## Vest-SDK

**一．SDK简介**

Vest-SDK由三个依赖库组成，分别是：

1. vest-core核心库，主要是一些工具类方法和Adjust、Thinking Data数据记录模块。
2. vest-shf实现了A/B面切换开关，该开关配置在服务器端，需要联网才能获得。
3. vest-sdk实现游戏展示框架，只支持显示H5游戏。

**二．SDK使用**

1. 把加密后的配置放到assets目录
2. Application中初始化SDK，传入配置文件名

VestSDK.init(getBaseContext(), "config");

1. 其中config文件的原始内容如下：

{

"channel": "website",

"brand": "test",

"country": "br",

"shf\_base\_domain": "https://shf.test.baowengame.com",

"shf\_spare\_domains": [

"https://www.ozt4axm9.com",

"https://www.6r4hx6e2.com",

"https://www.cictnjac.com"

],

"adjust\_app\_id": "3h9btar5b3i8",

"adjust\_event\_start": "15wkgy",

"adjust\_event\_greeting": "h5twnz",

"adjust\_event\_access": "mza6nh",

"adjust\_event\_updated": "gz7ht9",

"facebook\_app\_id": "",

"facebook\_client\_token": "",

"thinking\_data\_app\_id": "4edaf2728be644dd83f04c54d60f0fa0",

"thinking\_data\_host": "https://data.kneil.com/",

"httpdns\_auth\_id": "95244",

"httpdns\_app\_id": "ADHPTGT49H4T8CJ6",

"httpdns\_des\_key": "CM1BMqgH",

"httpdns\_ip": "43.132.55.55"

}

数据分为五大块

（1）A/B面开关参数，每个品牌需要更换

* channel：渠道号，功能上没有用到，但是也要按照厂商给的填写正确
* brand：品牌号，开关按照品牌独立返回
* country：国家，用于推广活动统计中区分数据
* shf\_base\_domain：开关服务器主域名，每个品牌配置一个
* shf\_spare\_domains：备用服务器域名，当主域名无法访问时，轮询访问备用域名，备用域名每个品牌都是一样，主要是为了减少域名的购买成本。

（2）Adjust统计的有关参数，每个品牌需要更换

* adjust\_app\_id：用于初始化Adjust SDK
* adjust\_event\_start：记录程序首次启动次数
* adjust\_event\_greeting：记录A/B开关请求成功事件
* adjust\_event\_access：记录进入B面游戏事件
* adjust\_event\_updated：没有用

（3）Facebook相关参数（SDK没有集成Facebook功能，参数弃用）

* facebook\_app\_id：AppId
* facebook\_client\_token：Client Token

（4）Thinking Data相关参数，每个品牌使用一样的参数

* thinking\_data\_app\_id：用于初始化Thinking Data SDK
* thinking\_data\_host：厂商服务器地址，用于接收Thinking Data服务器的回传事件

（5）HttpDns相关参数，每个品牌使用一样的参数

* httpdns\_auth\_id： 授权
* httpdns\_app\_id： AppId
* httpdns\_des\_key： DES解密密钥
* httpdns\_ip： HttpDns服务器地址，向该地址发起请求可以拿到域名的Ip

1. 实现A/B面切换，开关在厂商后台控制。打开开关表示跳转到B面，调用方法onShowOfficialGame；关闭开关表示跳转到A面，调用方法onShowVestGame。

VestSHF.getInstance().inspect(this, new VestInspectCallback() {

//这里跳转到A面，A面请自行实现

@Override

public void onShowVestGame() {

Log.d(TAG, "show vest game");

Intent intent = new Intent(getBaseContext(), VestGameActivity.class);

intent.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

startActivity(intent);

AppTestSDKActivity.this.finish();

}

//这里跳转到B面，B面由SDK提供，使用VestSDK.gotoGameActivity()方法跳转

@Override

public void onShowOfficialGame(String url) {

Log.d(TAG, "show official game: " + url);

VestSDK.gotoGameActivity(getBaseContext(), url);

AppTestSDKActivity.this.finish();

}

});

**三．SDK功能模块说明**

1. Adjust统计，实现类code.sdk.core.manager.AdjustManager。

在vest-core中，主要用于统计有关事件。

1. Thinking Data统计，实现类code.sdk.core.manager.ThinkingDataManager。

在vest-core中，另一个事件统计SDK。

1. HttpDns放劫持，实现类code.sdk.httpdns.HttpDnsMgr。

在vest-sdk中，用来实现域名防劫持功能，在JavascriptBridge中由B面游戏调用。**该功能依赖的okhttp版本要求 <= 4.4.1，请注意您工程中的okhttp版本号。**

1. OneSignal推送，实现类code.sdk.manage.OneSignalManager

在vest-sdk中，用来实现Notification推送，只在B面游戏启用该功能，配置中的country字段能按照国家初始化OneSignal，以便按国家进行推送。

1. JavascriptBridge，实现类code.sdk.bridge. JavascriptBridge。

实现B面游戏在WebView中与Android原生环境的互相调用。

1. WebView，实现类code.sdk.ui.WebActivity。

用于展示B面游戏的UI实现

1. 配置存储中心，实现类code.sdk.core.util .ConfigPreference。

用来存储从assets读取到的配置，也就是VestSDK.init(getBaseContext(), "config")传入的配置。之所以要存储起来是为了让在vest-sdk和vest-shf中都能读取到配置，因为vest-sdk和vest-shf作为独立的sdk，无法与vest-core共享内存，只能用Preference作为中介实现配置共享。

**四．关于过包技巧**

1. SDK本身不提供代码混淆，要是审核遇到问题，可以尝试修改工程代码，业务代码不建议修改。可以自己添加类或者方法，并确保被业务代码引用，可以到github找一些看起来写得不怎么样的代码。
2. 最好有50%的业务方法能引用到垃圾代码，这都是要手动加的，这样就能在代码审查中看到比较复杂的类引用关系。引用垃圾代码时，要保证那些垃圾代码不会被执行。可以用一个if判断永假来实现，如下代码：

* JunkCode.java 这是垃圾代码的永假实现

public class JunkCode extends Activity {

public static final String TAG = "CzlfbdjoActivity";

public static void inject() {

if (TAG.compareTo("hxpdfskevuwqmatbnyjl") == 0) {

//引用的垃圾代码，这里永远不执行

ImtavxoActivity instanceImtavxoActivity = new ImtavxoActivity();

instanceImtavxoActivity.kosjbfp();

} else {

// else的情况要实现，要不然proguard混淆阶段会因为if的永假条件，代码会被删除掉

// 最好是不涉及功能的，打印日志，setProperty都可以

System.setProperty("property1", "1");

}

}

}

* FileUtil.java 这是SDK的业务代码，引用了永假条件的垃圾代码

public final class FileUtil {

public static void ensureFile(File file) {

//这里引用垃圾代码

JunkCode.inject();

if (file != null && !file.exists()) {

ensureDirectory(file.getParentFile());

try {

file.createNewFile();

} catch (IOException e) {

e.printStackTrace();

}

}

}

…

}

1. SDK分标准版和精简版。标准版包含OneSignal和HttpDns功能，精简版不包含。当过包困难时不妨使用精简版，去掉以下依赖即可切换到精简版。

implementation 'com.onesignal:OneSignal:4.8.6'

implementation 'io.github.dnspod:httpdns-sdk:4.4.0-intl'

implementation 'androidx.room:room-rxjava2:2.1.0'