0 更新日志

日期	调整内容	修改人
2020/09/25	1.调整为统一服务API，故每个APP所拥有的独立字段在方法中以参数的形式手动传入（如app_version） 2.改为jar包	柯圣川
2020/10/21	1.经测试发现，jar包无法把相关配置文件打进去，故现在重新使用arr包 2.将fastjson的依赖打入arr包	柯圣川
2020/10/23	1.修复了linux环境下某些依赖与windows环境不兼容的问题（在windows中测试第三方依赖fastjson可以打进arr并成功运行，但在linux环境找不到依赖，推测是由于windows下依赖的路径\lib在linux中无法识别导致的。目前解决方式：使用安卓自带的json）	柯圣川
2020/10/30	1.增加了【7.埋点方法 - 追踪APP生命周期】 2.根据VDF平台对数据格式的规定，调整了传输数据格式，见附2 3.根据VDF平台对数据格式的规定，调整了 6/8/9 埋点方法的参数 4.为减少传输开销，埋点数据的字段名映射为编号再传输，见附3	柯圣川
2020/11/11	1.去掉了【7.埋点方法 - 追踪APP生命周期】 2.去掉了计时方法，该方法业务性太强，在目前场景不适用 3.增加进入事件&退出事件，在进入/退出页面时调用	柯圣川
2020/11/17	1.所有方法加上 appId（app唯一标识），curAactivity（当前activity名称）字段 2.某些场景中有唤起别的app，故增加 13跳转app的方法 3.增加追踪 Fragment 的方法，包含：进入、退出、跳转（2.5 ~ 2.7）	柯圣川
2020/11/20	1.修改了某些不完善的用例 2.对隐式跳转如何使用跳转方法做了说明	柯圣川
2020/12/18	解决了SDK在车机端读取SSL证书失败的问题	柯圣川
2020/12/25	1.增加设置埋点模式说明，见1.2 设置埋点模式 2.增加独立集成模式下的埋点方法（3.1 ~ 3.8）	柯圣川
2021/01/05	1.添加统一服务与独立集成的数据上报方法说明，见2.2 数据上报方法与3.2 数据上报方法 2.添加5 注意事项	柯圣川
2021/01/21	1.初始化SDK的方法有改变，增加了设置埋点模式，去掉了设置服务URL（将由SDK自动设置），请参见1.2 初始化 SDK	柯圣川

日期	调整内容	修改人
2021/03/19	1.由于接收数据端URL变更，SDK更新了测试与生产环境发送数据的URL 上个版本 SDK 的模式设置为 DEV / PROD 均向测试环境发送数据 此版本： configOptions.mode(ModeEnum.DEV) 向测试环境发送数据 configOptions.mode(ModeEnum.PROD) 向生产环境发送数据	柯圣川
2021/03/31	1.增加字段 usr_id（G33） 2.增加方法1.7 ⑥设置用户ID，用于设置新字段 usr_id 3.为了方便定位问题与防止SDK包版本混淆，SDK的名称增加版本号，请注意引入新的SDK时修改代码中的引入语句	柯圣川
2021/04/25	1.由于业务调整，将 G12、G23、G24、G25、G28 改为非必需字段 2.更新了1.1 引入SDK，增加引入方式②使用本地 jar 包引入 SDK 所需依赖，对于本地环境无外网无法使用 gradle 的用户，请尝试这种方式引入	柯圣川
2021/05/21	1.增加注意事项5.2 用户自定义字段不支持直接设置JsonArray	柯圣川
2021/07/06	1.修复了在生产环境下的日志禁用问题 2.此外新增通过初始化配置可以自行控制日志打印 configOptions.enableLog(true)为开启打印日志 configOptions.enableLog(false)为关闭打印日志;	姬晓亚
2021/08/04	1.对普通事件以外埋点方法涉及 trackJump* 类型的埋点，使参数具备默认“事件名称”的功能	姬晓亚
2021/09/09	1.优化了SDK中运营商字段特殊环境下会自动获取的问题	姬晓亚
2021/12/03	1.增加远程禁用指定字段功能	姬晓亚
2023/06/06	1.DCP测试环境请求接口变动	洪祥祥
2024/05/30	1. 特殊处理：针对埋点字段值长度超过限制8191且为json字符串格式，则进行截取后补全json格式。（只针对上传格式为json字符串） 2.文档补充相关说明（详见1.10 其它说明）	洪祥祥
2025/03/28	1. 将数据发送使用的 httpclient 修改为okhttp .**具体未修改相关逻辑，若更换包参考下方引入sdk依赖部分修改（删除旧引入更换新引入即可其它无需调整）**	洪祥祥

日期	调整内容	修改人
2025/04/08	移除manifest中无效配置APP_KEY.(仅移除无效配置，其它未调整)	洪祥祥

1 基础配置

1.1 引入SDK

由于SDK中引用了外部依赖，用户在使用SDK时需要同时导入SDK的依赖jar包，这里提供两种引入SDK所需依赖的方式

①通过 gradle 引入 SDK 所需依赖

导入方式：**将 SDK 的 arr 包放入 lib 目录，然后在 build.gradle 加入以下配置引入 SDK 和 SDK 所需的依赖（目前只有 okhttp）**

`注意：这里用的通配符引入，使用新的 arr 时应先删除原来的 arr`

```
implementation fileTree(include: ['*.jar', '*.aar'], dir: 'libs')
implementation 'com.squareup.okhttp3:okhttp:4.12.0'
```

注：若报错`More than one file was found with OS independent path 'META-INF/DEPENDENCIES'`，请在 build.gradle 中加入

```
android {
    packagingOptions {
        exclude 'META-INF/DEPENDENCIES'
        exclude 'META-INF/LICENSE'
        exclude 'META-INF/LICENSE.txt'
        exclude 'META-INF/license.txt'
        exclude 'META-INF/NOTICE'
        exclude 'META-INF/NOTICE.txt'
        exclude 'META-INF/notice.txt'
        exclude 'META-INF/ASL2.0'
        exclude("META-INF/*.kotlin_module")
    }
}
```

②使用本地 jar 包引入 SDK 所需依赖

由于部分使用者本地环境无外网，无法使用 gradle 下载依赖，我们自`OAS-20210425-v1-release`版本开始将 SDK 依赖的 jar 包与 SDK 本身的 arr 包一并提供给使用者

导入方式：**将 SDK 的 arr 包与 SDK 依赖的本地 jar 包一起放入 lib 目录，然后在 build.gradle 加入以下配置同时引入 SDK 和 SDK 所需的依赖 jar 包**

`注意：这里用的通配符引入，使用新的 arr 时应先删除原来的 arr`

```
implementation fileTree(dir: 'libs', include: ['*.jar', '*.aar'])
```

1.2 初始化 SDK

静态方法：**startWithConfigOptions**

参数：

参数	说明
Context context	App 的 Context
SAConfigOptions saConfigOptions	SDK 的配置项

返回值类型：void

使用示例：

在程序入口Application的onCreate()中调用

```
public class MyApplication extends Application {

    @Override
    public void onCreate() {

        super.onCreate();

        // 初始化SDK
        initSensorsDataAPI();
    }

    /**
     * 初始化 SDK
     */
    private void initSensorsDataAPI() {

        SAConfigOptions configOptions = new SAConfigOptions();

        // 设置埋点模式
        // UNIFIED_SERVICE：统一服务
        // INTEGRATION：独立集成）
        configOptions.buriedPointMode(BuriedPointModeEnum.UNIFIED_SERVICE);

        // 设置SDK的模式，如DEV PROD等
        // DEV模式为开发模式，能够输出调试信息，使用枚举定义这两种模式
        configOptions.mode(ModeEnum.PROD);

        // 设置预警
        configOptions.alarm(AlarmModeEnum.LOG);

        // 每缓存 ？ 条日志发送一次
        configOptions.setFlushBulkSize(50);
        // 设置每 ？ 毫秒发送一次
        configOptions.setFlushInterval(5 * 1000);
        // 设置本地数据缓存上限值为 16 MB
```

```
        configOptions.setMaxCacheSize(16 * 1024 * 1024);

        // 初始化 SDK
        OASAPI.startWithConfigOptions(this, configOptions);

        // 注意：如果要把初始化与设置vin/basicAuth等操作写在一起
        // 应在初始化 startWithConfigOptions() 之后再调用这些方法
    }
}
```

1.3 设置埋点模式

说明：初始化 SDK 时，设置SDK的埋点模式，分为统一服务和独立集成

> 统一服务：由于无法获取APP相关信息，这种模式只会采集运行环境相关的信息；APP相关信息将作为参数传入
>
> 独立集成：当SDK集成到每个应用APP内部时，可以开启这种模式，SDK会自动采集APP相关信息和运行环境相关的信息，注意区别于统一服务的是：APP相关信息不需要再通过参数传入

`此参数必须设置，否则抛出异常；设置后请用户调用相应类别的埋点方法进行数据采集`

`如：设置埋点模式为统一服务则只能调用 2.1 ~ 2.8 的方法;设置埋点模式为独立集成则只能调用 3.1 ~ 3.8 的方法`

方法：**buriedPointMode**

参数：

参数	说明
BuriedPointModeEnum buriedPointMode	埋点的模式 UNIFIED_SERVICE：统一服务 INTEGRATION：独立集成

返回值类型：void

异常：未设置埋点模式或埋点模式与方法不匹配时会抛出异常 `BuriedPointModeNotMatchMethodException`，请用户在调用埋点方法时手动捕获这个异常以避免这一错误

使用示例：

```
// 初始化不定义告警模式，默认NONE数据都发送
configOptions.alarm(BuriedPointModeEnum.UNIFIED_SERVICE);
```

调试信息：

```
I/SA.AnalyticsMessages: 已设置埋点模式为【统一服务】
```

1.4 查看调试信息

说明：初始化 SDK 时，配置SDK的模式，可输出调试信息

方法：**mode**

参数：

参数	说明
ModeEnum mode	SDK的模式 DEV：开发模式，能够输出调试信息 PROD：生产模式，不输出调试信息

返回值类型：void

使用示例：

```
// 初始化不定义模式，默认不输出调试信息
configOptions.mode(ModeEnum.DEV);
```

调试信息：

```
I/SA.SensorsDataAPI: track event:
...
```

1.5 设置告警模式

说明：初始化 SDK 时，配置告警的模式，可在特有属性没有设置或者不全时拒绝发送数据并提供告警（`目前会检测的属性有：vin、aid、passwd、infoType`，设置这些属性的方法见：6.设置并获取特有属性）

方法：**alarm**

参数：

参数	说明
AlarmModeEnum alarmMode	告警的模式 LOG：日志告警模式，特有属性没有设置或者不全，则埋点数据不发送且提供日志告警 NONE：无告警模式，埋点数据都进行发送 --默认采用

返回值类型：void

使用示例：

```
// 初始化不定义告警模式，默认NONE数据都发送
configOptions.alarm(AlarmModeEnum.LOG);
```

调试信息：

```
I/SA.AnalyticsMessages: 车机vin或硬件版本缺失
```

1.6 获取预设属性

说明：可以通过此方法获取 App 端的预置属性传给服务端。预设属性通常是一些默认信息，比如网络状况，屏幕尺寸，系统版本等等。

方法：**getPresetProperties**

参数：无

返回值类型：JSONObject

使用示例：

```
JSONObject presetProperties = OASAPI.sharedInstance().getPresetProperties();
```

调试信息：

打印presetProperties

```
I/MainActivity: presetProperties:
    {
        "lib":"Android",
        "lib_version":"1.0.0",
        "manufacturer":"Google",
        "model":"Android SDK built for x86",
        "os":"Android",
        "os_version":"10",
        "screen_height":"1920",
        "screen_width":"1080",
        "wifi":"true",
        "network_type":"WIFI",
        "is_first_day":"false",
        "timezone_offset":"0"
    }
```

1.7 设置并获取特有属性

车机中必须设置的属性，如vin，提供了显式的方法，避免混淆或遗漏

① 设置vin、hardwareVersion、infoType

方法：**setVin / setHardwareVersion / setInfoType**

说明：setVin方法内部会对vin号进行加密

`2020/12/18 注：setInfoType() 方法将在下个版本中添加`

`2021/05/21 注：该方法仍未添加，因业务需求暂时搁置`

参数：

方法	参数	说明	是否必须
setVin	String vin	车机识别码	是
setHardwareVersion	String hardwareVersion	硬件版本	否
setInfoType	String infoType	车机版本	是

返回值类型：void

使用示例：

```
OASAPI.sharedInstance().setVin("A20");

OASAPI.sharedInstance().setHardwareVersion("v1.0");

OASAPI.sharedInstance().setInfoType("xx");
```

② 设置BasicAuth的账号和密码

方法：**setBasicAuth**

说明：根据文档《SGM-车联网数据工厂平台-API接口说明文档_v0.4.5》说明的接口鉴权认证，添加此 setBasicAuth 方法，`统一服务使用统一的应用账户和密码进行vdf接口的basic认证，但是帐号与数据模板权限需要在vdf平台申请`

参数：

参数	说明
String aid	应用id（App唯一识别码） 同时也是 BasicAuth 的账号
String passwd	BasicAuth 的密码

返回值类型：void

使用示例：

```
OASAPI.sharedInstance().setBasicAuth("aid", "passwd");
```

③ 加密解密方法

由于用户可能需要在设置某些属性前加密这些属性，SDK提供加密方法

注意：`在setVin方法中已默认使用这种加密方法对vin号进行加密了，故在setVin之前无需调用此方法加密vin`

加密方法：encryptData

参数	说明
String msg	待加密数据
String key（可选）	加密的密钥，不传时默认为空字符串

解密方法：decryptData

参数	说明
String msg	待解密数据
String key（可选）	解密的密钥，需要和加密时传入的密钥保持一致

使用示例：

```
// 假设用户在 setChannel 前要对 channel 进行加密
String channel = "channel";
String encodedChannel = OASAPI.sharedInstance().encryptData(channel, "密钥");
OASAPI.sharedInstance().setChannel(encodedChannel);

// 对 encodedChannel 解密
String decodedChannel = OASAPI.sharedInstance().decryptData(encodedChannel, "密钥");
```

④ 获取

方法：**getSuperProperties**

参数：无

返回值类型：JSONObject

使用示例：

```
JSONObject superProperties = OASAPI.sharedInstance().getSuperProperties();
if (superProperties.has("vin")) {
    try {
        String vin = superProperties.getString("vin");
        SALog.i("MainActivity", "vin:" + vin);
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
if (superProperties.has("hardwareVersion")) {
    try {
        String hardwareVersion = superProperties.getString("hardwareVersion");
        SALog.i("MainActivity", "hardwareVersion:" + hardwareVersion);
    } catch (JSONException e) {
        e.printStackTrace();
    }
}

// 获取 aid 和 passwd 同理...
```

打印设置后的vin及hardwareVersion

```
I/MainActivity:
    vin:H4sIAAAAAAAAAHM0MgAAOFqwRQMAAAA=
    hardwareVersion:v1.0
```

⑤ 告警

如果这些特有属性没有设置或者不全， 则埋点数据发送失败， 提供日志告警。配置详情见：3.设置告警模式

⑥设置用户ID

方法：**setUsrId**

参数：usrId

返回值类型：无

说明：由于项目规划加入公共字段 usr_id（idp_userid），故添加 setUsrId() 方法，用户可根据此方法设置idp_userid，最终产生的埋点信息将带上 usr_id 字段（对应编号为 G33）。此方法可多次调用，每次调用设置后此字段的值会改变，并将在后续埋点信息中携带，直到用户再次重新设置。

参数：

参数	说明
String usrId	设置 idp_userid

返回值类型：void

使用示例：

```
OASAPI.sharedInstance().setUsrId("THIS_IS_IDP_USERID");
```

调试信息：

```
{
    "value":{
        ...
        "d":{
            ...
            "G33":"THIS_IS_IDP_USERID",
            ...
        }
    }
}
```

1.8 埋点数据上报策略

① 修改与上次发送的时间间隔

说明：默认的与上次发送的时间间隔 ***flushInterval*** 为 **15 秒**，最小值 5 秒，若小于 5 秒，则按 5 秒处理。满足与上次发送的时间间隔大于该值，会向服务器上传数据。初始化 SDK 时，通过该方法，可修改 *flushInterval*默认值。

方法：**setFlushInterval**

参数：

参数	说明
int flushInterval	时间间隔，单位毫秒

返回值类型：SAConfigOptions

使用示例：

```
// 设置每 30 秒发送一次
configOptions.setFlushInterval(30000);
```

② 修改本地缓存日志数目

说明：默认的本地缓存日志数目 ***flushBulkSize*** 为**100 条**，由于上传接口对最小/大数据量有要求，故：最小值 50 条，若小于 50 条，则按 50 条处理；最大值为200条，若超过200条，则按200条处理。满足本地缓存日志数目大于该值，会向服务器上传数据。初始化 SDK 时，通过该方法，可修改*flushBulkSize*默认值。

方法：**setFlushBulkSize**

参数：

参数	说明
int flushBulkSize	缓存数目

返回值类型：SAConfigOptions

使用示例：

```
// 每缓存 50 条日志发送一次
configOptions.setFlushBulkSize(50);
```

③ 强制发送数据

说明：如果追求数据采集的时效性，可以调用该方法，强制将已经存储在本地的数据发送到服务端。

- 全部数据（推荐）

说明：创建任务放到事件队列中，执行到 flashSync() 时，任务队列中flashSync之前的埋点任务都已经入本地缓存库完成，所有事件数据能强制发送成功。

方法：**flushSync**

参数：无

返回值类型：void

使用示例：

```
// 记录用户登录事件
OASAPI.sharedInstance().track("UserLogin");
// 强制发送数据
OASAPI.sharedInstance().flushSync();
```

- 仅入库成功的数据

说明：在调用 flash() 之前的埋点的事件可能还在埋点采集的任务队列中，还没有入本地缓存库，该类事件数据无法强制上报。

方法：**flush**

参数：无

返回值类型：void

使用示例：

```
OASAPI.sharedInstance().track("UserLogin");
OASAPI.sharedInstance().flush();
```

④ 设置本地缓存上限

说明：默认的缓存上限是**32Mb**，最小 16MB，若小于 16MB，则按 16MB 处理。初始化 SDK 时，通过该方法，来设定缓存数据的上限值。

方法：**setMaxCacheSize**

参数：

参数	说明
long maxCacheSize	设置本地缓存上限值，单位 byte，默认为 32MB：32 * 1024 * 1024

返回值类型：SAConfigOptions

使用示例：

```
// 设置本地数据缓存上限值为16MB
configOptions.setMaxCacheSize(16 * 1024 * 1024);
```

⑤ 清空本地缓存

说明：通过该方法，删除 App 本地存储的所有事件。**如果不是特殊要求，请不要调用此方法。**

方法：**deleteAll**

参数：无

返回值类型：void

使用示例：

```
OASAPI.sharedInstance().deleteAll();
```

1.9 日志控制

说明：初始化 SDK 时，配置日志打印功能，DEV模式下日志默认开启，PROD模式下日志默认关闭。为了方便排查问题建议测试情况下，开启日志。

方法：**enableLog**

参数：

参数	说明
boolean enableLog	日志是否打印 true：开启日志打印 false：关闭日志打印

返回值类型：void

使用示例：

```
//日志开启
configOptions.enableLog(true);
```

1.10 其它说明

字段说明

SDK上传字段，整体应按照dcp模板设置字段进行上传。

SDK上传字段由**sdk预设字段+用户自定义字段** 组成。

预设字段

- 关于sdk默认预设字段说明见该文档结尾附件说明部分（附件三中预设属性说明）。
- 预设字段并非全部由sdk自动获取，部分需要集成应用进行上传。
- 预设字段不允许修改应根据要求上传。

自定义字段

- 自定义字段应符合创建要求，具体见**4附件**说明

字段值说明

- SDK中会对字段值进行限制，上传值应控制在**value.length<8191** 否则sdk将会对数据进行截取，上传后仅保留length<8191长度，目前针对如果上传格式为json字符串，截取后进行了补全json字符串格式操作（仅针对json截取后补全）。**一般情况下埋点单个字段值不会超过控制值，此外埋点SDK也不适用单字段大数据量采集上传。**
- SDK会对字段值进行处理行为，会替换相关转义字符 "","$ 。(目的是符合dcp上传格式要求)
- 上传字段值数据若为json值，注意转义问题。（可能导致上传结果显示bad data format)

上传链路说明

- 应用集成sdk，调用sdk相关接口采集埋点数据
- SDK将数据进行格式处理（符合dcp平台接口上传数据要求）
- SDK根据发送策略将数据上传到dcp平台

SDK仅为一个第三方包用于埋点，而非服务，整体是提供埋点方法，用户调用方法后SDK将埋点数据进行整合，调用dcp接口将数据上传给dcp。

开启日志SDK将会打印整合后发给dcp数据明细，强烈建议测试开启日志。

问题排查

详情具体请查看该文档结尾部分 **4附件、5注意事项、6问题排查及解决方案**

2 统一服务的方法

2.1 埋点方法

2.1.1 埋点方法 - 普通事件

说明：SDK 初始化后，可以通过 track() 方法追踪事件，可以为事件添加自定义属性

方法：**track**

type：track

参数：

参数	说明	示例
String eventName	追踪的事件名称（根据确认的规范填充）	MsgEvent
String sid	数据模板ID	***
String svid	数据模板版本ID	***
String appId	App的标识，请直接从APP获取后填充，避免版本迭代时字段未更新等问题。	com.xx.xxApp
String appVersion	App版本号，请直接从APP获取后填充，避免版本迭代时字段未更新等问题。	1.1.1
String curScreen	当前页面名称（activity 对应的名称）	首页
String curActivity	当前 activity 名称 获取方式：`this.getClass().getName()`	com.xx.xxApp.XxActivity
String properties（可选）	包含自定义的json字符串 用户自定义事件的数据类型会被转为String 注意：`自定义属性的key以编号的形式写入，key必须满足：①第一位是字母G，②第二至四位是 >=100 且 <=999 的整数` 详细说明见：附3：事件传输说明	new JSONObject() .put("G222", "123456") .put("G333", "Hello") .toString()

返回值类型：void

使用示例：

```
try {
    JSONObject properties = new JSONObject();
    properties.put("G222", "123456");    // 设置信息ID
    properties.put("G333", "Hello!");    // 设置信息内容
```

```
    OASAPI.sharedInstance().track(
        "MsgEvent",
        "sid",
        "svid",
        "appId",
        "appVersion",
        "curScreen",
        this.getClass().getName(),
        properties.toString()
    );
} catch (JSONException e) {
    e.printStackTrace();
}
```

调试信息:

```
{
    "records":[
        {
            "value":{
                "sid":"sid",
                "svid":"svid",
                "did":"H4sIAAAAAAAAAHM0MgAAOFqwRQMAAAA=",
                "aid":"aid",
                "d":{
                    "lib":"Android",
                    "os_version":"10",
                    "lib_version":"1.0.0",
                    "model":"Android SDK built for x86",
                    "timezone_offset":"0",
                    "os":"Android",
                    "screen_width":"1080",
                    "screen_height":"1920",
                    "manufacturer":"Google",
                    "wifi":"true",
                    "network_type":"WIFI",
                    "G222":"123456",    // 这里是用户添加的自定义事件
                    "G333":"Hello!",    // 这里是用户添加的自定义事件
                    "app_version":"app_version_##",
                    "cur_screen":"首页",
                    "track_id":"-1615850241",
                    "time":"1604285487457",
                    "type":"track",  // 普通事件的type为 track
                    "event":"MsgEvent",
                    "is_first_day":"false",
                    "lib_method":"code",

 "lib_detail":"com.bigtimes.sdk.BigtimesDataAPI##trackEvent##BigtimesDataAPI.java
##3761",
                    "hardwareVersion":"v1.0",
                    "channel":"channel",
                    "flush_time":"1604285502610"
                }
            }
        }
```

```
    ]
}
```

2.1.2 页面方法 - 进入

说明：SDK 初始化后，此方法用于进入某页面时调用，可追踪到进入页面的事件；进入事件中可以选择添加自定义属性

方法：**trackEnterPage**

type：enter_page

参数：

> 注意：进入页面不支持传入名称，名称默认为 `EnterPage`

参数	说明	示例
String sid	数据模板ID	***
String svid	数据模板版本ID	***
String appId	App的标识，请直接从APP获取后填充，避免版本迭代时字段未更新等问题。	com.xx.xxApp
String appVersion	App版本号，请直接从APP获取后填充，避免版本迭代时字段未更新等问题。	1.1.1
String curScreen	当前页面名称（activity 对应的名称）	首页
String curActivity	当前 activity 名称 获取方式：`this.getClass().getName()`	com.xx.xxApp.XxActivity
String properties（可选）	包含自定义的json字符串 用户自定义事件的数据类型会被转为String 注意：`自定义属性的key以编号的形式写入，key必须满足：①第一位是字母G，②第二至四位是 >=100 且 <=999 的整数` 详细说明见：附3：事件传输说明	new JSONObject() .put("G222", "123456") .put("G333", "Hello") .toString()

返回值类型：void

使用示例：

```
try {
    JSONObject properties = new JSONObject();
    properties.put("G222", "123456");    // 设置信息ID
    properties.put("G333", "Hello!");    // 设置信息内容

    OASAPI.sharedInstance().trackEnterPage(
        "sid",
        "svid",
        "app_id",
        "app_version",
        "首页",
```

```
        this.getClass().getName(),
        properties.toString()
    );
} catch (JSONException e) {
    e.printStackTrace();
}
```

调试信息：

```
{
    "records":[
        {
            "value":{
                "sid":"sid",
                "svid":"svid",
                "did":"H4sIAAAAAAAAAHM0MgAAOFqwRQMAAAA=",
                "aid":"aid",
                "d":{
                    "lib":"Android",
                    "os_version":"10",
                    "lib_version":"1.0.0",
                    "model":"Android SDK built for x86",
                    "timezone_offset":"0",
                    "os":"Android",
                    "screen_width":"1080",
                    "screen_height":"1920",
                    "manufacturer":"Google",
                    "wifi":"true",
                    "network_type":"WIFI",
                    "G222":"123456",    // 这里是用户添加的自定义事件
                    "G333":"Hello",     // 这里是用户添加的自定义事件
                    "app_version":"app_version_##",
                    "cur_screen":"首页",
                    "track_id":"-1615850241",
                    "time":"1604285487457",
                    "type":"enter_page",  // 进入页面的type为 enter_page
                    "event":"EnterPage",
                    "is_first_day":"false",
                    "lib_method":"code",

 "lib_detail":"com.bigtimes.sdk.BigtimesDataAPI##trackEvent##BigtimesDataAPI.java
##3761",
                    "hardwareVersion":"v1.0",
                    "channel":"channel",
                    "flush_time":"1604285502610"
                }
            }
        }
    ]
}
```

2.1.3 页面方法 - 退出

说明：SDK 初始化后，此方法用于关闭页面时调用，可追踪到退出页面的事件；退出事件中可以选择添加自定义属性

方法：**trackExitPage**

type：exit_page

参数：

> 注意：退出页面不支持传入名称，名称默认为 `ExitPage`

参数	说明	示例
String sid	数据模板ID	***
String svid	数据模板版本ID	***
String appId	App的标识，请直接从APP获取后填充，避免版本迭代时字段未更新等问题。	com.xx.xxApp
String appVersion	App版本号，请直接从APP获取后填充，避免版本迭代时字段未更新等问题。	1.1.1
String curScreen	当前页面名称（activity 对应的名称）	首页
String curActivity	当前 activity 名称 获取方式：`this.getClass().getName()`	com.xx.xxApp.XxActivity
String properties（可选）	包含自定义的json字符串 用户自定义事件的数据类型会被转为String 注意：`自定义属性的key以编号的形式写入，key必须满足：①第一位是字母G，②第二至四位是 >=100 且 <=999 的整数` 详细说明见：附3：事件传输说明	new JSONObject() .put("G222", "123456") .put("G333", "Hello") .toString()

返回值类型：void

使用示例：

```
try {
    JSONObject properties = new JSONObject();
    properties.put("G222", "123456");    // 设置信息ID
    properties.put("G333", "Hello!");    // 设置信息内容

    OASAPI.sharedInstance().trackExitPage(
        "sid",
        "svid",
        "app_id",
        "app_version",
        "首页",
        this.getClass().getName(),
        properties.toString()
    );
```

```
} catch (JSONException e) {
    e.printStackTrace();
}
```

调试信息：

```
{
    "records":[
        {
            "value":{
                "sid":"sid",
                "svid":"svid",
                "did":"H4sIAAAAAAAAAHM0MgAAOFqwRQMAAAA=",
                "aid":"aid",
                "d":{
                    "lib":"Android",
                    "os_version":"10",
                    "lib_version":"1.0.0",
                    "model":"Android SDK built for x86",
                    "timezone_offset":"0",
                    "os":"Android",
                    "screen_width":"1080",
                    "screen_height":"1920",
                    "manufacturer":"Google",
                    "wifi":"true",
                    "network_type":"WIFI",
                    "G222":"123456",     // 这里是用户添加的自定义事件
                    "G333":"Hello",      // 这里是用户添加的自定义事件
                    "app_version":"app_version_##",
                    "cur_screen":"首页",
                    "track_id":"-1615850241",
                    "time":"1604285487457",
                    "type":"exit_page",  // 退出页面的type为 exit_page
                    "event":"ExitPage",
                    "is_first_day":"false",
                    "lib_method":"code",

 "lib_detail":"com.bigtimes.sdk.BigtimesDataAPI##trackEvent##BigtimesDataAPI.java
##3761",
                    "hardwareVersion":"v1.0",
                    "channel":"channel",
                    "flush_time":"1604285502610"
                }
            }
        }
    ]
}
```

2.1.4 页面方法 - 跳转

说明：SDK 初始化后，在点击响应事件里的页面跳转过程中，可以通过 trackJump() 方法追踪页面跳转事件；跳转事件中可以选择添加自定义属性

方法：**trackJumpPage**

type：jump_page

参数：

> 注意：跳转页面不支持传入名称，名称默认为`JumpPage`

参数	说明	示例
String sid	数据模板ID	***
String svid	数据模板版本ID	***
String appId	App的标识，请直接从APP获取后填充，避免版本迭代时字段未更新等问题。	com.xx.xxApp
String appVersion	App版本号，请直接从APP获取后填充，避免版本迭代时字段未更新等问题。	1.1.1
String curScreen	当前页面名称（activity 对应的名称）	首页
String curActivity	当前 activity 名称 获取方式：`this.getClass().getName()`	com.xx.xxApp.XxActivity
String tarScreen	目标页面名称（activity 对应的名称） `注意：若隐式跳转不知道目标页面，这里填IMPLICIT_TAR_SCREEN`	用户中心
String tarActivity	目标 activity 名称 获取方式：`NextActivity.class.getName()` `注意：若隐式跳转不知道目标activity，这里填IMPLICIT_TAR_ACTIVITY`	com.xx.yyApp.YyActivity
String properties（可选）	包含自定义的json字符串 用户自定义事件的数据类型会被转为String 注意：`自定义属性的key以编号的形式写入，key必须满足：①第一位是字母G，②第二至四位是 >=100 且 <=999 的整数` 详细说明见：附3：事件传输说明	new JSONObject() .put("G222", "123456") .put("G333", "Hello") .toString()

返回值类型：void

使用示例：

```
findViewById(com.bigtimes.analytics.android.demo.R.id.mainButton).setOnClickListe
ner(v -> {

    Intent intent = new Intent(NextActivity.this, MainActivity.class);

    try {
        JSONObject properties = new JSONObject();
        properties.put("G111", "#000001");

        OASAPI.sharedInstance().trackJumpPage(
            "sid",
```

```
            "svid",
            "app_id",
            "app_version",
            "首页2",
            this.getClass().getName(),
            "首页1",
            MainActivity.class.getName(),
            properties.toString()
        );
    } catch (JSONException e) {
        e.printStackTrace();
    }
    startActivity(intent);
});
```

调试信息:

```
{
    "records":[
        {
            "value":{
                "sid":"sid",
                "svid":"svid",
                "did":"H4sIAAAAAAAAAHM0MgAAOFqwRQMAAAA=",
                "aid":"aid",
                "d":{
                    "lib":"Android",
                    "os_version":"10",
                    "lib_version":"1.0.0",
                    "model":"Android SDK built for x86",
                    "timezone_offset":"0",
                    "os":"Android",
                    "screen_width":"1080",
                    "screen_height":"1920",
                    "manufacturer":"Google",
                    "wifi":"true",
                    "network_type":"WIFI",
                    "app_version":"app_version_##",
                    "cur_screen":"首页",    // 当前页面
                    "tar_screen":"用户中心", // 目标页面
                    "track_id":"-853912013",
                    "time":"1604285943144",
                    "type":"jump_page",  // 跳转页面的type为 jump_page
                    "event":"JumpPage",
                    "is_first_day":"false",
                    "lib_method":"code",

"lib_detail":"com.bigtimes.sdk.BigtimesDataAPI##trackEvent##BigtimesDataAPI.java#
#3761",
                    "hardwareVersion":"v1.0",
                    "channel":"channel",
                    "flush_time":"1604285958307"
                }
            }
        }
```

```
    ]
}
```

2.1.5 Fragment方法 - 进入

说明：SDK 初始化后，此方法用于进入某 Fragment 时调用，可追踪到进入 Fragment 的动作；可以选择添加自定义属性

方法：**trackEnterFragment**（type 为 EnterFragment）

type：enter_fragment

参数：

> 注意：进入Fragment不支持传入名称，名称默认为 `EnterFragment`

参数	说明	示例
String sid	数据模板ID	***
String svid	数据模板版本ID	***
String appId	App的标识，请直接从APP获取后填充，避免版本迭代时字段未更新等问题。	com.xx.xxApp
String appVersion	App版本号，请直接从APP获取后填充，避免版本迭代时字段未更新等问题。	1.1.1
String curScreen	当前页面名称（activity 对应的名称）	首页
String curActivity	当前 activity 名称 获取方式：`this.getClass().getName()`	com.xx.xxApp.XxActivity
String fragmentName	进入的 Fragment 的页面名称	详情页1
String properties（可选）	包含自定义的json字符串 用户自定义事件的数据类型会被转为String 注意：`自定义属性的key以编号的形式写入，key必须满足：①第一位是字母G，②第二至四位是>=100 且 <=999 的整数` 详细说明见：附3：事件传输说明	new JSONObject() .put("G222", "123456") .put("G333", "Hello") .toString()

返回值类型：void

使用示例：

```
try {
    JSONObject properties = new JSONObject();
    properties.put("G222", "123456");    // 设置信息ID
    properties.put("G333", "Hello!");    // 设置信息内容

    OASAPI.sharedInstance().trackEnterFragment(   "sid",
        "svid",
        "app_id",
        "app_version",
```

```
            "首页",
            this.getClass().getName(),
            "Fragment01",
            properties.toString()
        );
    } catch (JSONException e) {
        e.printStackTrace();
    }
```

调试信息：

```
...
```

2.1.6 Fragment方法 - 退出

说明：SDK 初始化后，此方法用于退出 Fragment 时调用，可追踪到退出 Fragment 的动作；可以选择添加自定义属性

方法：**trackExitFragment**

type：exit_fragment

参数：

> 注意：退出Fragment不支持传入名称，名称默认为 `ExitFragment`

参数	说明	示例
String sid	数据模板ID	***
String svid	数据模板版本ID	***
String appId	App的标识，请直接从APP获取后填充，避免版本迭代时字段未更新等问题。	com.xx.xxApp
String appVersion	App版本号，请直接从APP获取后填充，避免版本迭代时字段未更新等问题。	1.1.1
String curScreen	当前页面名称（activity 对应的名称）	首页
String curActivity	当前 activity 名称 获取方式：`this.getClass().getName()`	com.xx.xxApp.XxActivity
String fragmentName	当前 Fragment 的页面名称	详情页1
String properties（可选）	包含自定义的json字符串 用户自定义事件的数据类型会被转为String 注意：`自定义属性的key以编号的形式写入，key必须满足：①第一位是字母G，②第二至四位是>=100 且 <=999 的整数` 详细说明见：附3：事件传输说明	new JSONObject() .put("G222", "123456") .put("G333", "Hello") .toString()

返回值类型：void

使用示例：

```
try {
    JSONObject properties = new JSONObject();
    properties.put("G222", "123456");    // 设置信息ID
    properties.put("G333", "Hello!");    // 设置信息内容

    OASAPI.sharedInstance().trackExitFragment(
        "sid",
        "svid",
        "app_id",
        "app_version",
        "首页",
        this.getClass().getName(),
        "Fragment01",
        properties.toString()
    );
} catch (JSONException e) {
    e.printStackTrace();
}
```

调试信息:

```
...
```

2.1.7 Fragment方法 - 跳转

说明：SDK 初始化后，在 Fragment 跳转过程中，可以通过此方法追踪 Fragment 跳转的动作；可以选择添加自定义属性

方法：**trackJumpFragment**（type 为 JumpFragment）

type：jump_fragment

参数:

> 注意：跳转Fragment不支持传入名称，名称默认为 `JumpFragment`

参数	说明	示例
String sid	数据模板ID	***
String svid	数据模板版本ID	***
String appId	App的标识，请直接从APP获取后填充，避免版本迭代时字段未更新等问题。	com.xx.xxApp
String appVersion	App版本号，请直接从APP获取后填充，避免版本迭代时字段未更新等问题。	1.1.1
String curScreen	当前页面名称（activity 对应的名称）	首页
String curActivity	当前 activity 名称 获取方式：`this.getClass().getName()`	com.xx.xxApp.XxActivity
String fragmenName	当前 Fragment 的页面名称	详情页1

参数	说明	示例
String tarFragmentName	目标 Fragment 的页面名称	详情页2
String properties（可选）	包含自定义的json字符串 用户自定义事件的数据类型会被转为String 注意：`自定义属性的key以编号的形式写入，key必须满足：①第一位是字母G，②第二至四位是 >=100 且 <=999 的整数` 详细说明见：附3：事件传输说明	new JSONObject() .put("G222", "123456") .put("G333", "Hello") .toString()

返回值类型：void

使用示例：

```
try {
    JSONObject properties = new JSONObject();
    properties.put("G222", "123456");    // 设置信息ID
    properties.put("G333", "Hello!");    // 设置信息内容

    OASAPI.sharedInstance().trackJumpFragment(
        "sid",
        "svid",
        "app_id",
        "app_version",
        "首页",
        this.getClass().getName(),
        "Fragment01",
        "Fragment02",
        properties.toString()
    );
} catch (JSONException e) {
    e.printStackTrace();
}
```

调试信息：

```
...
```

2.1.8 App方法 - 跳转

说明：SDK 初始化后，在某些拉起其它App的地方可调用此方法收集埋点数据；可以选择添加自定义属性

方法：**trackJumpApp**（type 为 JumpApp）

type：jump_app

参数：

> 注意：跳转App不支持传入名称，名称默认为 `JumpApp`

参数	说明	示例
String sid	数据模板ID	***
String svid	数据模板版本ID	***
String appId	当前App的标识，请直接从APP获取后填充，避免版本迭代时字段未更新等问题。	com.xx.xxApp
String appVersion	当前App版本号，请直接从APP获取后填充，避免版本迭代时字段未更新等问题。	1.1.1
String curScreen	当前页面名称（activity 对应的名称）	首页
String curActivity	当前 activity 名称 获取方式：`this.getClass().getName()`	com.xx.xxApp.XxActivity
String tarApp	唤起的目标App 用包名表示	com.xx.tarApp
String tarScreen	要跳转的目标App的目标 页面名称 注意：若隐式跳转不知道目标页面，这里填`IMPLICIT_TAR_SCREEN`	促销
String tarActivity	要跳转的目标App的目标 activity 注意：若隐式跳转不知道目标`activity`，这里填`IMPLICIT_TAR_ACTIVITY`	com.xx.tarApp.Sale
String properties（可选）	包含自定义的json字符串 用户自定义事件的数据类型会被转为String 注意：自定义属性的`key`以编号的形式写入，`key`必须满足：①第一位是字母`G`，②第二至四位是 `>=100` 且 `<=999` 的整数 详细说明见：附3：事件传输说明	new JSONObject() .put("G222", "123456") .put("G333", "Hello") .toString()

返回值类型：void

使用示例：

```
try {
    JSONObject properties = new JSONObject();
    properties.put("G222", "123456");    // 设置信息ID
    properties.put("G333", "Hello!");    // 设置信息内容

    OASAPI.sharedInstance().trackJumpApp(
        "sid",
        "svid",
        "app_id",
        "app_version",
        "首页",
        this.getClass().getName(),
        "com.xx.tarApp",
        "促销",
        "com.xx.tarApp.Sale",
```

```
        properties.toString()
    );
} catch (JSONException e) {
    e.printStackTrace();
}
```

调试信息：

```
...
```

2.2 数据上报方法

2.2.1 利用track方法上报项目相关信息

说明：对于某些非页面埋点的场景，比如只进行数据上传而不关心页面、控件等无关信息，可以利用track方法的自定义字段携带要上传的信息，并将无关字段设置为空字符串""（此方法实际上就是 2.1.1 的 track 方法，只是使用时侧重点不同）

方法：**track**

type：track

参数：

参数	说明	示例
String eventName	事件名称（用户自行设置）	MsgEvent
String sid	数据模板ID	***
String svid	数据模板版本ID	***
String appId	不关心的信息，用户传入空字符串	""
String appVersion	不关心的信息，用户传入空字符串	""
String curScreen	不关心的信息，用户传入空字符串	""
String curActivity	不关心的信息，用户传入空字符串	""
String properties（可选）	包含自定义的json字符串 用户自定义事件的数据类型会被转为String 注意：自定义属性的key以编号的形式写入，key必须满足：①第一位是字母G，②第二至四位是 >=100 且 <=999 的整数 详细说明见：附3：事件传输说明	new JSONObject() .put("G222", "123456") .put("G333", "Hello") .toString()

返回值类型：void

使用示例：

```
try {
```

```
    JSONObject properties = new JSONObject();
    properties.put("G222", "123456");    // 设置信息ID
    properties.put("G333", "Hello!");    // 设置信息内容

    OASAPI.sharedInstance().track(
        "MsgEvent",
        "sid",
        "svid",
        "",
        "",
        "",
        "",
        properties.toString()
    );
} catch (JSONException e) {
    e.printStackTrace();
}
```

调试信息：

```
...
```

3 独立集成的方法

3.1 埋点方法

3.1.1 埋点方法 - 普通事件

说明：SDK 初始化后，可以通过 track() 方法追踪事件，可以为事件添加自定义属性

方法：**track**

type：track

参数：

参数	说明	示例
String eventName	追踪的事件名称（根据确认的规范填充）	MsgEvent
String sid	数据模板ID	***
String svid	数据模板版本ID	***
String properties（可选）	包含自定义的json字符串 用户自定义事件的数据类型会被转为String 注意：`自定义属性的key以编号的形式写入，key必须满足：①第一位是字母G，②第二至四位是 >=100 且 <=999 的整数` 详细说明见：附3：事件传输说明	new JSONObject() .put("G222", "123456") .put("G333", "Hello") .toString()

返回值类型：void

使用示例：

```
try {
    JSONObject properties = new JSONObject();
    properties.put("G222", "123456");    // 设置信息ID
    properties.put("G333", "Hello!");    // 设置信息内容

    OASAPI.sharedInstance().track(
        "MsgEvent",
        "sid",
        "svid",
        properties.toString()
    );
} catch (JSONException e) {
    e.printStackTrace();
}
```

调试信息：

```
{
    "records":[
        {
            "value":{
                "sid":"sid",
                "svid":"svid",
                "did":"H4sIAAAAAAAAAHM0MgAAOFqwRQMAAAA=",
                "aid":"aid",
                "d":{
                    "lib":"Android",
                    "os_version":"10",
                    "lib_version":"1.0.0",
                    "model":"Android SDK built for x86",
                    "timezone_offset":"0",
                    "os":"Android",
                    "screen_width":"1080",
                    "screen_height":"1920",
                    "manufacturer":"Google",
                    "wifi":"true",
                    "network_type":"WIFI",
                    "G222":"123456",    // 这里是用户添加的自定义事件
                    "G333":"Hello!",    // 这里是用户添加的自定义事件
                    "app_version":"app_version_##",
                    "cur_screen":"首页",
                    "track_id":"-1615850241",
                    "time":"1604285487457",
                    "type":"track",  // 普通事件的type为 track
                    "event":"MsgEvent",
                    "is_first_day":"false",
                    "lib_method":"code",

 "lib_detail":"com.bigtimes.sdk.BigtimesDataAPI##trackEvent##BigtimesDataAPI.java
##3761",
                    "hardwareVersion":"v1.0",
                    "channel":"channel",
                    "flush_time":"1604285502610"
```

```
                }
            }
        }
    ]
}
```

3.1.2 页面方法 - 进入

说明：SDK 初始化后，此方法用于进入某页面时调用，可追踪到进入页面的事件；进入事件中可以选择添加自定义属性

方法：**trackEnterPage**

type：enter_page

参数：

> 注意：进入页面不支持传入名称，名称默认为 `EnterPage`

参数	说明	示例
String sid	数据模板ID	***
String svid	数据模板版本ID	***
String properties（可选）	包含自定义的json字符串 用户自定义事件的数据类型会被转为String 注意：`自定义属性的key以编号的形式写入，key必须满足：①第一位是字母G，②第二至四位是 >=100 且 <=999 的整数` 详细说明见：附3：事件传输说明	new JSONObject() .put("G222", "123456") .put("G333", "Hello") .toString()

返回值类型：void

使用示例：

```
try {
    JSONObject properties = new JSONObject();
    properties.put("G222", "123456");    // 设置信息ID
    properties.put("G333", "Hello!");    // 设置信息内容

    OASAPI.sharedInstance().trackEnterPage(
        "sid",
        "svid",
        properties.toString()
    );
} catch (JSONException e) {
    e.printStackTrace();
}
```

调试信息：

```
{
    "records":[
```

```
        {
            "value":{
                "sid":"sid",
                "svid":"svid",
                "did":"H4sIAAAAAAAAAHM0MgAAOFqwRQMAAAA=",
                "aid":"aid",
                "d":{
                    "lib":"Android",
                    "os_version":"10",
                    "lib_version":"1.0.0",
                    "model":"Android SDK built for x86",
                    "timezone_offset":"0",
                    "os":"Android",
                    "screen_width":"1080",
                    "screen_height":"1920",
                    "manufacturer":"Google",
                    "wifi":"true",
                    "network_type":"WIFI",
                    "G222":"123456",     // 这里是用户添加的自定义事件
                    "G333":"Hello",      // 这里是用户添加的自定义事件
                    "app_version":"app_version_##",
                    "cur_screen":"首页",
                    "track_id":"-1615850241",
                    "time":"1604285487457",
                    "type":"enter_page",  // 进入页面的type为 enter_page
                    "event":"EnterPage",
                    "is_first_day":"false",
                    "lib_method":"code",

 "lib_detail":"com.bigtimes.sdk.BigtimesDataAPI##trackEvent##BigtimesDataAPI.java
##3761",
                    "hardwareVersion":"v1.0",
                    "channel":"channel",
                    "flush_time":"1604285502610"
                }
            }
        }
    ]
}
```

3.1.3 页面方法 - 退出

说明：SDK 初始化后，此方法用于关闭页面时调用，可追踪到退出页面的事件；退出事件中可以选择添加自定义属性

方法：**trackExitPage**

type：exit_page

参数：

> 注意：退出页面不支持传入名称，名称默认为 `ExitPage`

参数	说明	示例
String sid	数据模板ID	***

参数	说明	示例
String svid	数据模板版本ID	***
String properties（可选）	包含自定义的json字符串 用户自定义事件的数据类型会被转为String 注意：`自定义属性的key以编号的形式写入，key必须满足：①第一位是字母G，②第二至四位是 >=100 且 <=999 的整数` 详细说明见：附3：事件传输说明	new JSONObject() .put("G222", "123456") .put("G333", "Hello") .toString()

返回值类型：void

使用示例：

```
try {
    JSONObject properties = new JSONObject();
    properties.put("G222", "123456");    // 设置信息ID
    properties.put("G333", "Hello!");    // 设置信息内容

    OASAPI.sharedInstance().trackExitPage(
        "sid",
        "svid",
        properties.toString()
    );
} catch (JSONException e) {
    e.printStackTrace();
}
```

调试信息：

```
{
    "records":[
        {
            "value":{
                "sid":"sid",
                "svid":"svid",
                "did":"H4sIAAAAAAAAAHM0MgAAOFqwRQMAAAA=",
                "aid":"aid",
                "d":{
                    "lib":"Android",
                    "os_version":"10",
                    "lib_version":"1.0.0",
                    "model":"Android SDK built for x86",
                    "timezone_offset":"0",
                    "os":"Android",
                    "screen_width":"1080",
                    "screen_height":"1920",
                    "manufacturer":"Google",
                    "wifi":"true",
                    "network_type":"WIFI",
                    "G222":"123456",    // 这里是用户添加的自定义事件
                    "G333":"Hello",     // 这里是用户添加的自定义事件
```

```
                    "app_version":"app_version_##",
                    "cur_screen":"首页",
                    "track_id":"-1615850241",
                    "time":"1604285487457",
                    "type":"exit_page",  // 退出页面的type为 exit_page
                    "event":"ExitPage",
                    "is_first_day":"false",
                    "lib_method":"code",

 "lib_detail":"com.bigtimes.sdk.BigtimesDataAPI##trackEvent##BigtimesDataAPI.java
##3761",
                    "hardwareVersion":"v1.0",
                    "channel":"channel",
                    "flush_time":"1604285502610"
                }
            }
        }
    ]
}
```

3.1.4 页面方法 - 跳转

说明：SDK 初始化后，在点击响应事件里的页面跳转过程中，可以通过 trackJump() 方法追踪页面跳转事件；跳转事件中可以选择添加自定义属性

方法：**trackJumpPage**

type：jump_page

参数：

> 注意：跳转页面不支持传入名称，名称默认为`JumpPage`

参数	说明	示例
String sid	数据模板ID	***
String svid	数据模板版本ID	***
String tarScreen	目标页面名称（activity 对应的名称） 注意：若隐式跳转不知道目标页面，这里填`IMPLICIT_TAR_SCREEN`	用户中心
String tarActivity	目标 activity 名称 获取方式：`NextActivity.class.getName()` 注意：若隐式跳转不知道目标`activity`，这里填`IMPLICIT_TAR_ACTIVITY`	com.xx.yyApp.YyActivity
String properties（可选）	包含自定义的json字符串 用户自定义事件的数据类型会被转为String 注意：`自定义属性的key以编号的形式写入，key必须满足：①第一位是字母G，②第二至四位是 >=100 且 <=999 的整数` 详细说明见：附3：事件传输说明	new JSONObject() .put("G222", "123456") .put("G333", "Hello") .toString()

返回值类型：void

使用示例：

```
findViewById(com.bigtimes.analytics.android.demo.R.id.mainButton).setOnClickListe
ner(v -> {

    Intent intent = new Intent(NextActivity.this, MainActivity.class);

    try {
        JSONObject properties = new JSONObject();
        properties.put("G111", "#000001");

        OASAPI.sharedInstance().trackJumpPage(
            "sid",
            "svid",
            "首页1",
            MainActivity.class.getName(),
            properties.toString()
        );
    } catch (JSONException e) {
        e.printStackTrace();
    }
    startActivity(intent);
});
```

调试信息：

```
{
    "records":[
        {
            "value":{
                "sid":"sid",
                "svid":"svid",
                "did":"H4sIAAAAAAAAAHM0MgAAOFqwRQMAAAA=",
                "aid":"aid",
                "d":{
                    "lib":"Android",
                    "os_version":"10",
                    "lib_version":"1.0.0",
                    "model":"Android SDK built for x86",
                    "timezone_offset":"0",
                    "os":"Android",
                    "screen_width":"1080",
                    "screen_height":"1920",
                    "manufacturer":"Google",
                    "wifi":"true",
                    "network_type":"WIFI",
                    "app_version":"app_version_##",
                    "cur_screen":"首页",    // 当前页面
                    "tar_screen":"用户中心", // 目标页面
                    "track_id":"-853912013",
                    "time":"1604285943144",
                    "type":"jump_page",  // 跳转页面的type为 jump_page
                    "event":"JumpPage",
```

```
                    "is_first_day":"false",
                    "lib_method":"code",

"lib_detail":"com.bigtimes.sdk.BigtimesDataAPI##trackEvent##BigtimesDataAPI.java#
#3761",
                    "hardwareVersion":"v1.0",
                    "channel":"channel",
                    "flush_time":"1604285958307"
                }
            }
        }
    ]
}
```

3.1.5 Fragment方法 - 进入

说明：SDK 初始化后，此方法用于进入某 Fragment 时调用，可追踪到进入 Fragment 的动作；可以选择添加自定义属性

方法：**trackEnterFragment**（type 为 EnterFragment）

type：enter_fragment

参数：

> 注意：进入Fragment不支持传入名称，名称默认为 `EnterFragment`

参数	说明	示例
String sid	数据模板ID	***
String svid	数据模板版本ID	***
~~String fragmentName~~	进入的 Fragment 的页面名称（待验证：目前尚不确定自动获取到的是Fragmen的 包.类名 还是 文字名）	详情页1
String properties（可选）	包含自定义的json字符串 用户自定义事件的数据类型会被转为String 注意：自定义属性的key以编号的形式写入，key必须满足：①第一位是字母G，②第二至四位是 >=100 且 <=999 的整数 详细说明见：附3：事件传输说明	new JSONObject() .put("G222", "123456") .put("G333", "Hello") .toString()

返回值类型：void

使用示例：

```
try {
    JSONObject properties = new JSONObject();
    properties.put("G222", "123456");    // 设置信息ID
    properties.put("G333", "Hello!");    // 设置信息内容

    OASAPI.sharedInstance().trackEnterFragment(   "sid_##",
        "svid",
        "Fragment01",
        properties.toString()
    );
} catch (JSONException e) {
    e.printStackTrace();
}
```

调试信息：

```
...
```

3.1.6 Fragment方法 - 退出

说明：SDK 初始化后，此方法用于退出 Fragment 时调用，可追踪到退出 Fragment 的动作；可以选择添加自定义属性

方法：**trackExitFragment**

type：exit_fragment

参数：

> 注意：退出Fragment不支持传入名称，名称默认为 `ExitFragment`

参数	说明	示例
String sid	数据模板ID	***
String svid	数据模板版本ID	***
~~String fragmentName~~	当前 Fragment 的页面名称（待验证：目前尚不确定自动获取到的是Fragmen的 包.类名 还是 文字名）	详情页1
String properties（可选）	包含自定义的json字符串 用户自定义事件的数据类型会被转为String 注意：自定义属性的key以编号的形式写入，key必须满足：①第一位是字母G，②第二至四位是 >=100 且 <=999 的整数 详细说明见：附3：事件传输说明	new JSONObject() .put("G222", "123456") .put("G333", "Hello") .toString()

返回值类型：void

使用示例：

```
try {
    JSONObject properties = new JSONObject();
    properties.put("G222", "123456");    // 设置信息ID
```

```
    properties.put("G333", "Hello!");    // 设置信息内容

    OASAPI.sharedInstance().trackExitFragment(
        "sid",
        "svid",
        "Fragment01",
        properties.toString()
    );
} catch (JSONException e) {
    e.printStackTrace();
}
```

调试信息：

```
...
```

3.1.7 Fragment方法 - 跳转

说明：SDK 初始化后，在 Fragment 跳转过程中，可以通过此方法追踪 Fragment 跳转的动作；可以选择添加自定义属性

方法：**trackJumpFragment**（type 为 JumpFragment）

type：jump_fragment

参数：

> 注意：跳转Fragment不支持传入名称，名称默认为 `JumpFragment`

参数	说明	示例
String sid	数据模板ID	***
String svid	数据模板版本ID	***
~~String fragmenName~~	当前 Fragment 的页面名称（待验证：目前尚不确定自动获取到的是Fragmen的 包.类名 还是 文字名）	详情页1
String tarFragmentName	目标 Fragment 的页面名称	详情页2
String properties（可选）	包含自定义的json字符串 用户自定义事件的数据类型会被转为String 注意：自定义属性的key以编号的形式写入，key必须满足：①第一位是字母G，②第二至四位是 >=100 且 <=999 的整数 详细说明见：附3：事件传输说明	new JSONObject() .put("G222", "123456") .put("G333", "Hello") .toString()

返回值类型：void

使用示例：

```
try {
```

```
    JSONObject properties = new JSONObject();
    properties.put("G222", "123456");    // 设置信息ID
    properties.put("G333", "Hello!");    // 设置信息内容

    OASAPI.sharedInstance().trackJumpFragment(
        "sid",
        "svid",
        "Fragment01",
        "Fragment02",
        properties.toString()
    );
} catch (JSONException e) {
    e.printStackTrace();
}
```

调试信息:

```
...
```

3.1.8 App方法 - 跳转

说明：SDK 初始化后，在某些拉起其它App的地方可调用此方法收集埋点数据；可以选择添加自定义属性

方法：**trackJumpApp**（type 为 JumpApp）

type：jump_app

参数:

> 注意：跳转App不支持传入名称，名称默认为 `JumpApp`

参数	说明	示例
String sid	数据模板ID	***
String svid	数据模板版本ID	***
String tarApp	唤起的目标App 用包名表示	com.xx.tarApp
String tarScreen	要跳转的目标App的目标 页面名称 注意：若隐式跳转不知道目标页面，这里填 `IMPLICIT_TAR_SCREEN`	促销
String tarActivity	要跳转的目标App的目标 activity 注意：若隐式跳转不知道目标`activity`，这里填 `IMPLICIT_TAR_ACTIVITY`	com.xx.tarApp.Sale
String properties（可选）	包含自定义的json字符串 用户自定义事件的数据类型会被转为String 注意：`自定义属性的key以编号的形式写入，key必须满足：①第一位是字母G，②第二至四位是 >=100 且 <=999 的整数` 详细说明见：附3：事件传输说明	new JSONObject() .put("G222", "123456") .put("G333", "Hello") .toString()

返回值类型：void

使用示例：

```
try {
    JSONObject properties = new JSONObject();
    properties.put("G222", "123456");    // 设置信息ID
    properties.put("G333", "Hello!");    // 设置信息内容

    OASAPI.sharedInstance().trackJumpApp(
        "sid",
        "svid",
        "com.xx.tarApp",
        "促销",
        "com.xx.tarApp.Sale",
        properties.toString()
    );
} catch (JSONException e) {
    e.printStackTrace();
}
```

调试信息：

```
...
```

3.2 数据上报方法

3.2.1 利用track方法上报项目相关信息

说明：对于某些非页面埋点的场景，比如只进行数据上传而不关心页面、控件等无关信息，可以利用track方法的自定义字段携带要上传的信息（此方法实际上就是3.1.1的track方法，只是使用时侧重点不同；与2.2.1不同的是：独立集成情况下无关字段也会自动采集，不关注这些信息即可）

方法：**track**

type：track

参数：

参数	说明	示例
String eventName	事件名称（用户自行设置）	MsgEvent
String sid	数据模板ID	***
String svid	数据模板版本ID	***

参数	说明	示例
String properties（可选）	包含自定义的json字符串 用户自定义事件的数据类型会被转为String 注意：`自定义属性的key以编号的形式写入，key必须满足：①第一位是字母G，②第二至四位是 >=100 且 <=999 的整数` 详细说明见：附3：事件传输说明	new JSONObject() .put("G222", "123456") .put("G333", "Hello") .toString()

返回值类型：void

使用示例：

```
try {
    JSONObject properties = new JSONObject();
    properties.put("G222", "123456");    // 设置信息ID
    properties.put("G333", "Hello!");    // 设置信息内容

    OASAPI.sharedInstance().track(
        "MsgEvent",
        "sid",
        "svid",
        properties.toString()
    );
} catch (JSONException e) {
    e.printStackTrace();
}
```

调试信息：

```
...
```

4 附件

附1 常用的事件名称定义

目前对事件名称并没有严格的约定或定义，由调用者遵守以下规则自行定义即可：

1. 不能以下划线开头、不能以数字开头
2. 只能包含：大小写字母、数字、下划线
3. 长度不超过100

附2 事件格式说明

根据文档《SGM-车联网数据工厂平台-API接口说明文档》（dcp平台）说明的格式规范，对埋点数据格式做出相应调整

> 平台标准的格式为固定格式：该格式外层是一个records数组，其中包含一个或多个value对象，其中value中包含消息ID（mid）、设备ID（did）、应用ID（aid）、数据模板ID（sid）、数据模板版本ID（svid）、数据采集时间戳（ts）、采集数据（d）

埋点数据格式更改如下

1. 该格式外层是一个`records数组`，其中包含一个或多个`value对象`
2. `value对象`中包含 mid）、设备ID（did）、应用ID（aid）、数据模板ID（sid）、数据模板版本ID（svid）、数据采集时间戳（ts）、`采集数据（d）`

> 消息ID（mid）：由程序自动生成uuid
>
> 设备ID（did）：对统一服务是加密后的vin，对于独立APP是SDK采集的加密后的设备id
>
> 应用ID（aid）：APP唯一标识
>
> 数据模板ID（sid）：传入
>
> 数据模板版本ID（svid）：传入
>
> 数据采集时间戳（ts）：采集数据的时间戳，由SDK生成
>
> 采集数据（d）：SDK采集的数据与用户自定义的属性

3. `采集数据（d）`中所有数据全部扁平化处理，只保留一层json
4. value对象中的`did`：（如果未来统一服务和独立APP要合成一套SDK），SDK在统一服务中使用则传入加密的vin号，如果在独立APP中使用则传入SDK自动采集的设备id

```
{
  "records":[
    {
      "value":{
        "mid": "d32707d8-3f7e-4795-9d57-de9957988f0b",      // 消息ID（由SDK生成）
        "did": "0593ce2837c2450329l0da1edd3a8a9e",          // 设备ID
                                                              //     统一服务：vin
                                                              //     独立APP：SDK采集的设备id
        "aid": "87654321",                                  // 应用ID/BasicAuth账号（set方法）
        "sid": "123456",                                    // 数据模板ID（传入）
        "svid": "1.0",                                      // 数据模板版本ID（传入）
        "ts": "1591189249156",                              // 数据采集时间戳（由SDK生成）
        "d":{                                               // 采集数据（d）
            "track_id":"-2008771570",
            "time":"1603682005724",
            "type":"track",
            "lib":"Android",
            "lib_version":"1.0.0",
            "lib_method":"code",
            "lib_detail":"com.bigtimes.sdk.BigtimesDataAPI##..."
            "event":"ForcedToSend",
            "os_version":"10",
            "lib_version":"1.0.0",
            "model":"Android SDK built for x86",
            "timezone_offset":"0",
            "os":"Android",
            "screen_width":"1080",
            "screen_height":"1920",
            "manufacturer":"Google",
            "app_version":"app_version_##",
```

```
            "hardwareVersion":"v1.0",
            "channel":"channel_##",
            "wifi":"true",
            "network_type":"WIFI",
            "is_first_day":"false",
            "flush_time":"1603682005793"
        }
      }
    },
    {},{},...{}
  ]
}
```

附3 事件传输说明

为减少传输开销，传输之前会将上面埋点数据 `采集数据（d）` 中的全部字段名映射为编号，再加密压缩后传输

- 用户自定义事件

用户传入包含自定义属性的json字符串，其中 `自定义属性的key必需满足：①是整数，②100 <= key <= 999`

根据平台设计：后续用户使用SDK时需要先在平台注册自定义字段名称及编号，平台将记录此映射关系，供解析数据使用

> 为防止冲突，用户自定义事件字段的属性名请避开下表的SDK预设属性的属性名

- SDK预设属性

SDK的预设属性编号为 1~99 的两位正整数，由SDK内部定义，预设属性与编号的映射关系表

如下

属性名	是否每种事件都带有	说明	编号	是否为SDK自动获取
lib	Y	SDK类型，如Android	G1	Y
os_version	Y	操作系统版本，例如8.1.0	G2	Y
lib_version	Y	SDK版本	G3	Y
model	Y	设备型号，例如iphone 8,4	G4	Y
timezone_offset	Y	设备所在的时区偏移量的分钟数 * -1，如：-480	G5	Y
os	Y	操作系统，例如Android	G6	Y
screen_width	Y	屏幕宽度	G7	Y
screen_height	Y	屏幕高度	G8	Y
manufacturer	Y	设备制造商，例如Google	G9	Y
wifi	Y	是否使用wifi	G10	Y

属性名	是否每种事件都带有	说明	编号	是否为SDK自动获取
network_type	Y	网络类型，例如4G	G11	Y
cur_screen	N（在独立集成模式下：用户未在activity调用则无）	当前页面名称	G12	N
event_duration	N（获取事件时长）	停留时长，单位为秒	G13	Y
app_version	Y	APP应用的版本	G14	N
track_id	Y	事件的标识	G15	Y
time	Y	事件发生的实际时间戳，单位为毫秒	G16	Y
type	Y	事件的类型，默认track追踪事件	G17	Y
event	Y	事件名称	G18	N
is_first_day	Y	是否首日访问	G19	Y
lib_method	Y	埋点的触发方式（全埋点、自定义埋点）	G20	Y
lib_detail	Y	触发埋点时的调用栈信息	G21	Y
app_id	Y	应用ID	G22	N
hardware_version	N（用户未设置则无）	硬件版本	G23	N
channel	N（用户未设置则无）	渠道，将在后续用来做渠道分布分析	G24	Y
referrer	N（无）	上一个页面名称	G25	N
flush_time	Y	发送数据的时间戳，单位为毫秒	G26	Y
tar_screen	N（跳转页面）	目标页面名称，用于跳转页面	G27	N
cur_activity	N（在独立集成模式下：用户未在activity调用则无）	当前activity名称	G28	N
tar_activity	N（跳转页面）	目标activity名称	G29	N
cur_fragment	N（Fragment相关）	当前fragment名称（详情页1）	G30	N

属性名	是否每种事件都带有	说明	编号	是否为SDK自动获取
tar_fragment	N（Fragment相关）	目标fragment名称（详情页2）	G31	N
tar_app	N（跳转App）	目标app名称（com.xx.tarApp）	G32	N
usr_id	N（用户传入则有）	idp_userid	G33	N

- 除了用户自定义事件与SDK预设属性

SDK默认每条消息嗨带有模板、应用、设备相关信息

属性名	是否每种事件都带有	说明
mid	Y	消息ID（由程序自动生成的uuid）
aid	Y	应用ID（用户在平台申请的应用的id，用户传入）
sid	Y	数据模板ID（用户在平台申请的应用的数据模板id，用户传入）
svid	Y	数据模板版本ID（用户在平台申请的模板的版本id，用户传入）
did	Y	设备ID 对于统一服务：vin 对于独立集成：SDK采集的设备id

进行字段映射后的整条埋点数据样例

```
{
    "records":[
        {
            "value":{
                "mid": "d32707d8-3f7e-4795-9d57-de9957988f0b",  // 消息ID（程序生成）
                "aid":"aid",                                    // 应用ID（传入）
                "sid":"sid",                                    // 数据模板ID（传入）
                "svid":"svid",                                  // 数据模板版本ID（传入）
                "did":"H4sIAAAAAAAAAHM0MgAAOFqwRQMAAAA=",       // 设备ID
                                                                //     统一服务：vin
                                                                  //     独立APP：SDK采集的设备id
                "ts": "1591189249156",                          // 数据采集时间戳（由SDK生成）
                "d":{
                    "G1":"Android",
                    "G2":"10",
                    "G3":"1.0.0",
```

```
                    "G4":"Android SDK built for x86",
                    "G5":"0",
                    "G6":"Android",
                    "G7":"1080",
                    "G8":"1920",
                    "G9":"Google",
                    "G10":"true",
                    "G11":"WIFI",
                    "G14":"app_version_##",
                    "G12":"com.big16s.analytics.android.demo.MainActivity",
                    "G15":"-572943831",
                    "G16":"1604041528539",
                    "G17":"track",
                    "G18":"EnterBackground",
                    "G19":"false",
                    "G20":"code",
                    "G21":"com.big16s.sdk.Big16sDataAPI...",
                    "G23":"v1.0",
                    "G24":"24",
                    "G26":"1604041528596",
                    ...
                    "G101":"用户自定义属性"
                }
            }
        }
    ]
}
```


进行加密和压缩后的埋点数据示例（最后实际发送的数据）


```
{
    "records":[
        {

 "value":"H4sIAAAAAAAAAFWQb2+CMBDGvwpp35qmxVqQZC8wzkkWsj8u2culApJOpBlUNmP87rvr4jb
6on2eu+vdrzOTbUqS4P5GKZmQ3tv+1w4/frgGSu/Xss/S61rnPK/T9GH18fn8lGPkBgtJciYCatO27Czc
mpAQnOAgpigYZ6jlXOmwWd4H26NpXLCzXfAVK8jPSMInRIOaRSQRPIZwDGIegphD/s7auqkgDTMS1xOrU
AjwmqOyjOIkPBGjsAe2NbVQPdOtbk7OFKj8AFZWB8tybdqOcGYw7oS3Zn7mLIyjuRTggUgoLrlUUnAhJI
QAi7hOF3usBzSyAP1iV7ar6s4eWOQXQLrTTQ9wIfcgJSKHYgzVl3u28HKpnU4fMOp959uhah2loxR714O
mdBopgZ3wcwfh/zbEJ8MG6j+uimJ5uXwDTgqWNPsBAAA="
        },
        {

 "value":"H4sIAAAAAAAAAFWQzW6DMBCEXwXZ19SywTUEqQeiJA2qUP8OPVYOEOSGYBUc2ijKu3fXVVr
BwczMrteffSbaVCTF9Z1SMiODt8OfHX/9eAOq7zdyyLPrtyl4OWTZ4/rz6+W5wOQOGOl6JgJ6s67qLeya
kRCc4CAiFIwz1PK/JXhdPgTbo2ldsLN98JOoqN+SlM+ImgyKSSp4AnECYh6CmEP93tqmraEMZ6SuP9agE
OAtX+eY4kn4R4zSHtjWNEINTHe6PTlTovIHsKo+WFZoO2WlM6NxJ9wFHDdCRFGUxDwBDORCccmlkkLFUk
EEWMT1utxjP6CRVefqfgFBO9tjh+QCQHe6HYAt5J6jQuJQTJmGas8WXi61O9lTTqkfvBrrzlE6KbEPPWp
Ko1gJnIRvOwr/tCHeGBZQU9pEXi4/bruTC/oBAAA="
        }
    ]
}
```

5 注意事项

5.1 在Service中调用SDK方法

如果用户需要在Service中使用SDK的方法，请注意初始化SDK与调用SDK方法是否都在Service中进行，如果初始化SDK不在Service中进行而调用SDK方法在Service中进行，那么请去掉`AndroidManifest.xml`中的如下配置

```
android:process=":remote"
```

否则Service会创建新的进程，导致初始化SDK与调用SDK方法不在同一进程中，从而无法正常工作

> Service 未设置 android:process=":remote"：Service会和Activity是在同一个进程中的，而且都是主线程
> Service 设置了 android:process=":remote"：会创建新进程，这时Activity可以通过跨进程访问Service中的数据

5.2 用户自定义字段不支持直接设置JsonArray

当前（20210521）用户自定义字段不支持直接设置JsonArray

如：

```
JSONArray jsonArray = new JSONArray();
jsonArray.put("field1", "...");
jsonArray.put("field2", "...");
jsonArray.put("field3", "...");
...

// 自定义字段G118直接设置jsonArray
properties.put("G118", jsonArray.toString());
// 或
properties.put("G118", jsonArray);
```

会报错：

```
W/System.err: org.json.JSONException: Value •������������������V*.I of type
java.lang.String cannot be converted to JSONObject
W/System.err:     at org.json.JSON.typeMismatch(JSON.java:112)
        at org.json.JSONObject.<init>(JSONObject.java:168)
        at org.json.JSONObject.<init>(JSONObject.java:181)
        at
com.sensorsdata.analytics.android.sdk.util.HttpClientUtils.httpClientSend(HttpCli
entUtils.java:128)
...
```

报错原因：对于这种设置方法，下游接收数据端无法成功解析，会返回CODE500与带乱码的Response，故SDK在发送数据后解析Response时报解析Json错误（HttpClientUtils.java:128）。这个错误是由于接收数据端不支持所致，仅从SDK端无法解决

可行的办法：`将JsonArray用JsonObject包装后再设置入自定义字段`

```
JSONArray time = new JSONArray();
time.put("field1", "...");
time.put("field2", "...");
time.put("field3", "...");
...

JSONObject jsonObject = new JSONObject();
jsonObject.put("time", time);

properties.put("G118", jsonObject.toString());
```

6 问题排查及解决方案

6.1 上传成功

SDk上传成功会打印“20000”

```
I/RESPONSE: **************************
I/发送条数: n 条
I/CODE: 20000
I/message: ok
I/RESPONSE: **************************
```

6.2 用户反馈问题

6.2.1 问题排查步骤

```
用户排查步骤，如下
1.网络环境正常，可以访问外网
2.检查Gn...的数据格式，所有Gx后面接的是字符串，并确认用户（aid）有该模板权限
3.SDK初始化时确认所在环境，并确认SDK配置为对应环境（dev或prod）
4.确认上传数据与模板配置字段一致
5.根据本地日志检查上传结果
```

6.2.2 问题表格

问题	解决方案
DCP显示 例如 “未定义字段G33”	找DCP同事检查模板定义是否存在问题，上传字段是否与模板定义一致
DCP显示 无效数据	点击 “原始数据”，里面有相关报错信息

问题	解决方案
SDK上传字段中没有VIN号字段	SDK上传字段中不存在vin号，用户需要可以自己增加字段上传
密码变更是否需要前端手动更改匹配	SDK只获取用户传入的密码，不会变更用户传入的密码
debug环境没有问题，replace环境出现闪退 FATAL EXCEPTION: com.sensorsdata.analytics.android.sdk.AnalyticsMessages.Worker	已发版SDK并未做代码混淆，操作客户SDK集成模式不同，更改混淆规则 代码混淆问题/需要对jar包配置
SDK报错：DCP禁用接口报错	DCP禁用字段接口存在问题，需要DCP同事进行检查
SDK报错：host报错 java .net.UnknowmHostException:Unable to resolve host "collector ,srgmlink.com" ;	用户网络问题，可以切换网络试一下，比如开热点，ping一下出错的域名通不通
SDK报错：W/System.err: org.json.JSONException: Value □□□□□□□□□□□□□□□□□V*.I of type java.lang.String cannot be converted to JSONObject	1.数据发送给了dcp，dcp返回非json格式，sdk打印这一段信息转换格式报错 2.确认上传字段是否有该模板权限 3.上传数据格式是否按照规则上传
SDK报错：java.lang. InterruptedException	重复调用了onAction方法

问题	解决方案
SDK报错：java.lang.NoClassDefFoundError: Failed resolution of: Lorg/apache/http/conn/ssl/SSLContexts;	1.用户本地环境网络问题，无法通过gradle下载SDK依赖，请使用SDK发布中包含四个jar包，具体参考“SDK说明文档
SDK提示：自定义事件的key{}：必需满足：1:第一位是字母G，2: 第二至四位是大于等于100 且 小于等于999 的整数	按照SDK说明文档进行操作