

Android 统计 SDK 开发者使用指南

腾讯移动分析出品

目录

Android 统计 SDK 开发者使用指南	1
1 开始嵌入 SDK	3
1.1 安装和部署	3
1.2 升级 SDK	6
2 初始化并启动 MTA	7
3 基础指标统计	9
3.1 页面统计	9
3.2 会话统计	10
3.3 错误统计	11
4 自定义事件	13
4.1 注册自定义事件	13
4.2 【次数统计】Key-Value 参数的事件	13
4.3 【次数统计】带任意参数的事件	14
4.4 【时长统计】Key-Value 参数事件	14
4.5 【时长统计】带有统计时长的自定义参数事件	15
5 接口监控	17
6 高级功能	19
6.1 游戏统计	19

6.2 在线配置更新	19
7 数据上报	21
7.1 数据上报策略	21
7.2 数据上报相关的设置接口	22
8 APP 设置接口	23
9 注意事项	25
9.1 何时调用 StatConfig 配置接口	25
9.2 调试 app	25
9.3 发布 app	26
9.4 SDK 冲突问题	26
9.5 提示兼容性错误	28
10 特殊需求	29
10.1 激活量统计	29
10.2 只做少数页面统计	29
10.3 只统计用户打开 app 的次数	29
10.4 只统计 app 的设备、网络等信息	29
10.5 只上报 app 未捕获的异常	29
10.6 只用于接口监控	29

1 开始嵌入 SDK

1.1 安装和部署

欢迎使用腾讯移动分析 (简称 MTA) Android 统计 SDK , 您可以按照下面 6 步开始 SDK 的统计。

Step 1 获取 AppKey

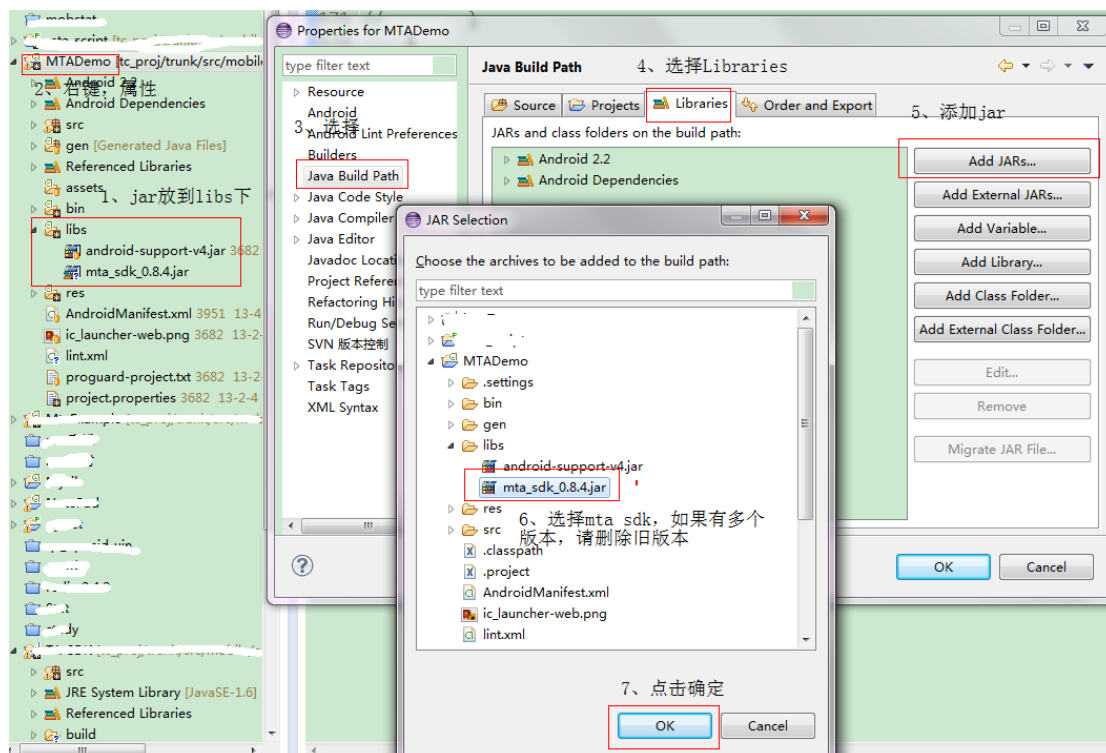
MTA 已经将开放平台的 appid 注册到系统上 , 开发者只需要在 appid 上添加前缀 , 即为 MTA 的 AppKey , 其中规则如下表 (大小写敏感) 。

系统平台	前缀
Android	Aqc
iOS	Iqc

例如 appid 为 “123456” 的应用对应于 MTA 的 **Android AppKey** 是 “**Aqc123456**” ,
iOS AppKey 是 “Iqc123456”。

Step 2 向工程中导入 SDK

(Andriod_SDK_V1.7 使用说明.doc、10 分钟接入 MTA 指南.doc---这两个文档中也有同样的操作说明 , 参照任意一个即可)



下载 SDK 压缩包，解压至本地目录，将其中 lib 目录下的 mta-sdk-x.x.x.jar¹复制到您的应用工程的库存储目录（通常为 libs 或 lib 目录）中。以 Eclipse 为例：右键点击工程根目录→选择 Properties → Java Build Path →Libraries →点击 Add JARs... 选中当前工程 libs 目录下的 mta-sdk-x.x.x.jar 文件，点击“OK”按钮即导入成功。

Step 3 配置 AndroidManifest.xml 文件

Appkey 和渠道设置

Meta-Data	类型	用途	必选
<i>TA_APPKEY</i>	String	MTA 提供给每个 app 的 appkey，用来定位该应用程序的唯一性	√
<i>InstallChannel</i>	String	用来标注应用推广渠道，区分新用户的来源来查看统计	√

（注意：appkey 和 installChannel 也能够代码中设置，见 [APP 设置接口](#)）

权限设置

¹x.x.x 为 SDK 版本号，以实际 SDK 的名字为准

需要的权限	用途	必选
<code>INTERNET</code>	允许应用程序联网，以便向我们的服务器端发送数据	√
<code>READ_PHONE_STATE</code>	获取用户手机的 IMEI，用来唯一的标识用户。(运行在平板上的应用会读取 mac 地址作为用户的唯一标识)	√
<code>ACCESS_NETWORK_STATE</code>	获取设备的网络状态	√
<code>ACCESS_WIFI_STATE</code>	获取设备的 WIFI 网络状态	√
<code>WRITE_EXTERNAL_STORAGE</code>	获取 SD 卡信息	√

示例文件：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest .....>
    <uses-sdk android:minSdkVersion="7" android:targetSdkVersion="7"/>
    <!--为MTA授权。 <-->
    <!--如果是第三方lib项目，请在手册中提示app开发者授予以下权限！ <-->
    <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.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <!-- 请在application处配置appkey和渠道或在代码处调用StatConfig类接口 <-->
    <application .....>
        <activity .....>
            .....
        </activity>
    <!-- 请将value改为MTA分配的appkey，即开放平台appid加上“Aqc”前缀 <-->
    <meta-data android:name="TA_APPKEY" android:value="Aqc123456"/>
    <!-- 请将value改为app发布对应的渠道，不同的发布渠道使用不同的名字 <-->
    <meta-data android:name="InstallChannel" android:value="play"/>
    </application>
</manifest>
```

其中，AppKey 也可通过初始化代码设置。

```
String open_appid = "开放平台 appid";
StatConfig.setAppKey(this, "Aqc" + open_appid);
```

Step 4 在代码中添加 SDK 的引用

```
import com.tencent.stat.StatConfig
```



```
import com.tencent.stat.StatService
```

StatConfig类：MTA配置类，可以设置上报策略、Debug开关、session超时时间等，**需要在初始化MTA之前被调用才能及时生效**，通常使用SDK默认配置即可。

StatService类：MTA统计类，需要开发者按下面的步骤主动调用接口。

Step 5 添加 SDK 的统计

在代码处调用类StatService提供的接口（见章节2、3、4、5、6），开始嵌入MTA的统计功能。

Step 6 验证数据上报是否正常

当您完成以下的MTA嵌入工作后，启动app，触发MTA统计接口，经过5秒左右，正常情况下，在您的app首页就能看到实时指标在更新，说明您已成功嵌入MTA，可继续深入的统计开发。

如果经过几分钟后，尚未看到实时指标更新，请检查以下事项：

- 1、设备的wifi是否打开，是否正常联网；
- 2、APPKEY、权限等设置是否正确；
- 3、确保已触发MTA统计接口；
- 4、打开MTA的debug开关，查看标签为“MtaSDK”的logcat提示，是否有错误日志。
- 5、如果logcat提示“Compatibility problem was found in this device! ”，

请先删除apk重新安装，可参考[兼容性错误](#)。

1.2 升级 SDK

新版本SDK兼容老版本接口，升级时只需要替换旧的jar包即可：先在工程所在libs目录下删除旧的jar包，复制新jar包到libs路径，同时，在Java Build Path里面删除旧的jar包，并添加新jar包引用。

2 初始化并启动 MTA

在所有其它 StatService 方法被调用之前调用以下接口初始化 MTA。 **第三方 lib 项目必须**

初始化 MTA，其它非 lib 类的项目可自行决定是否初始化，不影响正常使用。 初始化 MTA

并不会上报任何数据，仅仅是激活 MTA，并预加载数据库的配置信息。

boolean StatService.startStatService(Context ctx, String appkey,

String mtaSdkVersion)

参数：Ctx 页面的设备上下文

Appkey MTA 提供的 appkey，若为 null，则按读取 StatConfig.setAppKey()或

manifest.xml 配置的 appkey

requiredMtaVer 当前 app 依赖的 MTA SDK 版本号，只能为
com.tencent.stat.common.StatConstants.VERSION，用于 SDK 版本冲突检测

MtaSdkException 异常：启动失败时会抛出 MtaSdkException 异常，可能是参数出错，也可能是 SDK 版本冲突，具体的冲突解决办法见注意事项中的“[SDK 冲突问题](#)”。同时，MTA 会自动禁止所有功能。

调用位置：

- 1、对于普通 app：AndroidManifest.xml 指定首先启动的 activity 的 onCreate()处，StatConfig 类的方法之后。
- 2、对于 lib 工程，在其它所有 StatService 方法被调用之前，StatConfig 类的方法之后。

(注意 : StatConfig 配置类需要在此方法前才能及时生效)

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_mtamain);
    // androidManifest.xml指定本activity最先启动
    // 因此，MTA的初始化工作需要在本onCreate中进行
    // 在startStatService之前调用StatConfig配置类接口，使得MTA配置及时生效
    initMTAConfig(true);
    String appkey = "amtaandroid0";
    // 初始化并启动MTA
    // 第三方SDK必须按以下代码初始化MTA，其中appkey为规定的格式或MTA分配的代码。
    // 其它普通的app可自行选择是否调用
    try {
        // 第三个参数必须为：com.tencent.stat.common.StatConstants.VERSION
        StatService.startStatService(this, appkey,
            com.tencent.stat.common.StatConstants.VERSION);
    } catch (MtaSdkException e) {
        // MTA初始化失败
        logger.error("MTA start failed.");
        logger.error("e");
    }
}
```


3 基础指标统计

基础指标包括页面统计，会话统计，错误统计 3 个部分。

3.1 页面统计

使用下面的函数统计某个页面的访问情况：

- 标记一次页面访问的开始

void StatService.onResume(Context ctx)

参数：Ctx 页面的设备上下文

调用位置：每个 activity 的 onResume()

```
@Override
protected void onResume() {
    super.onResume();
    StatService.onResume(this);
}
```

(注意：每次调用 onResume，MTA 会检查是否产生新会话 (session 超时)，即生成启动次数。)

- 标记一次页面访问的结束

void StatService.onPause (Context ctx)

参数：Ctx 页面的设备上下文

调用位置：每个 activity 的 onPause()

```
@Override
protected void onPause() {
    super.onPause();
    StatService.onPause(this);
}
```

(注意：onResume 和 onPause 需要成对使用才能正常统计 activity，为了统计准确性，建议在每个 activity 中都调用以上接口，否则可能会导致 MTA 上报过多的启动次数，解决

办法参考 [“特殊需求”](#))

➤ 通过继承的方式统计页面访问

开发者可以通过 app 本身的 activity 基类，调用 MTA 的 onResume 和 onPause，并在所有子类中重载这 2 个方法，实现页面统计功能。可参考 MtaDemo 中的 BaseActivity 和 DrivedActivity 代码。

另外，MTA SDK 中的下面两个类实现了 StatService.onResume() 和 StatService.onPause() 的调用，可直接继承以下类并在子类中重载页面统计接口。

com.tencent.stat.EasyActivity 继承自 android.app.Activity

com.tencent.stat.EasyListActivity 继承自 android.app.ListActivity

3.2 会话统计

会话统计用于统计启动次数，由 SDK 本身维护，通常开发者无需额外设置或调用接口。

以下 3 种情况下，会视为用户打开一次新的会话：

- 1) 应用第一次启动，或者应用进程在后台被杀掉之后启动
- 2) 应用退到后台或锁屏超过 x 之后再次回到前台

x 秒通过 StatConfig.setSessionTimeoutMillis(int) 函数设置，默认为 30000ms，即 30 秒

举例说明：用户打开手机 QQ 连续操作 10 分钟，之后按 home 键（或锁屏）退到后台，超过 30 秒后再次回到 QQ，此时，SDK 会上报一次会话。

注意：请根据您的 app 业务情况决定是否调整超时时间。

- 3) 调用 SDK 提供的 startNewSession() 函数

void StatService.startNewSession(Context ctx)

参数：Ctx 页面的设备上下文

3.3 错误统计

收集应用程序的异常信息可以帮助您完善自己的程序。此外，SDK 可以帮助开发者检查 app 未捕获的异常。

1) 上报 app 捕获的错误或异常

void StatService.reportError(Context ctx, String errmsg)

参数：Ctx 页面的设备上下文

errmsg 出错信息字符串

调用位置：需要上报错误信息的地方

void StatService.reportException(Context ctx, Throwable err)

参数：Ctx 页面的设备上下文

err 抛出的异常

调用位置：需要上报异常信息的地方

```
try{
    int retval = totalNumber / personInRoom;
}
catch (ArithmeticException e){
    StatService.reportException(this, e);
}
```

2) 未捕获的异常

SDK 将异常处理类设置到线程上，作为所有线程出现未捕获异常时的缺省行为。因此，只要集成了 SDK，app 中未捕获到的异常也能上报。

(注意：mta 兼容其它未捕获的异常处理类，即如果 app 已设置 Thread.setDefaultUncaughtExceptionHandler()，MTA 会在处理完异常后，再调用原来的异常处理函数继续处理。)

可以通过下面的接口关闭自动捕获异常功能（默认为 true）

void StatConfig.setAutoExceptionCaught(boolean isAutoExceptionCaught)

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // app的最初始代码处
    StatConfig.setAutoExceptionCaught(false);
}
```

通过LogCat检查有以下警告，表明已经成功禁用了本功能。

```
MtaSDK [main(1): StatService.java:107] - AutoExceptionCaught is disable
```


4 自定义事件

可以统计某些用户自定义事件的发生次数,时间,变化趋势,例如按钮点击、购买数量等等,通常 `event_id` 用于表示某种行为或功能的统计(如统计“点击”按钮被触发多少次),而参数则用于标识统计的具体对象(如名称为“OK”的按钮),由“`event_id`”和“参数”唯一标识一个事件。

自定义事件分为 2 大类:

- 1、统计次数:统计指定行为被触发的次数
- 2、统计时长:统计指定行为消耗的时间,单位为秒。需要 `begin` 接口与 `end` 接口成对使用才生效。

其中每类事件都有 Key-Value 参数类型和不定长字符串参数类型,由于 Key-Value 参数类型的接口能表达更丰富的内容,我们**推荐优先使用 Key-Value 类参数接口**。另外,如果代码同时使用了这 2 种参数类型,`event_id` 最好不一样。

注意: `event_id` 需要先在腾讯移动分析网站上面注册,才能参与正常的数据统计。`event_id` 不能包含空格或转义字符。

4.1 注册自定义事件

自定义事件的注册(配置)包括事件 id 的注册和事件 id 下参数信息的注册。

- 1、登陆 mta 前台,选择左边选择“自定义事件”。
- 2、选择“新增事件”,按照需求填写事件 id、key、value 等信息。
- 3、可以在查看详情下的“参数”查看事件 id 下所有参数上报的明细。

4.2 【次数统计】Key-Value 参数的事件

```
void StatService.trackCustomKVEvent(Context ctx, String event_id,  
Properties properties)
```


参数： `Ctx` 页面的设备上下文

`event_id` 事件标识

`properties` Key-Value 参数对，key 和 value 都是 String 类型

调用位置：代码任意处

```
public void onOKBtnClick(View v) {
    // 统计按钮被点击次数，统计对象：OK按钮
    Properties prop = new Properties();
    prop.setProperty("name", " OK ");
    StatService.trackCustomKVEvent(this, " button_click", prop);
}

public void onBackBtnClick(View v) {
    // 统计按钮被点击次数，统计对象：back按钮
    Properties prop = new Properties();
    prop.setProperty("name", " back ");
    StatService.trackCustomKVEvent(this, " button_click", prop);
}
```

4.3 【次数统计】带任意参数的事件

void StatService.trackCustomEvent(Context ctx, String event_id, String... args)

参数： `Ctx` 页面的设备上下文

`event_id` 事件标识

`args` 事件参数

调用位置：代码任意处

```
public void onClick(View v) {
    // 统计按钮被点击次数，统计对象：OK按钮
    StatService.trackCustomEvent(this, "button_click", "OK ");
}
```

4.4 【时长统计】Key-Value 参数事件

可以指定事件的开始和结束时间，来上报一个带有统计时长的事件。

void StatService.trackCustomBeginKVEvent(

Context ctx, String event_id, Properties properties)

void StatService.trackCustomEndKVEvent(

Context ctx, String event_id, Properties properties)

参数： Ctx 页面的设备上下文

event_id 事件标识

properties Key-Value 参数对，key 和 value 都是 String 类型

调用位置：代码任意处

```
public void onClick(View v) {
    Properties prop = new Properties();
    prop.setProperty("level", "5");
    // 统计用户通关所花时长，关卡等级： 5
    // 用户通关前
    StatService.trackCustomBeginKVEvent(this, "playTime", prop);
    // 用户正在游戏中....
    // .....
    // 用户通关完成时
    StatService.trackCustomEndKVEvent(this, "playTime", prop);
}
```

注意：trackCustomBeginKVEvent 和 trackCustomEndKVEvent 必须成对出现，且 event_id 和参数列表完全相同，才能正常上报事件。

4.5 【时长统计】带有统计时长的自定义参数事件

可以指定事件的开始和结束时间，来上报一个带有统计时长的事件。

void StatService.trackCustomBeginEvent(

Context ctx, String event_id, String... args)

void StatService.trackCustomEndEvent(

Context ctx, String event_id, String... args)

参数： `Ctx` 页面的设备上下文

`event_id` 事件标识

`args` 事件参数

调用位置：代码任意处

```
public void onClick(View v) {  
    // 统计用户通关所花时长  
    // 用户通关前  
    StatService.trackCustomBeginEvent(this, " playTime", "level5");  
    // 用户正在游戏中....  
    // .....  
    // 用户通关完成时  
    StatService.trackCustomEndEvent(this, " playTime", " level5");  
}
```

注意：trackCustomBeginEvent 和 trackCustomEndEvent 必须成对出现，且 eventid 和参数列表完全相同，才能正常上报事件。

5 接口监控

统计应用对某个外部接口(特别是网络类的接口,如连接、登陆、下载等)的调用情况。

当开发者用到某个外部接口,可调用该函数将一些指标进行上报,MTA 将统计出每个接口的调用情况,并在接口可用性发生变化时进行告警通知;对于调用量很大的接口,也可以采样上报,云监控统计将根据 sampling 参数在展现页面进行数量的还原。

void StatService.reportAppMonitorStat (

Context ctx, StatAppMonitor monitor)

参数： `ctx` 页面的设备上下文

`monitor` 监控对象,需要根据接口情况设置接口名称、耗时、返回值类型、

返回码、请求包大小、响应包大小和采样率等信息,详见 doc/api 目录下的文档

调用位置：被监控的接口

StatAppMonitor 方法列表

接口名	说明
setInterfaceName(String interfaceName)	设置监控的接口名称
setReqSize(long reqSize)	请求包大小, 单位: byte
setRespSize(long respSize)	响应包大小, 单位: byte
setResultType(int resultType)	SUCCESS_RESULT_TYPE; FAILURE_RESULT_TYPE; LOGIC_FAILURE_RESULT_TYPE
setMillisecondsConsume(long millisecondsConsume)	调用耗时, 单位: 毫秒 (ms)
setReturnCode(int returnCode)	监控接口业务返回码
setSampling(int sampling)	采样率: 默认为 1, 表示 100%。如果是 1/2, 则填 2, 如果是 1/4, 则填 4, 若是 1/n, 则填 n

// 新建监控接口对象

```
StatAppMonitor monitor = new StatAppMonitor("ping:www.qq.com");
String ip = "www.qq.com";
Runtime run = Runtime.getRuntime();
java.lang.Process proc = null;
try {
    String str = "ping -c 3 -i 0.2 -W 1 " + ip;
```



```
long starttime = System.currentTimeMillis();
// 被监控的接口
proc = run.exec(str);
proc.waitFor();
long difftime = System.currentTimeMillis() - starttime;
// 设置接口耗时
monitor.setMillisecondsConsume(difftime);
int retCode = proc.waitFor();
// 设置接口返回码
monitor.setReturnCode(retCode);
// 设置请求包大小, 若有的话
monitor.setReqSize(1000);
// 设置响应包大小, 若有的话
monitor.setRespSize(2000);
// 设置抽样率
// 默认为 1, 表示 100%。
// 如果是 50%, 则填 2(100/50), 如果是 25%, 则填 4(100/25), 以此类推。
monitor.setSampling(2);
if (retCode == 0) {

    logger.debug("ping 连接成功");

    // 标记为成功
    monitor.setResultType(StatAppMonitor.SUCCESS_RESULT_TYPE);
} else {

    logger.debug("ping 测试失败");

    // 标记为逻辑失败, 可能由网络未连接等原因引起的
    // 但对于业务来说不是致命的, 是容忍的
    monitor.setResultType(StatAppMonitor.LOGIC_FAILURE_RESULT_TYPE);
}
} catch (Exception e) {
    logger.e(e);
    // 接口调用出现异常, 致命的, 标识为失败
    monitor.setResultType(StatAppMonitor.FAILURE_RESULT_TYPE);
} finally {
    proc.destroy();
}
// 上报接口监控
StatService.reportAppMonitorStat(ctx, monitor);
```


6 高级功能

6.1 游戏统计

统计游戏用户需要调用下面接口方法上报游戏用户 ID , 分区 , 等级相关信息。若在 App 用户使用过程中 , 相关信息发生改变 , 需要重新调用接口上报。

void StatService.reportGameUser(Context ctx, StatGameUser gameUser)

参数：ctx 页面的设备上下文

gameUser 游戏玩家对象 , 需要设置分区分服、玩家账号和等级 , 即需要调用 StatGameUser 类的 set 方法。

调用位置：游戏用户登陆

StatGameUser 方法列表

接口名	说明
setWorldName(String worldName)	游戏区服名称或标志
setAccount(String account)	玩家账号
setLevel(String level)	等级

```
@Override
protected void userLogin() {
    // 游戏用户上报
    StatGameUser gameUser = new StatGameUser();
    gameUser.setWorldName("world1");
    gameUser.setAccount("account1");
    gameUser.setLevel("100");
    StatService.reportGameUser(ctx, gameUser);
}
```

6.2 在线配置更新

开发者在腾讯移动分析网站上设置 Key-Value 值之后 , 可以调用下面的接口动态获取线上最新的参数值。

SDK 如何更新本地参数：用户在前台配置在线参数 , 并不是实时下发的 , 而是当 SDK 上报会话统计时才会更新。调试时 , 可在配置参数 10 分钟后 , 让 app 退到后台超过 30 秒

发生超时或把 app 进程杀死重启，便会更新。

String StatConfig.getCustomProperty(String key)

参数：key 用户在前台配置的 key

返回值：对应key的value值，若不存在则返回null

String StatConfig.getCustomProperty(String key, String defaultValue)

参数：key 用户在前台配置的 key

返回值：对应key的value值，若不存在则返回defaultValue

```
protected void someAction() {  
    // 获取在线参数onlineKey  
    String onlineValue = StatConfig.getCustomProperty("onlineKey", "off" );  
    if(onlineValue.equalsIgnoreCase("on")){  
        // do something  
    }else{  
        // do something else  
    }  
}
```

(注意：调用在线配置接口会立即返回)

7 数据上报

7.1 数据上报策略

设置数据上报策略,可以有效节省流量。使用以下 3 种方式调整 app 的数据上报策略:

- 1) app 启动时指定上报策略(默认为 APP_LAUNCH)

void StatConfig.setStatSendStrategy(StatReportStrategy statSendStrategy)

腾讯移动分析目前支持的上报策略包括 6 种。

编号	策略名称	说明
1	INSTANT	实时发送, app 每产生一条消息都会发送到服务器。
2	ONLY_WIFI	只在 wifi 状态下发送, 非 wifi 情况缓存到本地。
3	BATCH	批量发送, 默认当消息数量达到 30 条时发送一次。
4	APP_LAUNCH	只在启动时发送, 本次产生的所有数据在下次启动时发送。
5	DEVELOPER	开发者模式, 只在 app 调用 void commitEvents(Context) 时发送, 否则缓存消息到本地。
6	PERIOD	间隔一段时间发送, 每隔一段时间一次性发送到服务器。

SDK 默认为 APP_LAUNCH+wifi 下实时上报, 对于响应要求比较高的应用, 比如竞技类游戏, 可关闭 wifi 实时上报, 并选择 APP_LAUNCH 或 PERIOD 上报策略。

- 2) 考虑到 wifi 上报数据的代价比较小, 为了更及时获得用户数据, SDK 默认在 WIFI 网络下实时发送数据。可以调用下面的接口禁用此功能(在 wifi 条件下仍使用原定策略)。

void StatConfig.setEnableSmartReporting(boolean isEnabled)

- 3) 通过在 Web 界面配置, 开发者可以在线更新上报策略, 替换 app 内原有的策略, 替换后的策略立即生效并存储在本地, app 后续启动时会自动加载该策略。

上面 3 种方式的优先级顺序: wifi 条件下智能实时发送>web 在线配置>本地默认。

7.2 数据上报相关的设置接口

- 1) 设置最大缓存未发送消息个数 (默认 1024)

void StatConfig.setMaxStoreEventCount(int maxStoreEventCount)

缓存消息的数量超过阈值时, 最早的消息会被丢弃。

- 2) (**仅在发送策略为 BATCH 时有效**) 设置最大批量发送消息个数 (默认 30)

void StatConfig.setMaxBatchReportCount(int maxBatchReportCount)

- 3) (**仅在发送策略为 PERIOD 时有效**) 设置间隔时间 (默认为 24*60, 即 1 天)

void StatConfig.setSendPeriodMinutes(int minutes)

- 4) 开启 SDK LogCat 开关 (默认 false)

void StatConfig.setDebugEnable(boolean debugEnable)

(**注意: 在发布产品时, 请将此开关设为 false**)

8 APP 设置接口

使用这些函数可以动态调整 APP 和 SDK 的相关设置。

- 1) 会话时长 (默认 30000ms, 30000ms 回到应用的用户视为同一次会话)

void StatConfig.setSessionTimeoutMillis(int sessionTimeoutMillis)

- 2) 消息失败重发次数 (默认 3)

void StatConfig.setMaxSendRetryCount(int maxSendRetryCount)

- 3) 用户自定义时间类型事件的最大并行数量 (默认 1024)

void StatConfig.setMaxParallelTimingEvents(int max)

- 4) 设置安装渠道

void StatConfig.setInstallChannel(String installChannel)

- 5) 设置 app key

void StatConfig.setAppKey(Context ctx, String appkey)

- 6) 设置统计功能开关 (默认为 true)

void StatConfig.setEnableStatService(boolean enableStatService)

如果为false, 则关闭统计功能, 不会缓存或上报任何信息。

- 7) 设置 session 内产生的消息数量 (默认为 0, 即无限制)

***void StatConfig.setMaxSessionStatReportCount(int
maxSessionStatReportCount)***

如果为0, 则不限制; 若大于0, 每个session内产生的消息数量不会超过此值, 若超过了, 新产生的消息将会被丢弃。

- 8) 设置每天/每个进程时间产生的会话数量 (默认为 20)

为防止开发者调用 MTA 不合理导致上报大量的会话数量 (session), SDK 默认每天/每个进

程时间内最多产生的会话数量，当达到此值时，SDK 不再产生并上报新的会话。当进程重启或跨天时，会被清 0。

void StatConfig.setMaxDaySessionNumbers (int maxDaySessionNumbers)

9) 设置单个事件最大长度（默认为 4k，单位：bytes）

为防止上报事件长度过大导致用户流量增加，SDK 默认不上报超过 4k 的单个事件；对

于错误异常堆栈事件，异常堆栈长度不超过 100（可以超过 4k）。

void StatConfig.setMaxReportEventLength (int maxReportEventLength)

9 注意事项

开发者使用 SDK 过程中，需要注意的事项。

9.1 何时调用 StatConfig 配置接口

为了使StatConfig配置及时生效，请在app初始化时（例如mainActivity.onCreate函数）调用StatConfig提供的接口，保证StatConfig接口在StatService接口前被调用。

由于开放平台SDK修改过MTA某些配置项，当调用Tencent.createInstance()时，以下配置接口会被修改，即App原来的配置会被覆盖，请在此之后再重新配置一次。

```
StatConfig.setAutoExceptionCaught(); // 自动捕获 app 未处理的异常

StatConfig.setEnableSmartReporting(); // wifi 下实时上报

StatConfig.setSendPeriodMinutes(); //设置按 PERIOD 策略的时间间隔（只对 PERIOD 策略有效）

StatConfig.setStatSendStrategy();// 设置上报策略
```

9.2 调试 app

开发者在开发 app，调试的过程中，需要注意的事项。

1、SDK Debug 开关

如果需要查看 SDK 日志及上报数据内容，请设置 StatConfig.setDebugEnabled(true)为开启状态，但发布时，请保持关闭状态。

2、App 异常处理

SDK 默认开启捕获 app 未处理的异常并在 LogCat 打印异常信息，如果开发者在调试的时候，不希望 SDK 对 app 未处理的异常进行捕获，请将 StatConfig.setAutoExceptionCaught(false)设置为禁用状态，发布版本时，再根据情况决定是否开启。

9.3 发布 app

开发者在完成 app 开发，准备发布的过程中，需要注意的事项。

1、SDK Debug 开关

请保持 SDK 的 debug 开关为关闭状态，即调用 `StatConfig.setDebugEnabled(false)`。

2、App 异常处理

如果 app 上线时，需要收集 app 未处理的异常，请保持 `StatConfig.setAutoExceptionCaught(true)` 为开启状态。

3、渠道设置

请根据不同的发布渠道，在 `AndroidManifest.xml` 配置或代码中调用 `StatConfig.setInstallChannel()` 接口设置对应的渠道名称。

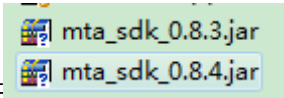
4、Appkey 及权限

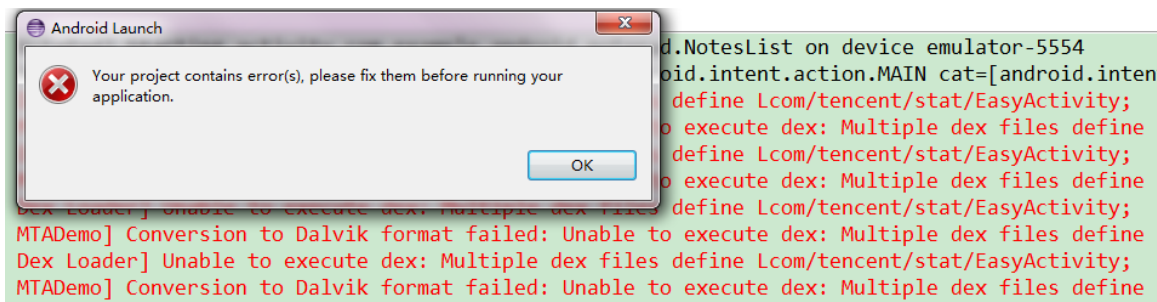
请检查 appkey 设置是否正确，且 `AndroidManifest.xml` 添加 SDK 所需权限。

9.4 SDK 冲突问题

合作方 lib 或其它 app 嵌 MTA 时，调用 MTA 初始化接口 `StatService.startStatService(Context ctx, String appkey, String requiredMtaVer)`，其中 `requiredMtaVer` 为当前 app 所依赖的版本号常量，当存在多个 app 或 lib 同时调用 MTA 时，MTA 能够根据此常量判断当前 SDK 版本是否满足条件，**只有当前使用的 SDK 版本不小于 `requiredMtaVer` 时，才不会有冲突**。当发生冲突时，可以根据以下方案排除冲突。

1 多个 SDK 文件

当存在多个 MTA SDK 文件 ，在启动 app 时，Console 提示：



解决方法：把版本号比较低的 SDK jar 包删除即可。

2 MtaSdkException

```

MtaSDK      [main(1): null:-1] - MTA SDK version conflicted, current: 0.8.3,required: 0.8.4. please delete the current SDK
and download the latest one. official website: http://mta.qq.com/ or http://mta.oa.com/
MtaSDK      [main(1): null:-1] - !!!!!MTA StatService has been disabled!!!!!!
System.err  com.tencent.stat.StatService$MtaSdkException: MTA SDK version conflicted, current: 0.8.3,required: 0.8.4. please
delete the current SDK and download the latest one. official website: http://mta.qq.com/ or http://mta.oa.com
/
System.err  at com.tencent.stat.StatService.startStatService(Unknown Source)
System.err  at com.tencent.MyLib.getService(MyLib.java:21)
System.err  at com.example.android.notepad.NotesList.onResume(NotesList.java:102)
System.err  at android.app.Instrumentation.callActivityOnResume(Instrumentation.java:1149)
System.err  at android.app.Activity.performResume(Activity.java:3763)
System.err  at android.app.ActivityThread.performResumeActivity(ActivityThread.java:2937)
System.err  at android.app.ActivityThread.handleResumeActivity(ActivityThread.java:2965)
System.err  at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:2516)
System.err  at android.app.ActivityThread.access$2200(ActivityThread.java:119)
System.err  at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1863)
System.err  at android.os.Handler.dispatchMessage(Handler.java:99)
System.err  at android.os.Looper.loop(Looper.java:123)
System.err  at android.app.ActivityThread.main(ActivityThread.java:4363)
System.err  at java.lang.reflect.Method.invokeNative(Native Method)
System.err  at java.lang.reflect.Method.invoke(Method.java:521)
System.err  at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:860)
System.err  at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:618)
System.err  at dalvik.system.NativeStart.main(Native Method)

```

原因：当前所使用的 MTA SDK 版本低于合作方要求的 MTA 版本

解决办法：

a) 确保 startStatService 传递的参数都是正确的,且 requiredMtaVer 为常量：

com.tencent.stat.common.StatConstants.VERSION

b) 使用最新 SDK：先删除当前版本，然后在 [MTA 官方网站](http://mta.qq.com/) 下载最新的 SDK，更新

当前开发环境即可。

3 0.8.4 版本之前的 SDK

对于早期的开发者，若使用 0.8.4 之前的旧 SDK，调用第三方 lib 时出现以下异常，

是由于老 SDK 不兼容新版本导致的，请更新至最新版本 MTA SDK 即可。

Tag	Text
	d com.tencent.sdk060.TestClass.<init>
dalvikvm	VFY: unable to resolve static method 3627: Lcom/tencent/stat/StatService;.startStatService () (Landroid/content/Context;Ljava/lang/String;Ljava/lang/String;)Z
dalvikvm	VFY: replacing opcode 0x71 at 0x0008
dalvikvm	Making a copy of Lcom/tencent/sdk060/TestClass;<init> code (133 bytes)
dalvikvm	VFY: unable to resolve exception class 506 (Lcom/tencent/stat/MtaSdkException;)
dalvikvm	VFY: unable to find exception handler at addr 0x29
dalvikvm	VFY: rejected Lcom/tencent/sdk060/TestClass;<init> (Landroid/content/Context;)V
dalvikvm	VFY: rejecting opcode 0x0d at 0x0029
dalvikvm	VFY: rejected Lcom/tencent/sdk060/TestClass;<init> (Landroid/content/Context;)V
dalvikvm	Verifier rejected class Lcom/tencent/sdk060/TestClass;
AndroidRuntime	Shutting down VM
dalvikvm	threadid=3: thread exiting with uncaught exception (group=0x4001b188)
AndroidRuntime	Uncaught handler: thread main exiting due to uncaught exception

9.5 提示兼容性错误

MTA 自带兼容性检测和保护，当出现兼容性问题时，logcat 会出现以下类似错误，此时，为了不影响 APP 正常使用，MTA 已自动禁止，不会采集或上报任何数据。

```
[main(1): MTAMainActivity.java:179] - Compatibility problem was found in this device!
```

解决办法：

- 1、确保 APP 已授权且没有第三方软件禁止权限；
- 2、确保 SDK 版本号是最新版本；
- 3、确保 APP 本身没有异常，并删除 apk，重装安装。

若通过以上 2 点还没排除问题，很可能是由于设备本身系统兼容性导致，请联系我们，并告知设备型号、Android 系统版本和异常信息。

10 特殊需求

10.1 激活量统计

统计首次使用 APP 的设备数量。

```
if(该用户是首次使用){  
    StatService.trackCustomEvent(this, "first", "");  
}
```

然后就可以在前台查看“first”这个自定义事件的数量，即当前 app 激活量。

10.2 只做少数页面统计

MTA 使用 android 系统的 onPause 和 onResume 做页面统计和监听用户是否退到后台，如果仅仅只是在某些页面调用了这 2 个方法，可能会导致上报大量的会话信息，可通过以下办法解决：

- 1、将 session 超时时间（默认为 30 秒）设置到一个合理值，如 1 小时
- 2、设置每天/每个进程时间产生的会话数量（默认为 20）

10.3 只统计用户打开 app 的次数

在 app 开始的地方，比如在第一个启动的 activity 的 onCreate 函数，调用
`StatService.trackCustomEvent(this, "onLaunch", "");`其中“onLaunch”为自定义事件 id，请根据实际情况修改。

10.4 只统计 app 的设备、网络等信息

在 app 开始的地方，可通过触发任意一个自定义事件激活 MTA 即可，见 10.2。

10.5 只上报 app 未捕获的异常

在 app 开始的地方调用 `StatConfig.setAutoExceptionCaught(true)` 开启 MTA 捕获异常的功能，然后再通过任意的自定义事件触发 mta 即可。

10.6 只用于接口监控

在被监控的地区调用监控接口 `reportAppMonitorStat(Context, StatAppMonitor)`

即可。