

汇川聚合sdk文档

版本号	变更内容	日期
1.0.0	初次发版	2020/07/14
1.0.1	1. 加入快手sdk和mobisdk的广告说明 2. 混淆说明修改 3.bug修改	2020/07/23

汇川聚合sdk文档

1. 接入准备

2. 接入SDK

2.1 添加依赖

2.2 全局配置

2.2.0 聚合配置

2.2.1 穿山甲广告

2.2.2 广点通广告

3. SDK初始化

3.1 初始化接口说明

4. 加载广告

4.1 开屏广告

4.1.1 开屏广告接口说明

4.1.2 调用开屏接口说明

4.1.3 显示开屏广告

4.2 信息流广告

4.2.1 信息流接口说明

4.2.2 调用信息流接口说明

4.2.3 显示信息流广告

4.3 插屏广告

4.3.1 插屏接口说明

4.3.2 调用插屏接口说明

4.3.3 显示插屏广告

4.4 激励视频广告

4.4.1 激励视频接口说明

4.4.2 调用激励视频接口说明

4.4.3 显示激励视频广告

4.5 全屏视频广告

4.5.1 全屏视频接口说明

1. 接入准备

接入广告聚合SDK前，请您联系官方平台申请您的AppId，广告位id等。

2. 接入SDK

SDK接入推荐使用aar的方式进行接入，请解压提供的广告SDK，在压缩包中找到mobimedia_1.0.0.aar。接入过程中可参考压缩包中的**SDKDemo**进行使用。

文件	说明	是否必要
mobimedia_1.0.0.aar	聚合的主要sdk支持包	是
mobimedia_tt_1.0.0.aar	穿山甲广告sdk支持包	否（接入穿山甲广告必要）
open_ad_sdk.aar	穿山甲广告sdk	否（接入穿山甲广告必要）
mobimedia_gdt_1.0.0.aar	广点通广告sdk支持包	否（接入广点通广告必要）
GDTSDK.unionNormal.4.230.1100.aar	广点通广告sdk	否（接入广点通广告必要）
mobimedia_ks_1.0.1.aar	快手广告sdk支持包	否（接入快手广告必要）
ks_adsdk-2.6.4.1.aar	快手广告sdk	否（接入快手广告必要）
mobimedia_mobi_1.0.1.aar	mobisdk广告sdk	否（接入mobisdk必要）
mobiad_1.0.0.aar	mobisdk广告sdk支持包	否（接入mobisdk必要）

具体说明如下。

2.1 添加依赖

导入Android Studio(推荐)，找到您的App工程下的libs文件夹，将上面提到的aar拷贝到该目录下，然后在项目根build.gradle文件中，以libs目录作为仓库地址添加本地仓库，然后在需要依赖的 module的 build.gradle文件中，添加SDK的dependencies依赖，代码如下：

```

android {
    compileOptions {
        targetCompatibility JavaVersion.VERSION_1_8
        sourceCompatibility JavaVersion.VERSION_1_8
    }
}

allprojects {
    repositories {
        flatDir { dirs 'libs' }
    }
}

```

```

//汇川聚合sdk 必须要
implementation files('libs/mobi_mediation_1.0.0.aar')

//汇川聚合sdk 穿山甲支持包
implementation files('libs/mobi_mediation_tt_1.0.0.aar')
//穿山甲sdk
implementation files('libs/open_ad_sdk.aar')

//汇川聚合sdk 优量汇支持包
implementation files('libs/mobi_mediation_gdt_1.0.0.aar')
//优量汇sdk
implementation files('libs/GDTSdk.unionNormal.4.230.1100.aar')

//汇川聚合sdk 快手支持包
implementation files('libs/mobi_mediation_ks_1.0.0.aar')
//快手sdk
implementation files('libs/ks_adsdk-2.6.4.1.aar')

//汇川聚合sdk mobisdk支持包
implementation files('libs/mobi_mediation_mobi_1.0.0.aar')
//mobisdk
implementation files('libs/mobiad_1.0.0.aar')

```

补充：SDK也支持jar包的方式接入，如果需要采用jar包方式接入，请解压sdk的aar找到jar包，同时将所有依赖的xml资源拷贝到您的工程中，即可使用。

2.2 全局配置

2.2.0 聚合配置

1. 混淆

```

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

```

2.2.1 穿山甲广告

1. 权限配置

```
<!-- 必要权限 -->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" /> <!--
可选权限 -->
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.REQUEST_INSTALL_PACKAGES" />
<uses-permission android:name="android.permission.GET_TASKS" /> <!-- 可选，穿山甲
提供“获取地理位置权限”和“不给予地理位置权限，开发者传入地理位置参数”两种方式上报用户位置，两种
方式均可不选，添加位置权限或参数将帮助投放定位广告 -->
<!-- 请注意：无论通过何种方式提供给穿山甲用户地理位置，均需向用户声明地理位置权限将应用于穿山
甲广告投放，穿山甲不强制获取地理位置信息 -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" /> <!--
如果有视频相关的广告且使用textureView播放，请务必添加，否则黑屏 -->
<uses-permission android:name="android.permission.WAKE_LOCK" />
```

2. 其他配置

在 res/xml 目录下，新建一个 xml 文件 file_paths，在该文件中添加如下代码：

```
<provider
    android:name="com.bytedance.sdk.openadsdk.TTFileProvider"
    android:authorities="${applicationId}.TTFileProvider"
    android:exported="false"
    android:grantUriPermissions="true">
    <meta-data
        android:name="android.support.FILE_PROVIDER_PATHS"
        android:resource="@xml/file_paths" />
</provider>
```

```
-keep class com.bytedance.sdk.openadsdk.** { *; }
-keep public interface com.bytedance.sdk.openadsdk.downloadnew.** {*; }
-keep class com.pgl.sys.ces.* {*; }
```

详细产考：[穿山甲广告文档](#)

2.2.2 广点通广告

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

```

<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<!-- 可选，如果需要精确定位的话请加上此权限 -->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.REQUEST_INSTALL_PACKAGES" />

<application android:usesCleartextTraffic="true" >
    <uses-library
        android:name="org.apache.http.legacy"
        android:required="false" />
    <!-- 声明SDK所需要的组件 -->
    <service
        android:name="com.qq.e.comm.DownloadService"
        android:exported="false" />
    <!-- 请开发者注意字母的大小写，ADActivity，而不是AdActivity -->
    <activity
        android:name="com.qq.e.ads.ADActivity"

    android:configChanges="keyboard|keyboardHidden|orientation|screenSize" />
        <activity
            android:name="com.qq.e.ads.PortraitADActivity"

    android:configChanges="keyboard|keyboardHidden|orientation|screenSize"
        android:screenOrientation="portrait" />
        <activity
            android:name="com.qq.e.ads.LandscapeADActivity"

    android:configChanges="keyboard|keyboardHidden|orientation|screenSize"
        android:screenOrientation="landscape" />
</application>

```

/res/xml/gdt_file_path.xml文件，文件内容如下

```

<paths>
<!-- 这个下载路径也不可以修改，必须为GDTDOWNLOAD -->
<external-cache-path
    name="gdt_sdk_download_path1"
    path="com_qq_e_download" />
<cache-path
    name="gdt_sdk_download_path2"
    path="com_qq_e_download" />
</paths>

```

详细产考：[广点通\(优量汇\)文档](#)

SDK要求最低系统版本为API 16，对于适配了Android6.0以上(API >= 23)的App，建议开发者在获得了动态权限之后，调用SDK的初始化代码，否则SDK可能受影响。

3. SDK初始化

请在您应用的 Application 的 onCreate() 方法中调用以下代码来初始化聚合广告sdk。

```
public class App extends Application {

    @Override
    public void onCreate() {
        super.onCreate();
        MobiPubSdk.init(this, "3014");

        //这一步可要，可不要
        //目的是为了可以在第一次sdk初始化成功的时候，可以把启动页迅速的展示出来
        TTSdkInit.init(this,
            buildConfig("5001121", "测试app"),
            BuildConfig.DEBUG);
    }

    private TTAdConfig buildConfig(String appId, String appName) {
        return new TTAdConfig.Builder()
            .appId(appId)
            .appName(appName)
            .useTextureView(true) //使用TextureView控件播放视频,默认为
//SurfaceView,当有SurfaceView冲突的场景,可以使用TextureView
            .titleBarTheme(TTAdConstant.TITLE_BAR_THEME_DARK)
            .allowShowNotify(true) //是否允许sdk展示通知栏提示
            .allowShowPageWhenScreenLock(true) //是否在锁屏场景支持展示广告落地
//页
            .debug(false) //测试阶段打开,可以通过日志排查问题,上线时去除该调用
            .directDownloadNetworkType(TTAdConstant.NETWORK_STATE_WIFI,
                TTAdConstant.NETWORK_STATE_MOBILE) //允许直接下载的网络状
//态集合
            .supportMultiProcess(true) //是否支持多进程
            .build();
    }
}
```

3.1 初始化接口说明

```
/**
 * 聚合sdk初始化的入口
 *
 * @param context
 * @param appId
```

```

*/
public static void init(Context context, String appId);

/**
 * @param context
 * @param appId
 * @param isDebug 建议上线的时候用true
 */
public static void init(Context context, String appId, boolean isDebug);

```

4. 加载广告

4.1 开屏广告

4.1.1 开屏广告接口说明

```

public interface ISplashAdListener {
    /**
     * 广告开始展示
     *
     * @param providerType
     */
    void onAdStartRequest(@NonNull String providerType);

    /**
     * 广告曝光
     * @param type
     */
    void onAdExposure(String type);

    /**
     * 广告点击
     * @param type
     */
    void onAdClick(String type);

    /**
     * 广告关闭
     * @param type
     */
    void onAdClose(String type);

    /**
     * 广告失败
     *
     */
    void onAdFail(List<StrategyError> strategyErrorList);

    /**

```

```

        * 广告加载成功
        *
        */
        void onAdLoad(String providerType, SplashAdView view);
    }

```

4.1.2 调用开屏接口说明

```

@MainThread
public static void loadSplash(Activity activity,
                             ViewGroup splashContainer,
                             @Nullable BaseSplashSkipView skipView,
                             AdParams adParams,
                             ISplashAdListener listener)

```

参数	说明	是否能为空
activity	activity对象	不能为空
splashContainer	显示启动页广告所需容器	不能为空
skipView	页面的倒计时按钮	可以为空
adParams	广告显示的必要参数	不能为空
listener	广告接口	不能为空

4.1.3 显示开屏广告

```

AdParams adParams = new AdParams.Builder()
    .setCodeId("3014001")
    .setImageAcceptedSize(1080, 1920)
    .setSupportDeepLink(true)
    .build();

MobiPubSdk.loadSplash(this, clRoot, view, adParams, new ISplashAdListener() {
    @Override
    public void onAdFail(List<StrategyError> strategyErrorList) {
        for (StrategyError strategyError : strategyErrorList) {
            LogUtils.e(TAG, "onLoadFailed type : " + strategyError.getProviderType()
                + " faildCode : " + strategyError.getCode() + ", faildMsg: " +
                strategyError.getMessage());
        }
        delayToHome(1000);
    }

    @Override
    public void onAdStartRequest(@NonNull String providerType) {
        Log.e(TAG, "onAdStartRequest providerType: " + providerType);
    }
}

```



```

}

@Override
public void onAdClick(String type) {
    Log.e(TAG, "onAdClicked " + "providerType: " + type);
    //          startActivity(new Intent(SplashActivity.this,
MainActivity.class));
}

@Override
public void onAdExposure(String type) {
    Log.e(TAG, "onAdExposure " + "providerType: " + type);
}

@Override
public void onAdClose(String type) {
    Log.e(TAG, "onAdDismissed " + "providerType: " + type);

    next();
}

@Override
public void onAdLoad(String providerType, SplashAdView view) {
    Log.e(TAG, "onAdLoaded " + "providerType: " + providerType);
    if (view != null) {
        view.show();
    }
}
});

```

4.2 信息流广告

4.2.1 信息流接口说明

```

public interface IExpressListener {
    void onAdExposure(String type); //广告曝光

    void onAdClick(String type); //广告点击

    void onAdClose(String type); //广告关闭

    void onAdRenderSuccess(String type); //广告渲染成功

    /**
     * 因为广告点击等原因离开当前 app 时调用
     */
    default void onADLeftApplication(String type) {

```

```

    }

    /**
     * 广告展开遮盖时调用
     */
    default void onADOpenOverlay(String type) {

    }

    /**
     * 广告关闭遮盖时调用
     */
    default void onADCloseOverlay(String type) {

    }

    /**
     * 广告失败
     *
     */
    void onAdFail(List<StrategyError> strategyErrorList);

    /**
     * 广告加载成功
     *
     */
    void onAdLoad(String providerType, ExpressAdView view);
}

```

4.2.2 调用信息流接口说明

```

@MainThread
public static void loadNativeExpress(Context context,
                                     ViewGroup viewContainer,
                                     AdParams adParams,
                                     IExpressListener listener)

```

参数	说明	是否能为空
activity	context对象	不能为空
viewContainer	显示信息流广告所需容器	不能为空
adParams	广告显示的必要参数	不能为空
listener	广告接口	不能为空

4.2.3 显示信息流广告

```

AdParams adParams = new AdParams.Builder()
    .setCodeId("3014002")
    .setAdCount(1)
    .setImageAcceptedSize(640, 320)
    .setExpressViewAcceptedSize(350, 0)
    .build();

//native
MobiPubSdk.loadNativeExpress(this, flContainer, adParams, new
IExpressListener() {
    @Override
    public void onAdClick(String type) {
        LogUtils.e(TAG, "onAdClick type : " + type);
    }

    @Override
    public void onAdLoad(String type, ExpressAdView view) {
        LogUtils.e(TAG, "onAdLoad type : " + type);
        if (view != null) {
            view.show();
        }
    }

    @Override
    public void onAdFail(List<StrategyError> strategyErrorList) {
        for (StrategyError strategyError : strategyErrorList) {
            LogUtils.e(TAG, "onLoadFailed type : " + strategyError.getProviderType()
                + " faildCode : " + strategyError.getCode() + ", faildMsg: " +
strategyError.getMessage());
        }
    }

    @Override
    public void onAdClose(String type) {
        LogUtils.e(TAG, "onAdDismissed type : " + type);
    }

    @Override
    public void onAdRenderSuccess(String type) {
        LogUtils.e(TAG, "onAdRenderSuccess type : " + type);
    }

    @Override
    public void onAdExposure(String type) {
        LogUtils.e(TAG, "onAdShow type : " + type);
    }
});

```

4.3 插屏广告

4.3.1 插屏接口说明

```
public interface IInteractionAdListener {  
    /**  
     * 广告曝光  
     *  
     * @param type  
     */  
    void onAdExposure(String type);  
  
    /**  
     * 广告点击  
     *  
     * @param type  
     */  
    void onAdClick(String type);  
  
    /**  
     * 广告关闭  
     *  
     * @param type  
     */  
    void onAdClose(String providerType);  
  
    /**  
     * 广点通打开  
     *  
     * @param type  
     */  
    default void onGdtOpened(String type) {  
  
    }  
  
    /**  
     * 广告缓存  
     *  
     * @param type  
     */  
    default void onGdtCached(String type) {  
  
    }  
  
    /**  
     * 因为广告点击等原因离开当前 app 时调用  
     *  
     * @param type  
     */  
}
```

```

default void onGdtLeftApplication(String type) {

}

/**
 * 广告失败
 *
 */
void onAdFail(List<StrategyError> strategyErrorList);

/**
 * 广告加载成功
 *
 */
void onAdLoad(String providerType, InteractionAdView view);
}

```

4.3.2 调用插屏接口说明

```

@MainThread
public static void loadInteractionExpress(Activity activity,
                                         AdParams adParams,
                                         IInteractionAdListener listener)

```

参数	说明	是否能为空
activity	activity对象	不能为空
adParams	广告显示的必要参数	不能为空
listener	广告接口	不能为空

4.3.3 显示插屏广告

```

AdParams adParams = new AdParams.Builder()
    .setCodeId("3014003")
    .setAutoShowAd(true)
    .build();
MobiPubSdk.loadInteractionExpress(this, adParams, new IInteractionAdListener()
{
    @Override
    public void onAdFail(List<StrategyError> strategyErrorList) {
        for (StrategyError strategyError : strategyErrorList) {
            LogUtils.e(TAG, "onLoadFailed type : " + strategyError.getProviderType()
                + " faildCode : " + strategyError.getCode() + ", faildMsg: " +
strategyError.getMessage());
        }
    }
}

```

```

@Override
public void onGdtOpened(String type) {
    LogUtils.e(TAG, "onADOpened type : " + type);
}

@Override
public void onAdExposure(String type) {
    LogUtils.e(TAG, "onADExposure type : " + type);
}

@Override
public void onAdLoad(String type, InteractionAdView view) {
    LogUtils.e(TAG, "onAdLoad type : " + type);
    if (view != null) {
        view.show();
    }
}

@Override
public void onAdClick(String type) {
    LogUtils.e(TAG, "onAdLoad type : " + type);
}

@Override
public void onAdClose(String providerType) {
    LogUtils.e(TAG, "onAdClose type : " + providerType);
}

@Override
public void onGdtCached(String type) {
    LogUtils.e(TAG, "onCached type : " + type);
}
});

```

4.4 激励视频广告

4.4.1 激励视频接口说明

```

public interface IRewardAdListener {
    /**
     * 广告曝光
     *
     * @param type
     */
    void onAdExpose(String type);
    /**
     * 广告点击

```

```

    *
    * @param type
    */
    void onAdClick(String type);

    /**
     * 广告关闭
     *
     * @param type
     */
    void onAdClose(String providerType);

    /**
     * 视频播放完成
     *
     * @param type
     */
    void onVideoComplete(String providerType);

    /**
     * 激励视频广告奖励
     *
     * @param type
     */
    void onRewardVerify(String providerType, boolean rewardVerify, int
rewardAmount, String rewardName);

    /**
     * 广告缓存完成
     *
     * @param type
     */
    void onCached(String type);

    /**
     * 广告失败
     *
     */
    void onAdFail(List<StrategyError> strategyErrorList);

    /**
     * 广告加载成功
     *
     */
    void onAdLoad(String providerType, RewardAdView view);
}

```

4.4.2 调用激励视频接口说明

```

@MainThread
public static void loadInteractionExpress(Activity activity,
                                         AdParams adParams,
                                         IInteractionAdListener listener)

```

参数	说明	是否能为空
activity	activity对象	不能为空
adParams	广告显示的必要参数	不能为空
listener	广告接口	不能为空

4.4.3 显示激励视频广告

```

AdParams adParams = new AdParams.Builder()
    .setAutoShowAd(false)
    .setCodeId("3014004")
    .build();

MobiPubSdk.loadRewardView(this, adParams, new IRewardAdListener() {
    @Override
    public void onAdFail(List<StrategyError> strategyErrorList) {
        for (StrategyError strategyError : strategyErrorList) {
            LogUtils.e(TAG, "onLoadFailed type : " + strategyError.getProviderType()
                + " faildCode : " + strategyError.getCode() + ", faildMsg: " +
strategyError.getMessage());
        }
    }

    @Override
    public void onAdLoad(String type, RewardAdView view) {
        LogUtils.e(TAG, "onAdLoad type : " + type);
        if (view != null) {
            view.show();
        }
    }

    @Override
    public void onAdExpose(String type) {
        LogUtils.e(TAG, "onAdExpose type : " + type);
    }

    @Override
    public void onAdShow(String type) {
        LogUtils.e(TAG, "onAdGdtShow type : " + type);
    }

    @Override

```



```

public void onAdClick(String type) {
    LogUtils.e(TAG, "onAdLoad type : " + type);
}

@Override
public void onAdClose(String providerType) {
    LogUtils.e(TAG, "onAdClose type : " + providerType);
}

@Override
public void onVideoComplete(String providerType) {
    LogUtils.e(TAG, "onVideoComplete type : " + providerType);
}

@Override
public void onSkippedVideo(String providerType) {
    LogUtils.e(TAG, "onSkippedVideo type : " + providerType);
}

@Override
public void onRewardVerify(String providerType, boolean rewardVerify, int
rewardAmount, String rewardName) {
    LogUtils.e(TAG, "onRewardVerify type : " + providerType);
}

@Override
public void onCached(String type) {
    LogUtils.e(TAG, "onCached type : " + type);
}
});

```

4.5 全屏视频广告

4.5.1 全屏视频接口说明

```

public interface IFullScreenVideoAdListener {
    /**
     * 全屏广告显示
     * @param providerType
     */
    void onAdExposure(String providerType);

    /**
     * 全屏广告点击
     * @param providerType
     */
    void onAdClick(String providerType);

    /**

```

```

    * 全屏广告关闭
    * @param providerType
    */
void onAdClose(String providerType);

/**
 * 全屏广告缓存
 * @param providerType
 */
void onCached(String providerType);

/**
 * 全屏广告播放完成
 * @param providerType
 */
void onVideoComplete(String providerType);

/**
 * 全屏广告跳过
 * @param providerType
 */
void onSkippedVideo(String providerType);

/**
 * 广告失败
 *
 */
void onAdFail(List<StrategyError> strategyErrorList);

/**
 * 广告加载成功
 *
 */
void onAdLoad(String providerType, FullscreenAdView view);
}

```

4.5.2 调用全屏视频接口说明

```

@MainThread
public static void loadFullscreen(Activity activity,
                                AdParams adParams,
                                IFullScreenVideoAdListener listener)

```

参数	说明	是否能为空
activity	activity对象	不能为空
adParams	广告显示的必要参数	不能为空
listener	广告接口	不能为空

4.5.3 显示全屏视频广告

```
AdParams adParams = new AdParams.Builder()
    .setCodeId("3014005")
    .setOrientation(ConstantValue.VERTICAL)
    .build();

MobiPubSdk.loadFullscreen(this, adParams, new IFullscreenVideoAdListener() {
    @Override
    public void onAdFail(List<StrategyError> strategyErrorList) {
        for (StrategyError strategyError : strategyErrorList) {
            LogUtils.e(TAG, "onLoadFailed type : " + strategyError.getProviderType()
                + " faildCode : " + strategyError.getCode() + ", faildMsg: " +
strategyError.getMessage());
        }
    }

    @Override
    public void onAdExposure(String type) {
        LogUtils.e(TAG, "onAdShow ");
    }

    @Override
    public void onAdLoad(String type, FullscreenAdView view) {
        LogUtils.e(TAG, "onAdLoad ");
        if (view != null) {
            view.show();
        }
    }

    @Override
    public void onCached(String type) {
        LogUtils.e(TAG, "onCached ");
    }

    @Override
    public void onAdClose(String providerType) {
        LogUtils.e(TAG, "onAdClose ");
    }

    @Override
    public void onVideoComplete(String providerType) {
        LogUtils.e(TAG, "onVideoComplete ");
    }

    @Override
    public void onSkippedVideo(String providerType) {
        LogUtils.e(TAG, "onSkippedVideo ");
    }
}
```

```
@Override
public void onAdClick(String providerType) {
    LogUtils.e(TAG, "onAdVideoBarClick ");
}
});
```