事项

12.2.1 AdReportHelper.reportPV 的正确调用

该方法用于广告位曝光打点，需要传入广告位 **id**，请媒体方正确调用。请在广告位展现时/有机会展示广告时调用进行广告位曝光上报，此方法与广告是否成功展示无关；方法调用次数与广告位曝光次数保持一致，请勿多次调用。

12.2.2 传入正确的参数

用到 12.1.2 中参数的地方，请使用提供给媒体的参数，而不要使用 **Demo** 中的参数。多个广告位的情况下，请在各广告位请求广告（初始化 **AdManager**）和上报广告位曝光（**reportPV**）时传入对应的 **TagId**，不要随意使用。

12.2.3 请求不到广告的常见原因

根据错误码排查原因，常见错误如下：

（1）10001，拉不到配置，检查拉取配置代码；

（2）10002，没有填充（Facebook 为 1001、AdMob 为 3，各 DSP 的广告错误码可在对应 Adapter 中 **onError** 或 **onAdFailedToLoad** 方法的打印中查看），检查是否连接海外 VPN、登录 Facebook 和 Google、哥伦布需要在嗅探 mock 数据，尝试在 Facebook 后台增加白名单；**新申请 ID 经常出现无填充为正常现象，只要指定 DSP 指定类型广告出现过一次即表明接入成功。**

（3）10005，没有 loader，检查是否通过 **addAdapterClass** 方法添加对应的 adapter。

12.2.4 广告预加载与重试

（1）可进行预加载

可在广告展示后进行预加载（奖励视频需要在关闭后再预加载），但要在进入广告位时 **check** 是否有广告，无广告还要再请求（避免加载不到就不会展示，从而不会再次加载），另外需要注意广告的有效期；

（2）避免高频率重复请求

请避免拉取不到就不停请求，需增加重试次数限制（比如 3 次，重试间隔分别为 5s、15s、30s）。