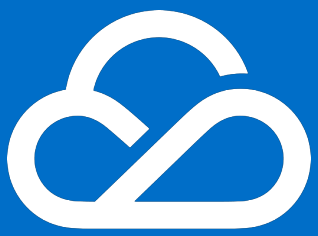
腾讯云腾讯移动分析

高级功能

产品文档



腾讯云

【版权声明】

【商标声明】

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或模式的承诺或保证。

文档目录

高级功能

高级功能接入一览表

高级功能可以帮助开发者实现更精细化的数据统计，包含质量管理、用户行为分析、用户画像分析以及更多自定义的功能。具体内容如下：

广告效果监测

错误分析

账号统计

自定义事件

接口监控

网速监控

用户画像

可视化埋点

在线参数

数据上报策略

App设置接口

用户分群

自定义用户属性

行业解决方案

自定义事件

功能介绍

自定义事件可以统计某些用户自定义埋点的发生次数、时间、变化趋势，例如广告点击、短信数量等，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。

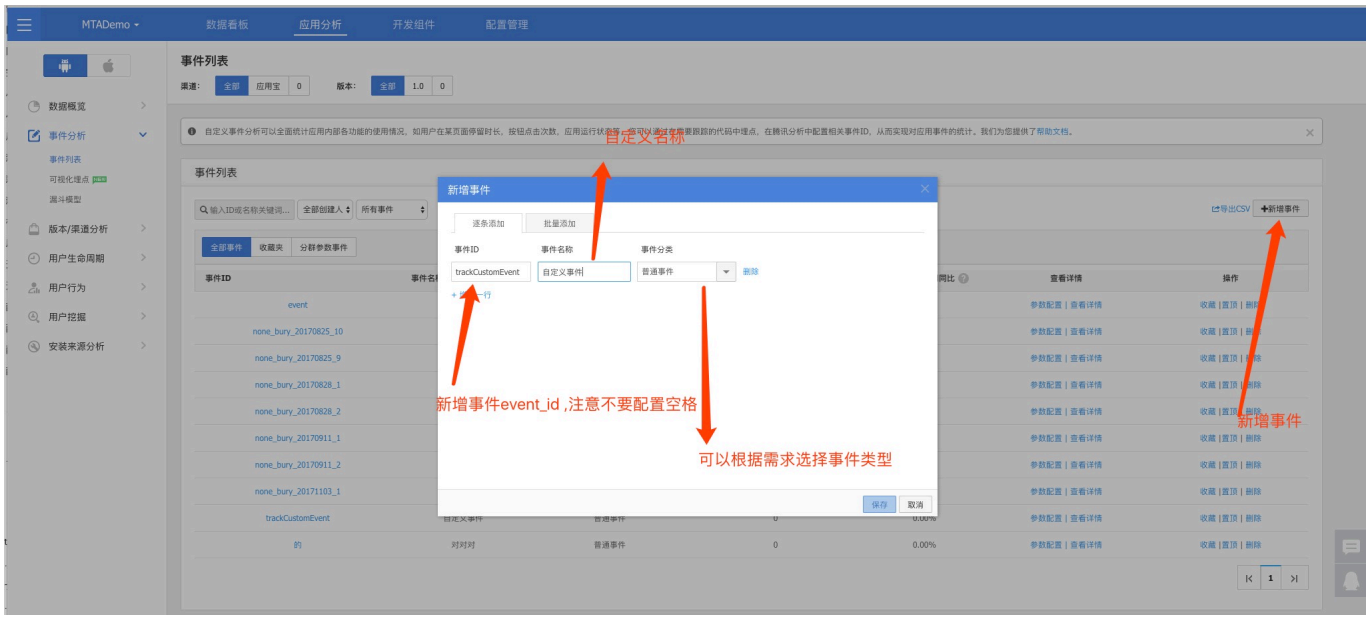
自定义事件主要用于两种场景：

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两种参数类型，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配置，才能参与正常的数据统计，event_id 不能包含空格或转义字符。

新增事件的方法如图：



新增事件参数的方法如图：

Android 自定义事件使用指南

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. 登录 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。

【次数统计】Key-Value 参数的事件

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

1. 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
2. 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);
   }

【次数统计】带任意参数的事件

1. 参数：
   Ctx：页面的设备上下文
   event_id：事件标识
   args：事件参数
2. 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 ");
   }

【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)

1. 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
2. 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);
}
```

【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)
```

1. 参数
   Ctx： 页面的设备上下文
   event_id：事件标识
   args：事件参数
2. 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 trackCustomEndKvent
   必须成对出现，且参数列表完全相同，才能正常上报事件。

iOS 自定义事件使用指南

NSDictionary 为参数的自定义事件

```
/**
 上报自定义事件

 @param event_id 事件的ID，ID需要先在MTA前台配置好才能生效
 @param kvs 事件的参数，参数需要先在MTA前台配置好才能生效
 */
+ (void)trackCustomKeyValueEvent:(NSString *)event_id props:(NSDictionary *)kvs;

/**
 上报自定义事件
 并且指定上报方式

 @param event_id 事件的ID，ID需要先在MTA前台配置好才能生效
 @param kvs 事件的参数，参数需要先在MTA前台配置好才能生效
 @param appkey 需要上报的appKey，若传入nil，则上报到启动函数中的appkey
 @param isRealTime 是否实时上报，若传入YES，则忽略全局上报策略实时上报。否则按照全局策略上报。
 */
+ (void)trackCustomKeyValueEvent:(NSString *)event_id
  props:(NSDictionary *)kvs
  appkey:(NSString *)appkey
  isRealTime:(BOOL)isRealTime;

/**
 开始统计自定义时长事件
 此接口需要跟trackCustomKeyValueEventEnd配对使用
 多次调用以第一次开始时间为准

 @param event_id 事件的ID，ID需要先在MTA前台配置好才能生效
 @param kvs 事件的参数，参数需要先在MTA前台配置好才能生效
 */
+ (void)trackCustomKeyValueEventBegin:(NSString *)event_id props:(NSDictionary *)kvs;
```

```
/**
 开始统计自定义时长事件
 并指定上报方式
 此接口需要跟trackCustomKeyValueEventEnd配对使用
 多次调用以第一次开始时间为准

 @param event_id 事件的ID，ID需要先在MTA前台配置好才能生效
 @param kvs 事件的参数，参数需要先在MTA前台配置好才能生效
 @param appkey 需要上报的appKey，若传入nil，则上报到启动函数中的appkey
 */
+ (void)trackCustomKeyValueEventBegin:(NSString *)event_id
  props:(NSDictionary *)kvs
  appkey:(NSString *)appkey;

/**
 结束统计自定义时长事件
 此接口需要跟trackCustomKeyValueEventBegin配对使用
 多次调用以第一次结束时间为准

 @param event_id 事件的ID，ID需要先在MTA前台配置好才能生效
 @param kvs 事件的参数，参数需要先在MTA前台配置好才能生效
  参数中的key和value必须跟开始统计时传入的参数一样才能正常配对
 */
+ (void)trackCustomKeyValueEventEnd:(NSString *)event_id props:(NSDictionary *)kvs;

/**
 结束上报自定义时长事件
 并指定上报方式
 此接口需要跟trackCustomKeyValueEventBegin配对使用
 多次调用以第一次结束时间为准

 @param event_id 事件的ID，ID需要先在MTA前台配置好才能生效
 @param kvs 事件的参数，参数需要先在MTA前台配置好才能生效
```

```
 参数中的key和value必须跟开始统计时传入的参数一样才能正常配对
 @param appkey 需要上报的appKey，若传入nil，则上报到启动函数中的appkey
 @param isRealTime 是否实时上报，若传入YES，则忽略全局上报策略实时上报。否则按照全局策略上报。
 */
+ (void)trackCustomKeyValueEventEnd:(NSString *)event_id
  props:(NSDictionary *)kvs
  appkey:(NSString *)appkey
  isRealTime:(BOOL)isRealTime;


/**
 直接统计自定义时长事件
 这个方法用于上报统计好的时长事件

 @param seconds 自定义事件的时长，单位秒
 @param event_id 事件的ID，ID需要先在MTA前台配置好才能生效
 @param kvs 事件的参数，参数需要先在MTA前台配置好才能生效
 */
+ (void)trackCustomKeyValueEventDuration:(uint32_t)seconds
  withEventid:(NSString *)event_id
  props:(NSDictionary *)kvs;

/**
 直接上报自定义时长事件
 并指定上报方式
 这个方法用于上报统计好的时长事件

 @param seconds 自定义事件的时长，单位秒
 @param event_id 事件的ID，ID需要先在MTA前台配置好才能生效
 @param kvs 事件的参数，参数需要先在MTA前台配置好才能生效
 @param appkey 需要上报的appKey，若传入nil，则上报到启动函数中的appkey
 @param isRealTime 是否实时上报，若传入YES，则忽略全局上报策略实时上报。否则按照全局策略上报。
 */
+ (void)trackCustomKeyValueEventDuration:(uint32_t)seconds
```

```
withEventid:(NSString *)event_id
props:(NSDictionary *)kvs
appKey:(NSString *)appkey
isRealTime:(BOOL)isRealTime;
```

示例

```
// 次数统计
- (IBAction)clickKVButton:(id)sender {
  [MTA trackCustomKeyValueEvent:@"KVEvent"
  props:[NSDictionary dictionaryWithObject:@"Value" forKey:@"Key"]];
}

// 时长统计
- (IBAction)clickStartKvButton:(id)sender {
  [MTA trackCustomKeyValueEventBegin:@"KVEvent"
  props:[NSDictionary dictionaryWithObject:@"Value" forKey:@"TimeKey"]];
}

- (IBAction)clickEndKvButton:(id)sender {
  [MTA trackCustomKeyValueEventEnd:@"KVEvent"
  props:[NSDictionary dictionaryWithObject:@"Value" forKey:@"TimeKey"]];
}
```

NSArray 为参数的自定义事件

接口

```
/**
 上报自定义事件

 @param event_id 事件的ID，ID需要先在MTA前台配置好才能生效
```

```
 @param array 事件的参数，参数需要先在MTA前台配置好才能生效
 */
+ (void)trackCustomEvent:(NSString *)event_id args:(NSArray *)array;


/**
 上报自定义事件
 并指定上报方式

 @param event_id 事件的ID，ID需要先在MTA前台配置好才能生效
 @param array 事件的参数，参数需要先在MTA前台配置好才能生效
 @param appkey 需要上报的appKey，若传入nil，则上报到启动函数中的appkey
 @param isRealTime 是否实时上报，若传入YES，则忽略全局上报策略实时上报。否则按照全局策略上报。
 */
+ (void)trackCustomEvent:(NSString *)event_id
  args:(NSArray *)array
  appkey:(NSString *)appkey
  isRealTime:(BOOL)isRealTime;

/**
 开始统计自定义时长事件
 此接口需要跟trackCustomEventEnd配对使用
 多次调用以第一次开始时间为准

 @param event_id 事件的ID，ID需要先在MTA前台配置好才能生效
 @param array 事件的参数，参数需要先在MTA前台配置好才能生效
 */
+ (void)trackCustomEventBegin:(NSString *)event_id args:(NSArray *)array;

/**
 开始统计自定义时长事件
 并指定上报方式
 此接口需要跟trackCustomEventEnd配对使用
 多次调用以第一次开始时间为准
```

```
 @param event_id 事件的ID，ID需要先在MTA前台配置好才能生效
 @param array 事件的参数，参数需要先在MTA前台配置好才能生效
 @param appkey 需要上报的appKey，若传入nil，则上报到启动函数中的appkey
 */
+ (void)trackCustomEventBegin:(NSString *)event_id
  args:(NSArray *)array
  appkey:(NSString *)appkey;

/**
 结束统计自定义时长事件
 此接口需要跟trackCustomKeyValueEventBegin配对使用
 多次调用以第一次结束时间为准

 @param event_id 事件的ID，ID需要先在MTA前台配置好才能生效
 @param array 事件的参数，参数需要先在MTA前台配置好才能生效
  参数中的各项必须跟开始统计时传入的参数一样才能正常配对
 */
+ (void)trackCustomEventEnd:(NSString *)event_id args:(NSArray *)array;

/**
 结束统计自定义时长事件
 并指定上报方式
 此接口需要跟trackCustomKeyValueEventBegin配对使用
 多次调用以第一次结束时间为准

 @param event_id 事件的ID，ID需要先在MTA前台配置好才能生效
 @param array 事件的参数，参数需要先在MTA前台配置好才能生效
  参数中的各项必须跟开始统计时传入的参数一样才能正常配对
 @param appkey 需要上报的appKey，若传入nil，则上报到启动函数中的appkey
 @param isRealTime 是否实时上报，若传入YES，则忽略全局上报策略实时上报。否则按照全局策略上报。
 */
+ (void)trackCustomEventEnd:(NSString *)event_id
  args:(NSArray *)array
```

```
  appkey:(NSString *)appkey
  isRealTime:(BOOL)isRealTime;
```

示例

```
// 次数统计
- (IBAction)clickNormaltButton:(id)sender {
  [MTA trackCustomEvent:@"NormalEvent" args:[NSArray arrayWithObject:@"arg0"]];
}

// 时长统计
- (IBAction)clickStartButton:(id)sender {
  [MTA trackCustomEventBegin:@"TimeEvent" args:[NSArray arrayWithObject:@"arg0"]];
}

- (IBAction)clickEndButton:(id)sender {
  [MTA trackCustomEventEnd:@"TimeEvent" args:[NSArray arrayWithObject:@"arg0"]];
}
```

上报当前缓存的事件

接口

```
/**
 上报当前缓存的数据
 若当前有缓存的事件（比如上报策略不为实时上报，或者有事件上报失败）时
 调用此方法可以上报缓存的事件

 @param maxStatCount 最大上报事件的条数
 */
+ (void)commitCachedStats:(int32_t)maxStatCount;
```

账号统计

功能介绍

通过账号统计，可以实现新增账号、活跃账号等维度的数据统计，相对于设备维度统计，可以帮助开发者更好地统计用户注册以及登录情况。

Android账号统计数据获取

上报自有账号：

void StatConfig.setCustomUserId(Context ctx, String customUserId)

iOS账号统计数据获取

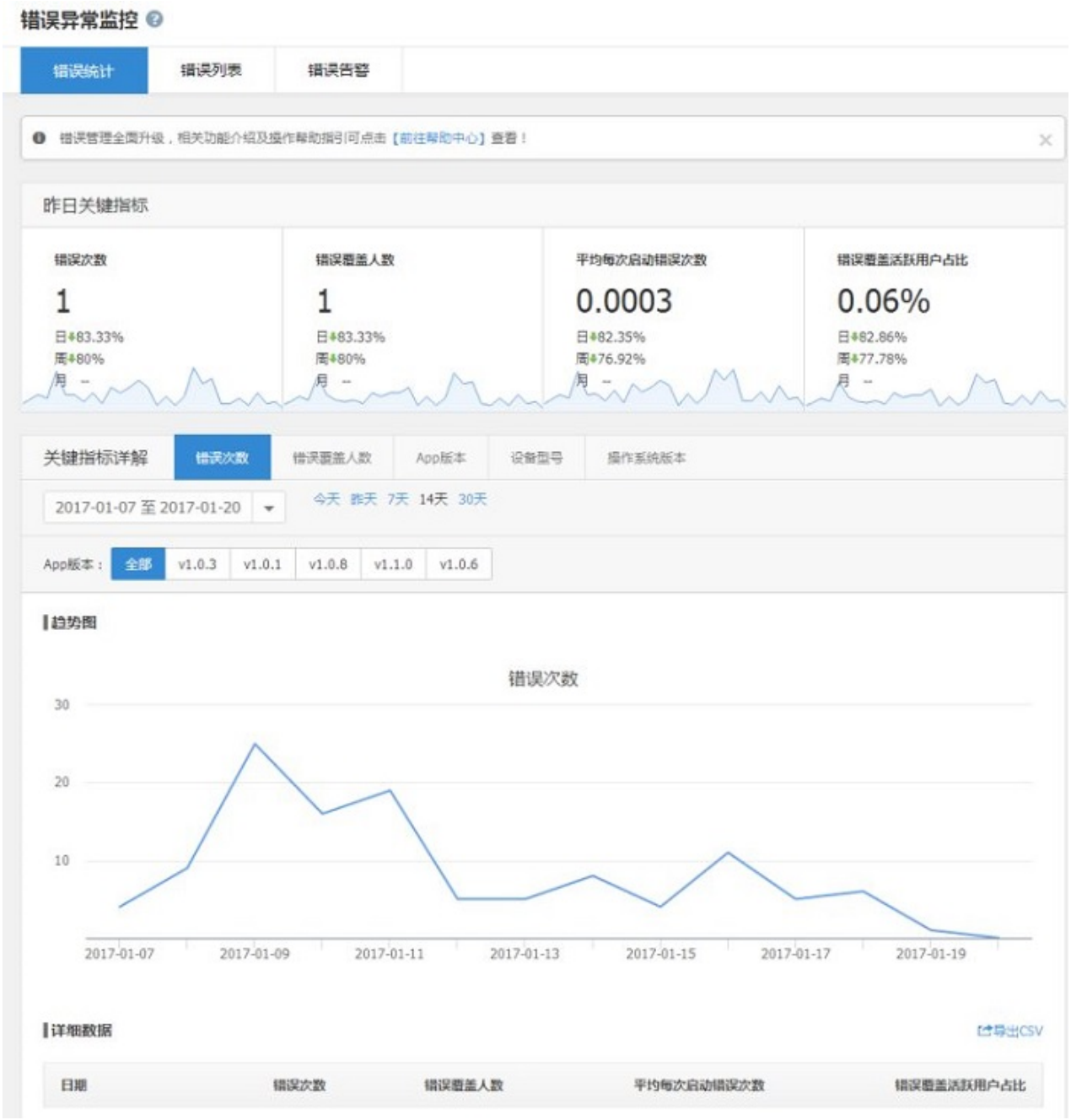
上报自有账号：

[MTA setAccount:@"其它账号" type:AT_OTH];

错误分析

功能介绍

1. 通过错误统计，运营人员可以知道 App 的质量情况。



2. 通过错误列表，开发者可以查看发生错误时的堆栈信息，为定位问题修复 BUG 提供有力的保证。

3. 通过错误告警，App 若发生重大的质量问题，将告警给相关人员，让相关人员做出相应的应急处理。

Android 错误统计

概览

本指南用于 MTA Android Crash V2 版本的接入，用于引导 crash 模块的配置升级、API 调用。

V2 版本的 Crash 模块已全新重构升级，支持 Java 和所有架构的 Native so

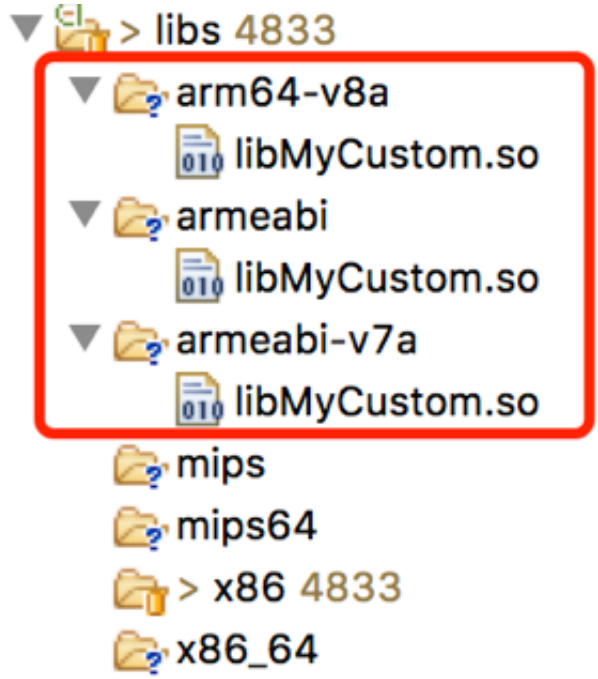
丰富的异常数据采集、完整的堆栈还原、实时堆栈展示报表和实时监控告警等一系列特性。

配置升级

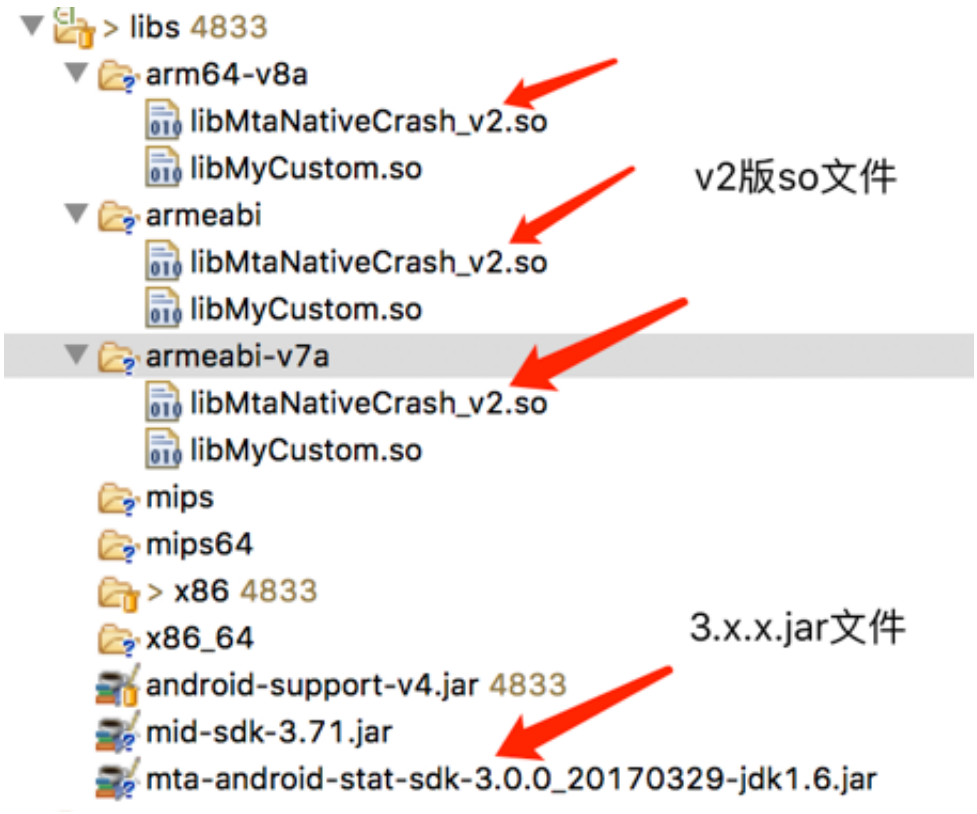
库文件

MTA 支持 Java 和 Native 异常捕获，其中 Java Crash 模块默认集成在 MTA 主体 jar 包中，Native Crash（即 c/c++ 或 so 的异常捕获）需要额外添加 so 文件并调用 API 启用，若您的工程涉及到 Native 编码，建议打开上 Native Crash 模块，否则不需要额外添加这部分的文件。
Java Crash：使用新的 mta 3.x.x.jar 替换旧版。
Native Crash：libMtaNativeCrash_v2.so，删除旧的 ibMtaNativeCrash.so，需要注意不同 ABI 的文件存放，具体见下。
MTA Native Crash 支持当前 Android 系统所支持的所有架构：armeabi、armeabi-v7a、arm64–v8a、x86、x86_64、mips、mips64。

在集成过程中，一定要注意不同架构 so
库文件的存放，否则可能会引起问题，总的原则是只保留工程所支持的架构 so，不支持的 so
千万不要额外导入，下面举例说明：
集成前：假设 libMyCustom.so 为实际工程的 so 文件，所支持架构列表只有 armeabi、armeabi-v7a、arm64-v8a 三种。



集成后：根据只保留工程所支持 so 架构的原则，那么只需要把 Mta Native Crash 对应的架构下的 so 文件复制到工程对应的目录中即可，即 MTA armeabi的libMtaNativeCrash_v2.so 复制到工程对应的 armeabi 目录下，注意一定要匹配，保持支持的架构目录的 so
文件一致，而不支持的架构不需要添加，一个集成带 native crash 模块的 MTA 后的工程示例见下图：



工程配置

主要的配置参考 MTA 的接入配置功能，下面列举 AndroidMenifest.xml 文件的配置部分。

```
<application>
……工程application指标的的其它配置

<!-- MTA配置项 < -->
<!-- 请将value改为MTA分配的appkey，若已通过API调用可跳过本配置 < -->
<meta-data
android:name="TA_APPKEY"
android:value="A91LM44JGFLV" />

<!-- 请将value改为APP的发布渠道（市场），若已通过API调用可跳过本配置 < -->
<meta-data
android:name="InstallChannel"
android:value="应用宝" />
```

```
<!-- MID 3.x版的配置部分，需要将"您的APP包名"替换成工程实际的包名 < -->
<provider
android:name="com.tencent.mid.api.MidProvider"
android:authorities="您的APP包名.TENCENT.MID.V3"
android:exported="true" >
</provider>
</application>

<!-- MTA权限配置项 -->
<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" />
```

API 接口

为了方便使用，最新的 Crash 模块统一封装成类 StatCrashReporter
对外提供服务，同时兼容旧的接口。为了方便管理，建议以新的 API 调用替换旧的接口，具体的 API 升级见下文。

Java Crash 异常捕获

void setJavaCrashHandlerStatus (boolean enable)

1. 说明
   开启或禁用 java 异常捕获，初始化不会带来任何的流量和性能消耗，生效后会注册 DefaultUncaughtExceptionHandler，crash 时捕获相关信息存储在本地并上报，可通过添加 StatCrashCallback 监听 Crash 发生。
2. 参数
   enable：是否开启java异常捕获，默认开启；
   true：开启；
   false：禁用；
   对应的 get 接口：

   getJavaCrashHandlerStatus()

   ；
   调用位置：App 初始化时调用，比如 Application.onCreate 或 MainActivity.onCreate。
3. 示例StatCrashReporter.getStatCrashReporter(getApplicationContext())
   .setJavaCrashHandlerStatus(true);

Native Crash 异常捕获

void setJniNativeCrashStatus (boolean enable)

1. 说明
   开启或禁用 Native 异步捕获，初始化不会带来任何的流量和性能消耗。生效后会注册 signal 到 native 层，crash 时捕获相关信息存储在本地并上报，可通过添加 StatCrashCallback 监听 Crash 发生。
2. 参数
   enable：是否开启 Native 异常捕获，默认为 false；
   true：开启；
   false：禁用；
   对应的 get 接口：

   getJniNativeCrashStatus ()

   ；
   调用位置：App 初始化时调用，比如 Application.onCreate 或 MainActivity.onCreate。
3. 示例StatCrashReporter.getStatCrashReporter(getApplicationContext())
   .setJniNativeCrashStatus(true);

监听 Crash 发生

void addCrashCallback(StatCrashCallback cb)

1. 说明
   添加 StatCrashCallback 监听 Java 或 Native 的 Crash 发生。
2. 参数
   StatCrashCallback crash 发生时的 StatCrashCallback 回调。public interface StatCrashCallback {
   // thread：crash的线程信息
   // throwable：crash的堆栈信息
   public abstract void onJavaCrash(Thread thread, Throwable throwable);
   // nativeCrashStacks：native crash的tombstone格式文件
   public abstract void onJniNativeCrash(String nativeCrashStacks);
   }

对应的 remove 接口：

removeCrashCallback(StatCrashCallback cb)

；

调用位置：App 初始化时调用，比如 Application.onCreate 或 MainActivity.onCreate。

3. 示例StatCrashReporter.getStatCrashReporter(getApplicationContext()).addCrashCallback(
new StatCrashCallback() {
@Override
public void onJniNativeCrash(String nativeCrashStacks) { // native crash happened
// do something
}
@Override
public void onJavaCrash(Thread thread, Throwable ex) {// java crash happened
// do something
}
});

上报策略

Crash 产生时，默认下次 App 启动时初始化 MTA 后的 3 秒开始上报，可通过 setReportDelaySecOnStart(int reportDelaySecOnStart) ，reportDelaySecOnStart 的单位为秒，范围 [0, 10*60]，也可以通过设置 setEnableInstantReporting 设置实时上报，即 crash 时有网络的条件下尽量立即上报，若上报失败，则下次启动时上报。

为方便开发者调试，在 Debug 模式，即 StatConfig.setDebugEnable(true)，采用实时上报策略。

多进程环境

在多进程环境中，若需要在多个进程捕获异常，需要在每个进程都初始化 MTA 或 Native Crash 接口，建议在

Application.onCreate 进行。

一个完整的示例

示例包括以上主要 API 调用，请根据需要自行决定调用哪些 API。

```
// 设置appkey，应用的唯一标识，在MTA官网申请得到，也可通过Manifest配置
// 记得把以下appkey替换成自己的
StatConfig.setAppKey(app, "A91LM44JGFLV");
// 设置投放渠道，即应用市场，也可通过Manifest配置
StatConfig.setInstallChannel(app, "应用宝");
StatService.setContext(app);
// 这个是开启Mta的统计功能
StatService.registerActivityLifecycleCallbacks(app);

StatCrashReporter crashReporter = StatCrashReporter.getStatCrashReporter(app);
// 开启异常时的实时上报
crashReporter.setEnableInstantReporting(true);
// 开启java异常捕获
crashReporter.setJavaCrashHandlerStatus(true);
// 开启Native c/c++，即so的异常捕获
// 请根据需要添加，记得so文件
crashReporter.setJniNativeCrashStatus(true);

// crash时的回调，业务可根据需要自选决定是否添加
crashReporter.addCrashCallback(new StatCrashCallback() {

@Override
public void onJniNativeCrash(String tombstoneString) {
// native dump内容，包含异常信号、进程、线程、寄存器、堆栈等信息
// 具体请参考：Android原生的tombstone文件格式
log("MTA StatCrashCallback onJniNativeCrash:\n" + tombstoneString);
}
```

```
@Override
public void onJavaCrash(Thread thread, Throwable ex) {
//thread:crash线程信息
// ex:crash堆栈
log("MTA StatCrashCallback onJavaCrash:\n", ex);
}
});
```

Android Crash V2 版本符号表上传说明

用于 MTA Android Crash V 2版本符号表上传说明，主要分为 Java 符号表和 Native（C/C++）符号表两部分，用于系统还原混淆后的堆栈信息。

Java 符号表

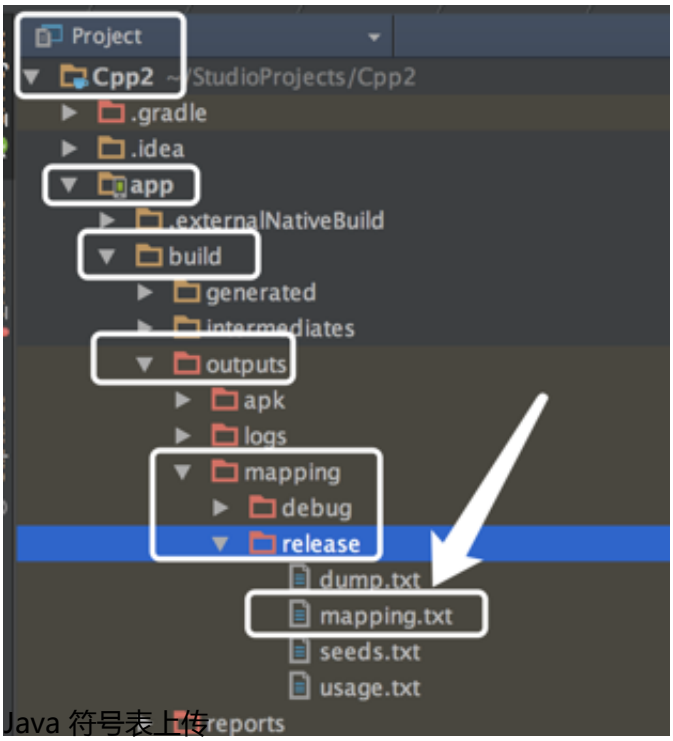
1. Eclipse

   通常符号表名为“mapping.txt”，位于 proguard 目录下，如果是使用 ant 脚本编译，则在脚本指定的目录下。
2. Android Studio

   在 build.gradle 文件中开启混淆代码，minifyEnabled 设置为 true 时生成 mapping 文件，路径一般为工程目录下的

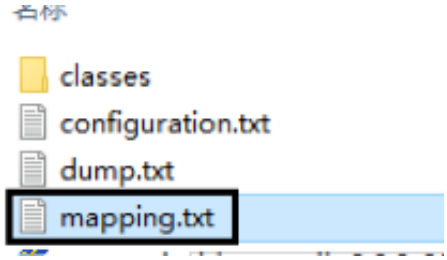
   build/outputs/mapping/release/mapping.txt

```
buildTypes {
    debug {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
    }
    release {
        minifyEnabled true
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
    }
}
externalNativeBuild {
    cmake {
        path "CMakeLists.txt"
    }
}
```



3. Java 符号表上传

   在前台打开符号表上传页面，填写 App 版本号，选择【Java】文件类型，单击【选择文件】，选取 mapping.txt 文件后单击上传即可。



Native 符号表

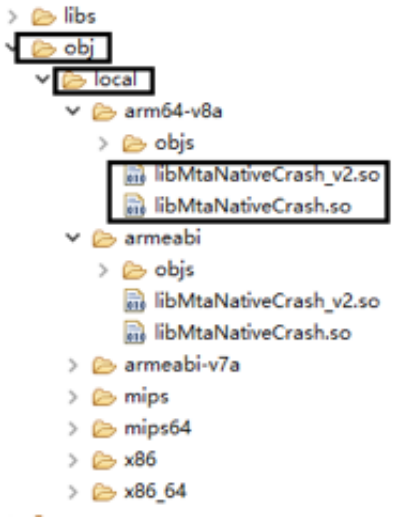
Native 符号表是为了找回 so 文件 Crash 堆栈还原使用的，由于编译器的问题，发布的 so 文件是已去符号化的，而编译过程中产生的中间文件才是带符号表信息的，因此建议大家每次构建版本时备份好 debug so 文件。

1. Eclipse

   使用 eclipse 构建的 so 文件，debug so 位于项目的

   /obj/local/xxxeabi/

   目录下，其中 xxxeabi 为具体的架构信息，见下图：



我们需要把 local 目录打包，建议打包前把架构下的 Objs 目录清除掉。

2. Android Studio

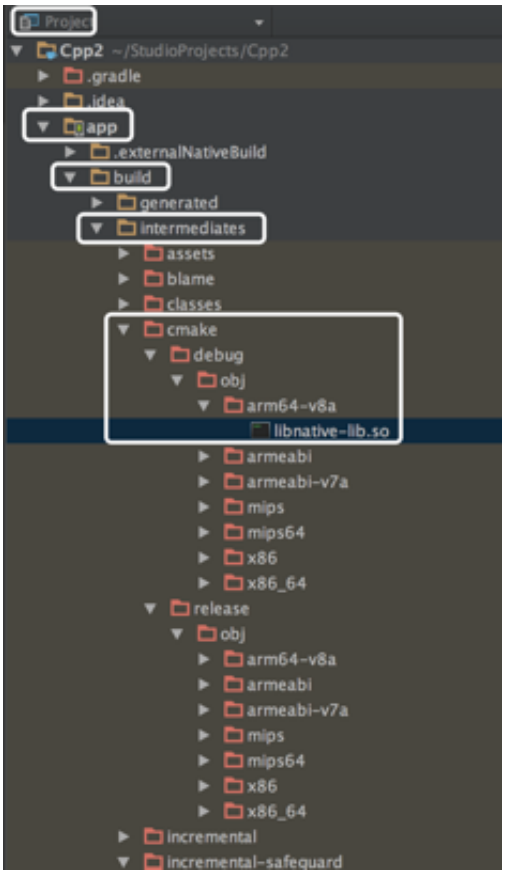
使用 Android Studio 编译的 so 文件，分为 Debug 和 Release 两个不同的版本，对应的目录通常为：

/<Module>/build/intermediates/cmake/debug/obj/xxxeabi/

和

/<Module>/build/intermediates/cmake/release/obj/xxxeabi/

，我们需要将 obj 目录整体打包成 zip 格式。

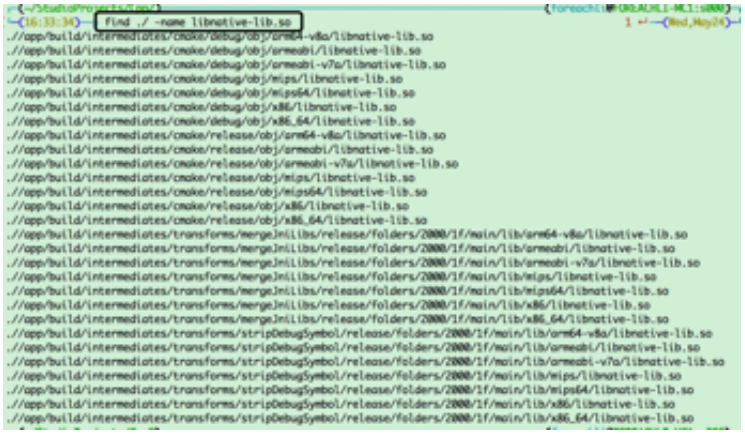


不同的 Android studio 版本可能存在路径差异，可以使用以下方法来判断，以 Linux/Mac OS 系统为例：

i. 打开终端命令行，cd 到当前工程目录，输入

find ./ -name your-lib-name.so

，找到 so 编译生成目录

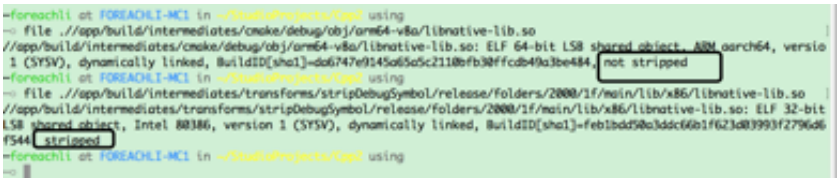


ii. 输入

file your-full-so-path

，如果输出【not stripped】表示是带符号信息的 so

文件，输出【stripped】表示是删除符号表信息的 so 文件。



3. Native 符号表上传

在前台打开符号表上传页面，填写 App

版本号，选择【native】文件类型，单击【选择文件】，选取刚刚打包的 zip 文件后单击上传即可。

iOS 错误统计

逻辑错误统计接口

错误统计既可统计 App 的 crash，也可统计 App 中的逻辑错误，crash 由 MTA
自动捕获并且上报，无需调用额外接口。逻辑错误需要开发者手动调用相关接口上报。

```
/**
统计程序逻辑错误
逻辑错误只有描述，没有堆栈信息

@param error 错误描述
*/
+ (void)trackError:(NSString *)error;

/**
统计程序逻辑错误
并且指定上报方式
逻辑错误只有描述，没有堆栈信息

@param error 错误描述
@param appkey 若此参数不为nil，则上报到此appkey。否则，上报到startWithAppkey中传入的appkey
@param isRealTime 是否实时上报，若传入YES，则忽略全局上报策略实时上报。否则按照全局策略上报。
*/
+ (void)trackError:(NSString *)error appkey:(NSString *)appkey isRealTime:(BOOL)isRealTime;

/**
统计异常
异常信息包括了异常的原因和堆栈

@param exception 异常信息
*/
+ (void)trackException:(NSException *)exception;

/**
统计异常
```

并且指定上报方式
异常信息包括了异常的原因和堆栈

@param exception 异常信息
@param appkey 若此参数不为nil，则上报到此appkey。否则，上报到startWithAppkey中传入的appkey
@param isRealTime 是否实时上报，若传入YES，则忽略全局上报策略实时上报。否则按照全局策略上报。
*/
+ (void)trackException:(NSException *)exception appkey:(NSString *)appkey
isRealTime:(BOOL)isRealTime;

崩溃上线文信息统计接口

在崩溃发生时候，MTA 会自动捕获崩溃的堆栈以及基本的上下文信息，并且在下次启动时候上报。除此之外，开发者还可以调用特定的 API
来储存额外的上下文信息，这些信息会在崩溃发生时跟随崩溃报告一起上报，以便 Debug。

```
/**
设置自定义的tag
崩溃发生时，会将已经设置的tag上报，以便定位问题

@param tagKey tag的Key，若key已经设置，则新的value会覆盖旧的value
@param tagValue tag的value
*/
+ (void)setCustomTag:(NSString *)tagKey value:(NSString *)tagValue;


/**
输出诊断信息
崩溃发生时，会将最后输出的50条诊断信息上报，以便定位问题

@param log 诊断信息
*/
+ (void)traceLog:(NSString *)log;
```

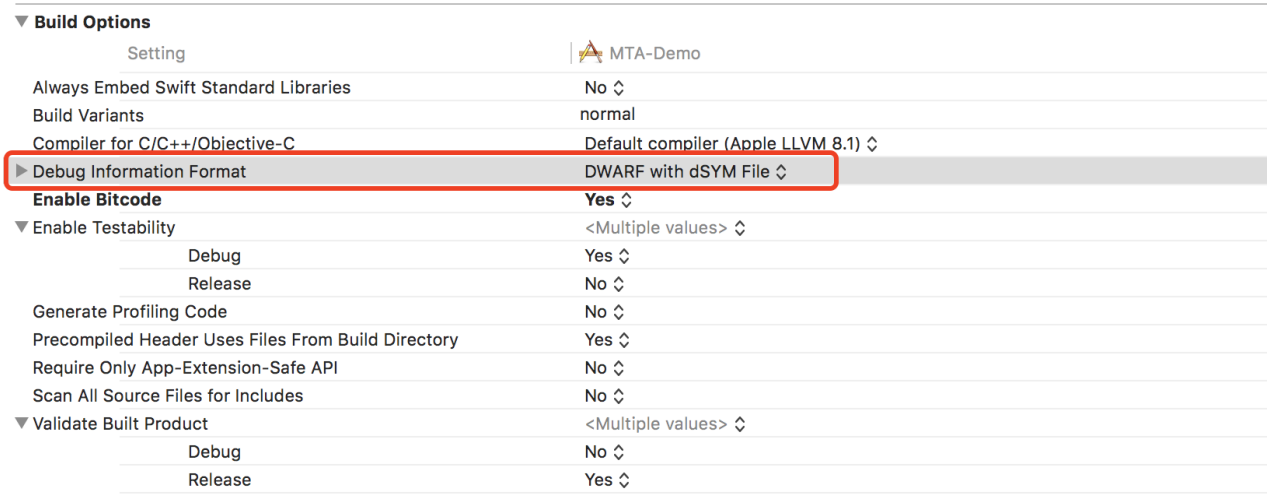
符号表上传指南

异常上报使用说明

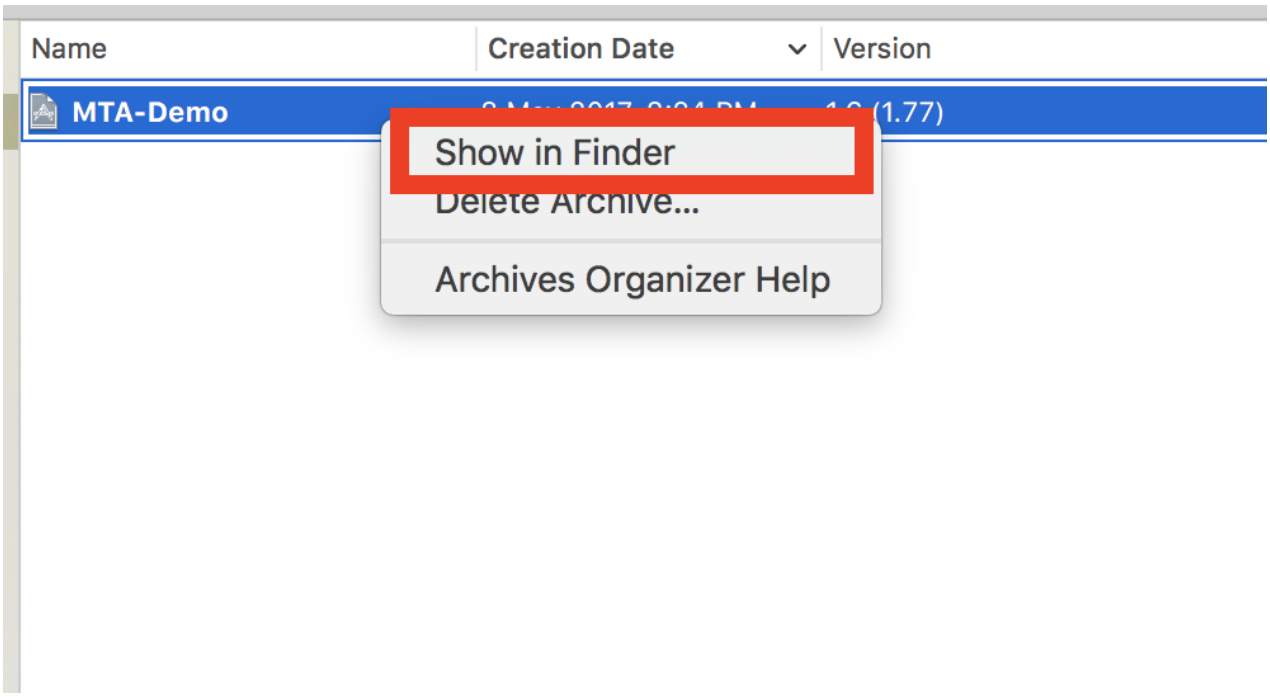
新版 MTA 增强了异常上报功能，在 App 发生异常时 MTA 会采集最近的 NSLog、页面的信息以及崩溃堆栈进行上报。除此之外，用户还可以根据需要，自行上报业务相关的 tag 和诊断信息，以便 debug，上报诊断信息的接口在 MTACrashReporter.h 头文件中，使用方法请看头文件注释。

除此之外，为了更准确的还原崩溃堆栈，需要用户提供 App 对应的 dSYM 文件，制作 dSYM 的步骤如下。

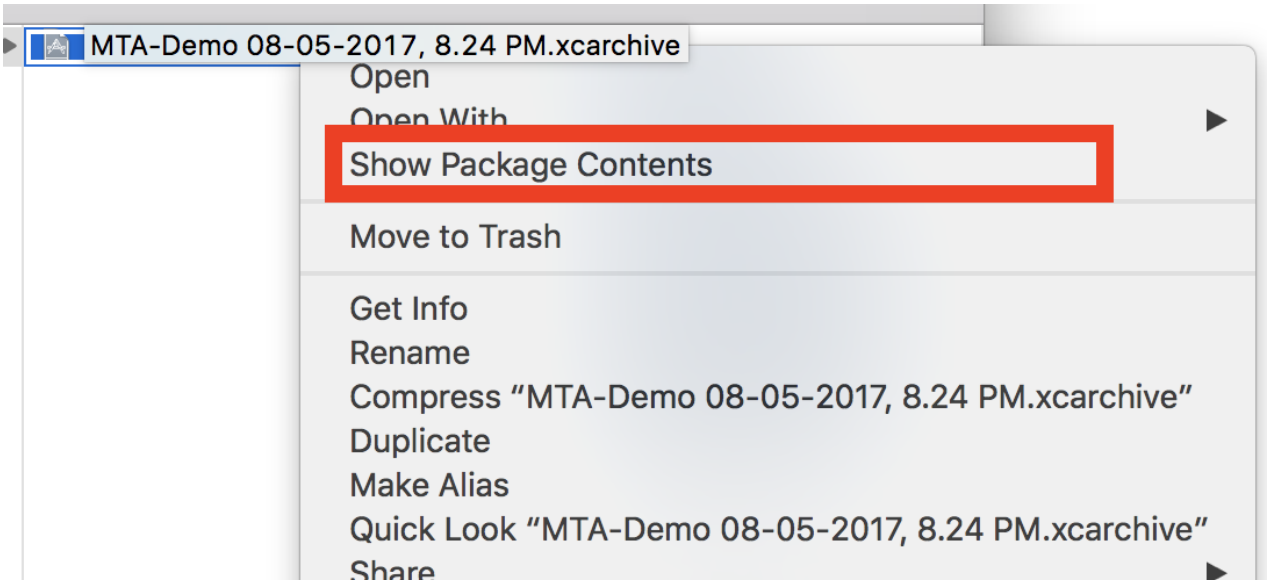
1. 在 Xcode 工程配置中找到【Debug Information Format】项，将该项的值改为【DWARF with dSYM File】，如图所示：



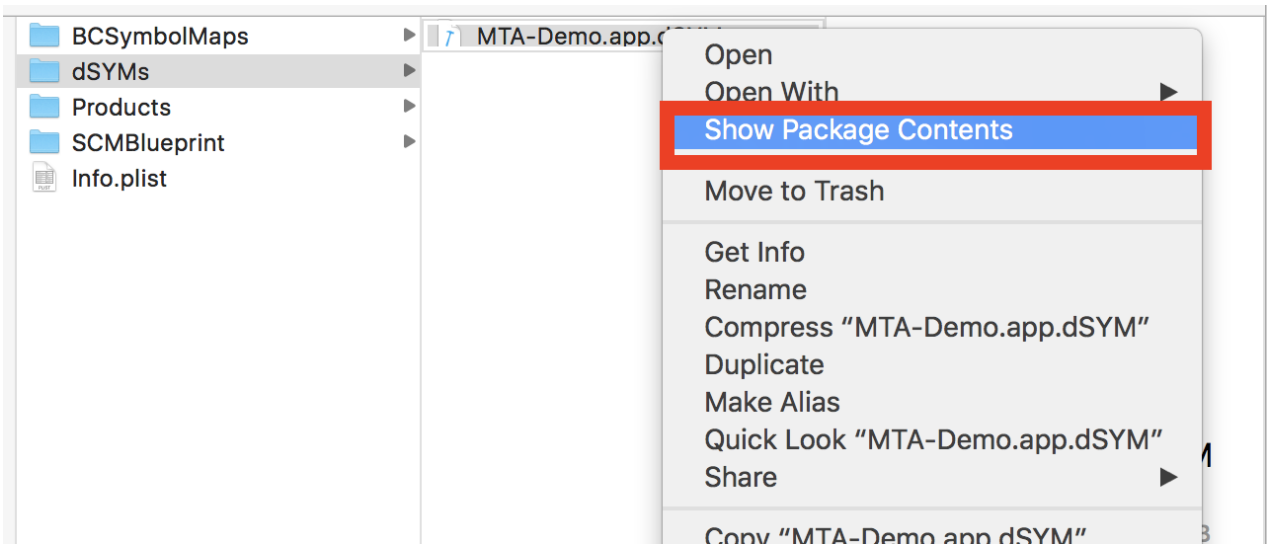
2. 按照正常步骤打包，打包完成后在弹出的页面单击右键，在弹出的菜单中选择【Show in Finder】，如图所示：



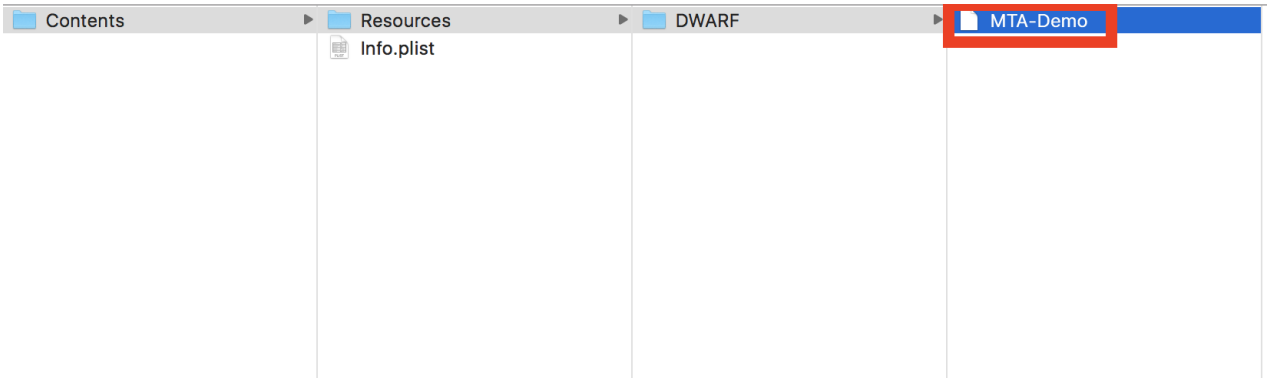
3. 在弹出的 Finder 页面中可以看到上一步生成的 xcarchive 文件，右键单击文件，选择【Show Package Contents】，如图所示：



4. 找到 dSYMs 目录内的 dSYM 文件，右键单击，选择【Show Package Contents】，如图所示：



5. 找到包内 Contents/Resources/DWARF 目录下的文件，上传这个文件即可。



找到 dSYM 文件后使用上传工具或者在网页控制台将 dSYM 文件上传即可。

iOS Crash 错误分析上传工具

iOS Crash 符号表上传工具主要用来上传 App 的符号表文件，即 dSYM 文件，以便网页端显示 App crash 的堆栈信息还原出符号，让开发者更直接的看到程序Crash发生的位置和原因，您可以单击 下载
或是在错误管理窗口下载，用法如下图：

关于工具的参数说明：

QQ 账号：MTA 前端网页中 App 的管理员；

App ID：MTA 前端【应用管理】选项卡中 App ID；

App Key：MTA前端【应用管理】选项卡中 App KEY；

dSYM：编译 App 时生成的符号表文件。

注意：

一般在【Xcode】>【Target】>【BuildSetting】>【Build Options】选项卡中的 DebugInformation 可以看到编译 App 生成的符号表文件，一般符号表文件的位置与生成的 App 在同一个位置。

用户画像

用户画像的价值

基于腾讯8亿QQ用户，形成完善可依赖的用户画像体系，包括但不限于年龄、性别、地域、学历、设备、应用偏好等，全方位一体化刻画用户，保证精准性和覆盖度，有效解决业务中冷启动、画像稀疏等各类问题，帮助开发者清晰洞察用户属性。

用户画像的价值

从用户出发设计产品

查看用户画像，有利于在产品设计的过程中抛开个人喜好，将焦点关注在目标用户的动机和行为上进行产品设计，提升用户体验。

精准化营销

用户的消费偏好差异很大，不同用户对于同一品类的选择有着不同的要求，甚至于选择同一品牌同一产品的用户也可能是因为不同的原因而购买，了解偏好需求，根据用户的需求特点将用户归类，不同类型的用户决策模式容易识别且变化比较小，有针对性的进行营销，有效提升营销效果，解决营销成本。

数据应用

针对用户推荐用户喜欢的产品，如猜你喜欢等，为用户提供个性化的服务。

提升产品价值

根据目标用户的特点，在用户偏好的渠道、产品版本上与其交互，促成增值服务，实现精准运营和营销。

用户画像使用指南

Android 用户画像使用指南

新版本 Android SDK 不需要用户进行配置，由 SDK 自主进行数据上报。

iOS 用户画像使用指南

iOS 用户画像数据获取依赖 QQ 账号和 idfa
数据上报，开发者可任意选择上报其一，为了保证画像数据准确性，建议开发者同时上报 QQ 账号和 idfa 数据。
1.上报 QQ 账号

```
/**
 上报账号对应实时数据中的新增账号数字段
 如果上报的账号类型是QQ号，还能同时激活用户画像功能

 @param account 账号名
 @param accountType 账号类型
 */
+ (void)setAccount:(NSString *)account type:(MTAAccountType)accountType;
```

示例

```
[MTA setAccount:@"991145990" type:AT_QQ];
```

2.上报 idfa

```
/**
 设备的idfa，建议有广告权限的app设置此字段，设置以后也可激活用户画像功能
 默认为空
 */
@property (nonatomic, copy) NSString *ifa;
```

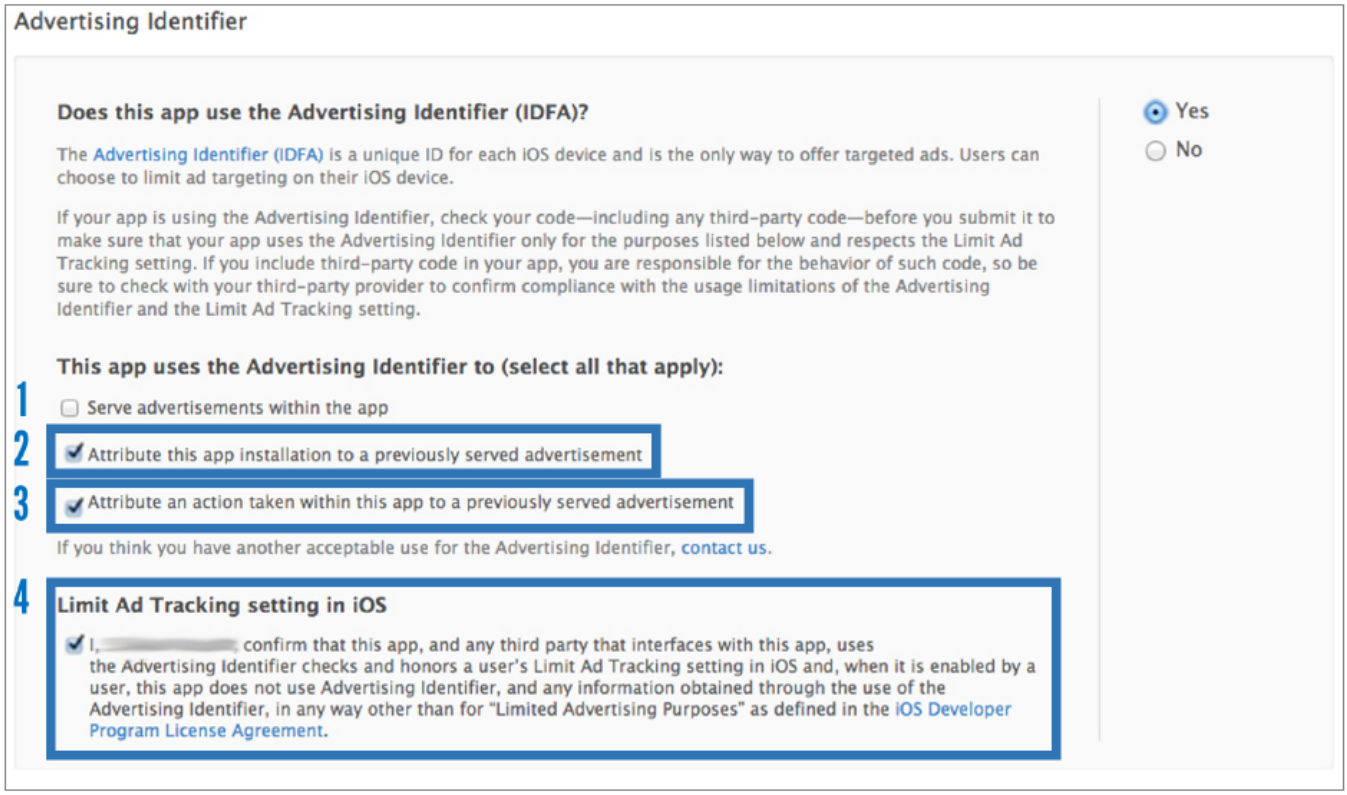
示例

[[MTAConfig getInstance] setIfa:@"yourIDFA"];

上传 IDFA 注意事项

使用 IDFA 原则上需要集成任意一家的广告 SDK，如果用户期望采集 IDFA

但是并未使用任何广告，可以采用一下方法通过 Appstore 审核。



其中图片的各个选项内容依次如下展示：

1.serve advertisements within the app

应用内广告服务，适用于应用内集成了广告的场景，如果您的情况符合，需要勾选此选项。

2.Attribute this app installation to a previously served advertisement.

用于跟踪和统计广告带来的安装量，需要勾选。

3.Attribute an action taken within this app to a previously served advertisement

用于跟踪和统计广告安装后带来的用户行为，需要勾选。

4.Limit Ad Tracking setting in iOS

此项属于确认项，需要勾选。

可视化埋点

功能说明

可视化埋点，主要适用于对组件和按钮埋点的场景，通过在腾讯移动分析（简称 MTA）管理台的快速操作，跳过部署代码和发版的过程，省去代码埋点的人力和时间成本，大大提高运营效率。

Android代码集成

接入指引

1. 确保正确接入普通版本 MTA；

   注意：
   后续如果要接可视化，必须在 AndroidManifest.xml 里面设置
   TA_APPKEY，不可以代码动态设置。

2. 添加 track-sdk_v3.2.1.jar，libthrift-0.10.0.jar，slf4j-api-1.7.24.jar 包；

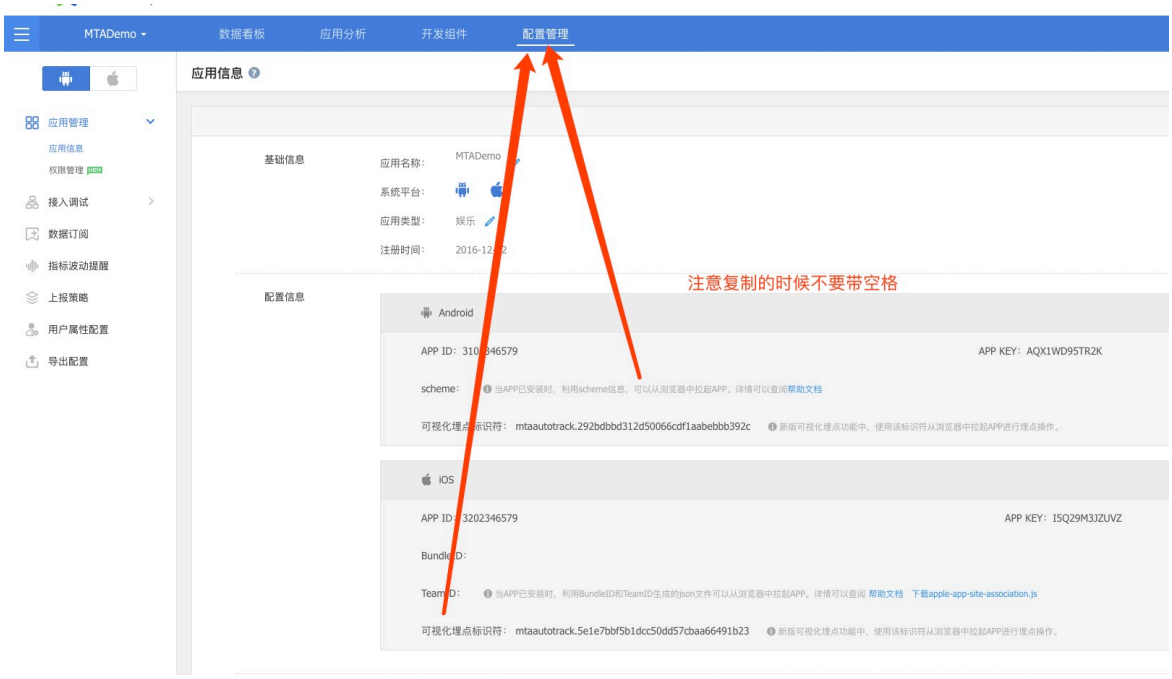
3. 添加URL Scheme。
   i.MTA 官网应用管理页面将 scheme 里面的内容拷贝出来；
   ii. 将下面的启动接口添加到您的 AndroidManifest.xml 中的 LAUNCHER Activity
   下以便我们唤醒您的程序。

```
<activity
android:name=".LauncherActivity"
android:label="@string/app_name"
android:theme="@style/AppTheme.NoActionBar">
<intent-filter>
  <action android:name="android.intent.action.MAIN"/>
  <category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
<!-- MTA可视化启动连接接口 -->
<intent-filter>
  <data android:scheme="配置管理页面中的shceme"/>
  <action android:name="android.intent.action.VIEW"/>
  <category android:name="android.intent.category.DEFAULT"/>
  <category android:name="android.intent.category.BROWSABLE"/>
</intent-filter>
</activity>
```

请添加一整个 intent-filter 区块，并确保其中只有一个 data 字段。

注意：

配置 Scheme 的时候不要复制空格，否则会链接不成功。

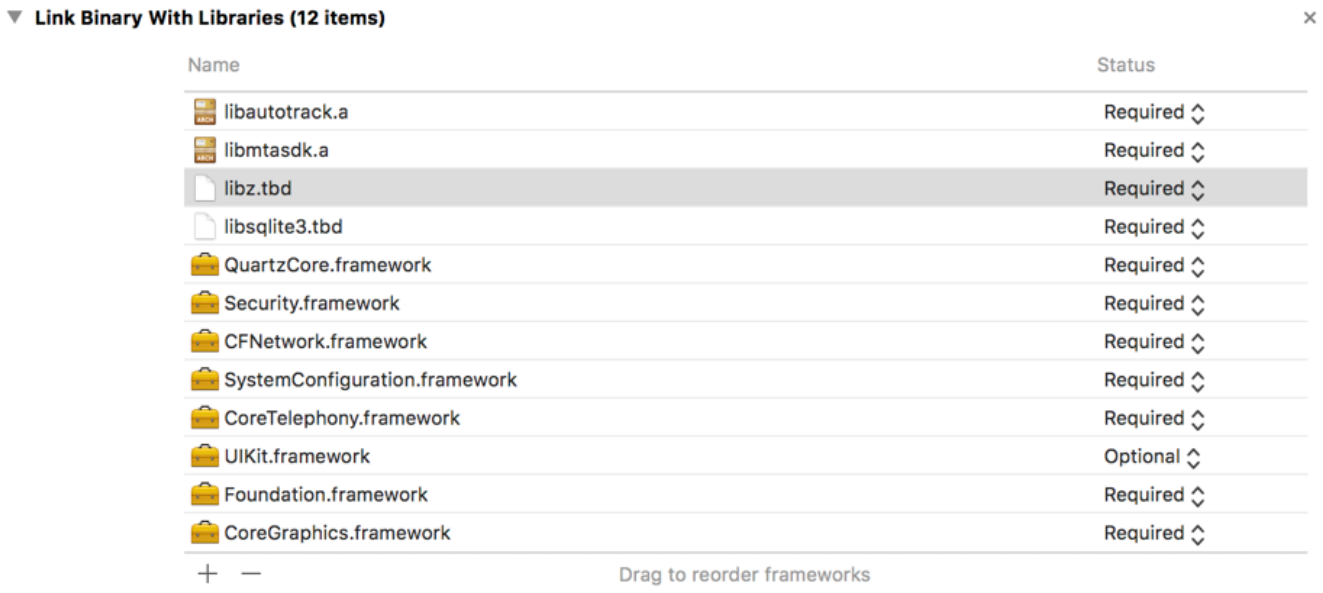


4. 初始化SDK ，在应用 Application 的 onCreat 里面添加 StatisticsDataAPI.instance(this)。

iOS代码集成

接入方法

1.将 libautotrack.a 链接到工程中，如下图：



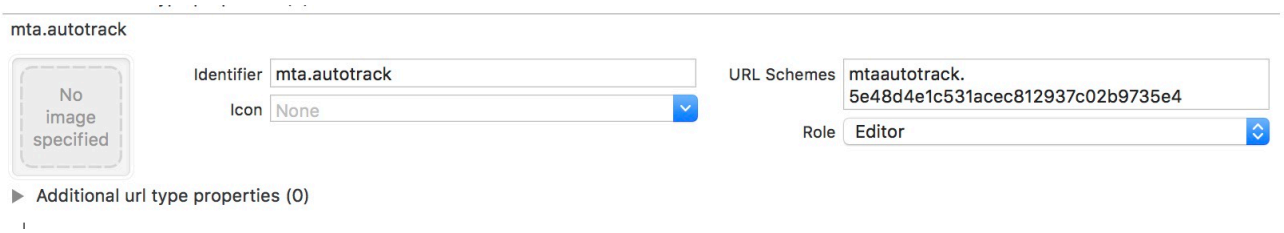
2.前往 MTA 前台的应用配置页面查看可视化埋点标识符。

3.在工程配中添加 URL Types。

Identifier: <随意填写>

URL Schemes: 第二步中的标识符

Role: <默认值>



4.参考 MTADemo，在 AppDelegate 的方法中添加初始化可视化埋点的代码。

```
- (BOOL)application:(UIApplication *)app
  openURL:(NSURL *)url
  options:(NSDictionary<UIApplicationOpenURLOptionsKey,id> *)options;



- (BOOL)application:(UIApplication *)app
  openURL:(NSURL *)url
  options:(NSDictionary<UIApplicationOpenURLOptionsKey,id> *)options {

  // 可视化埋点代码
  if ([MTAAutoTrack handleAutoTrackURL:url])
  return YES;



  // 原有代码
  // ...
  return NO;
}
```

5.前往腾讯 移动分析官网
，进入对应应用，在左侧进入可视化埋点分析页面，按照网页提示进行操作，如图所示：


数据概览
实时数据
历史趋势
事件分析
事件列表
可视化埋点
NEW
漏斗模型

操作指南

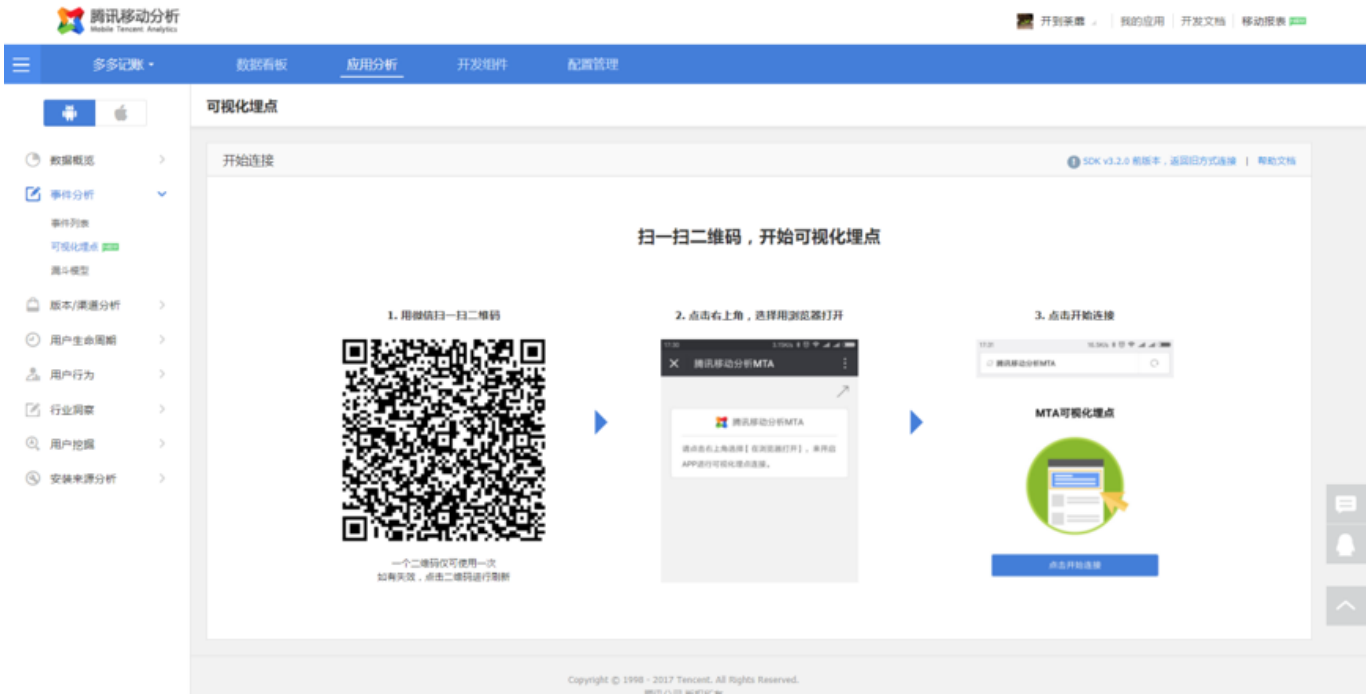
提前条件

1. 应用 App 集成 MTA 最新的统计 SDK（说明：V3.2.0 开始版本即新版，支持微信扫描连接，V3.2.0 之前版本即旧版，支持 4 指触摸和摇一摇连接）；
2. 在 MTA 管理台，成功注册自己应用 App。

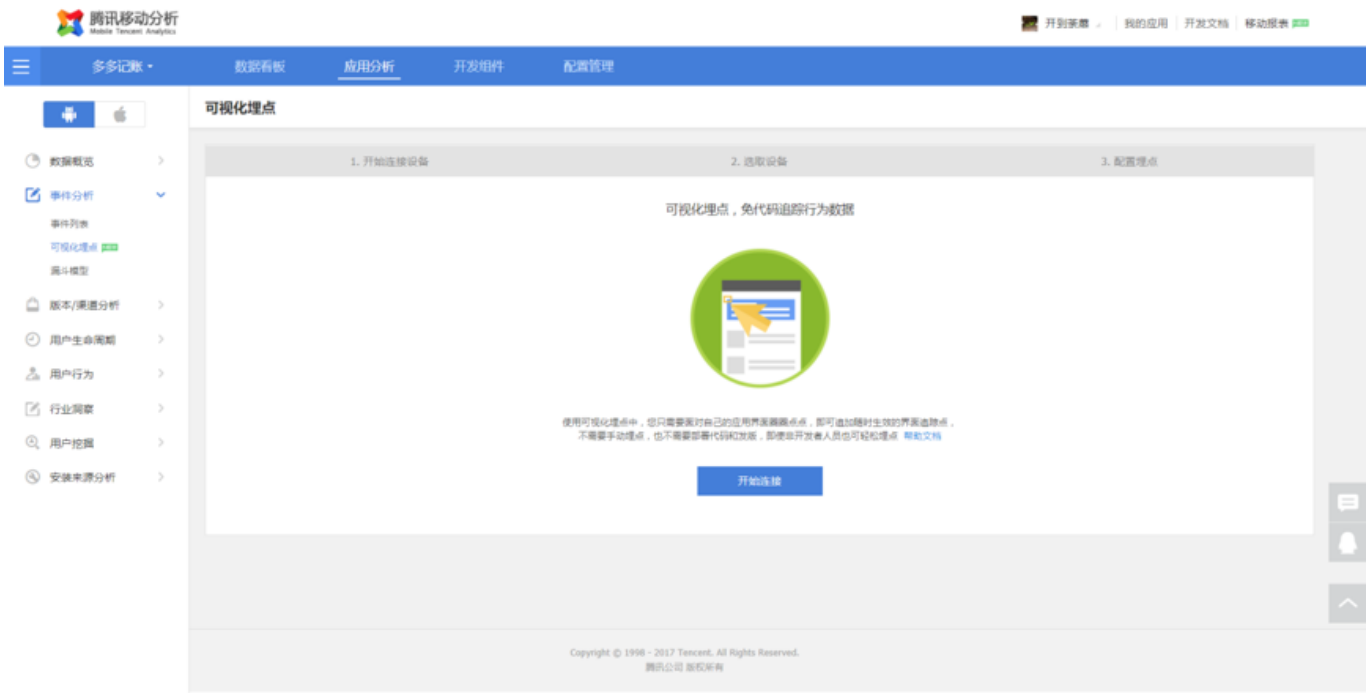
操作步骤

第一步：登录 MTA 管理台，进入【可视化埋点】功能页面；

新版：微信扫码连接页面



旧版：4 指触摸和摇一摇连接页面



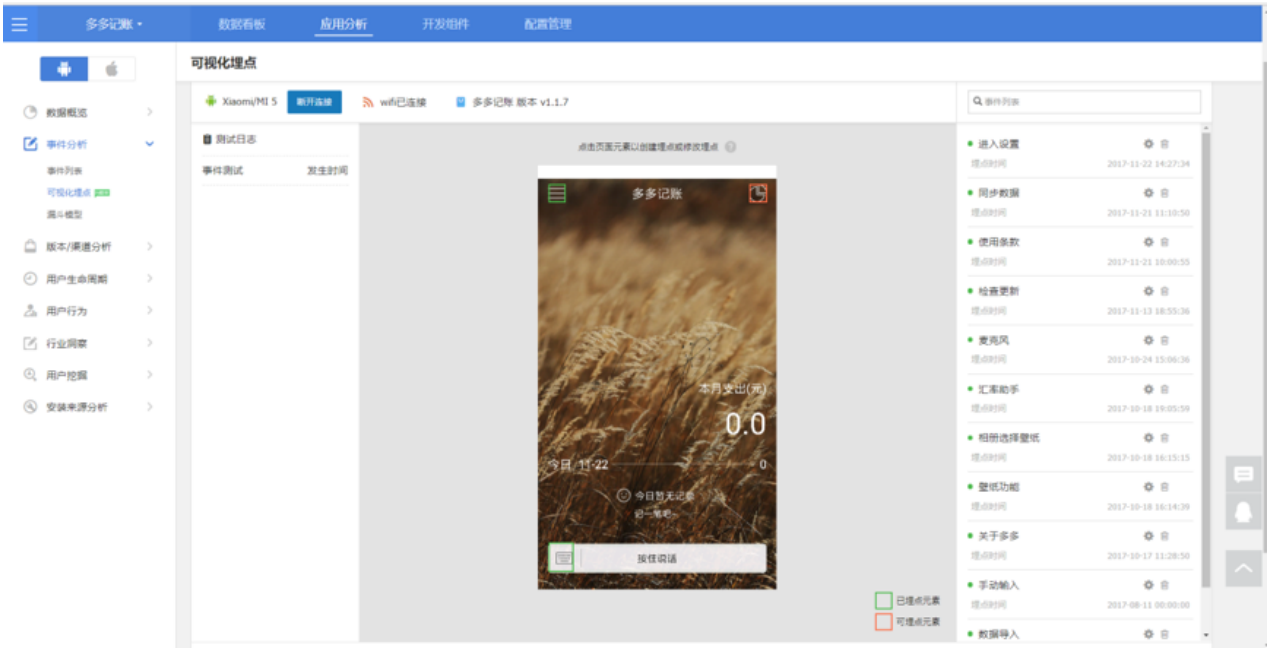
第二步：管理台连接设备；

新版：微信扫描连接

1. 手机微信扫描二维码，进入如下页面，提示浏览器打开链接

2. 手机浏览器打开后，进入如下页面，单击【开始连接】，会拉起待埋点的应用 App

3. 等待几秒钟，PC 端管理台进入如下页面，即为连接成功



旧版：4 指触摸和摇一摇

单击【开始连接设备】，会出现下面【尝试连接中】的界面，这时您需如下操作：

1. 在手机上打开需要进行埋点的 App；
2. 在 10 秒内，将 4 根手指微微张开，按在屏幕上（请不要急于在启动界面按下）；
3. 4 根手指在屏幕上至少保持 3 秒。

若连接过程中，出现下面的图片，说明有一台手机已经与 MTA 连接，请检查该手机是否为您自己的手

机，如果是请单击【连接】，如果不是请单击【取消】，并重复上步操作进行连接。

注意：

1.使用前，请确认您的 SDK 版本号高于或等于 1.3.0；

2.可视化埋点产生的实时数据可能消耗流量，请在 WiFi 环境下调试；

3.如果始终无法连接，请强行退出进程（Android、iOS 都需要强退）后重启App 再次进行尝试；

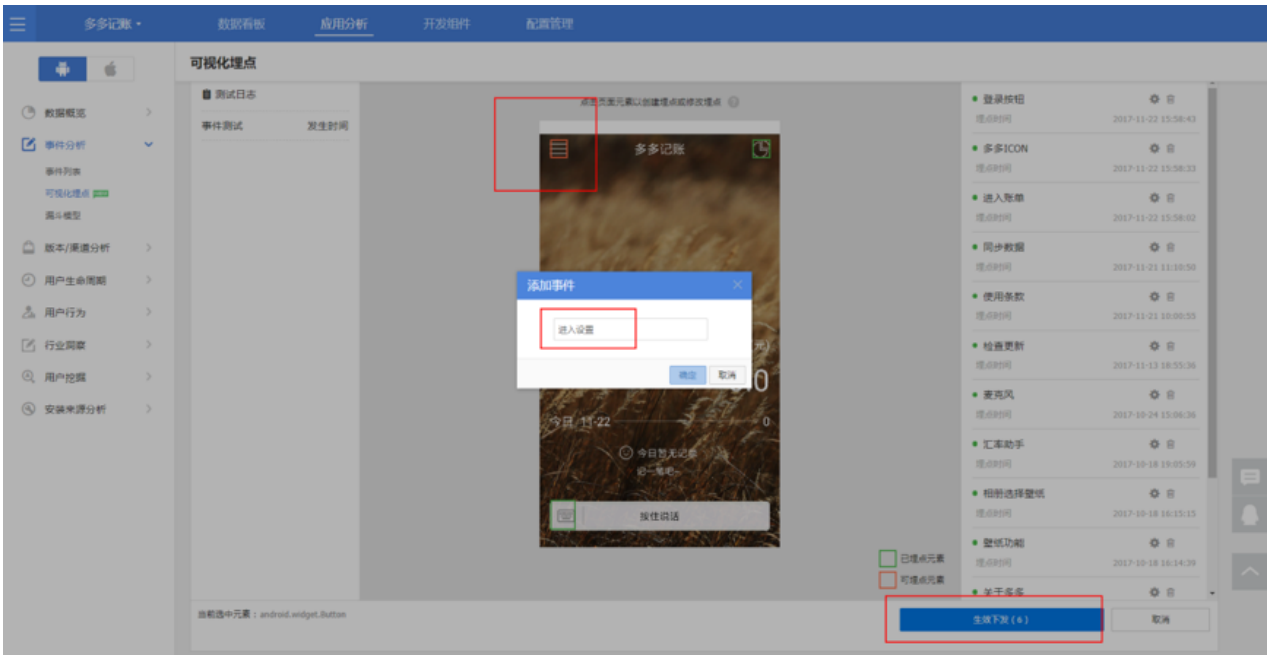
4.若项目中使用了 TencentOpenApi，需要把 TencentOpenApi 更新到新版本。

第三步：进行埋点操作

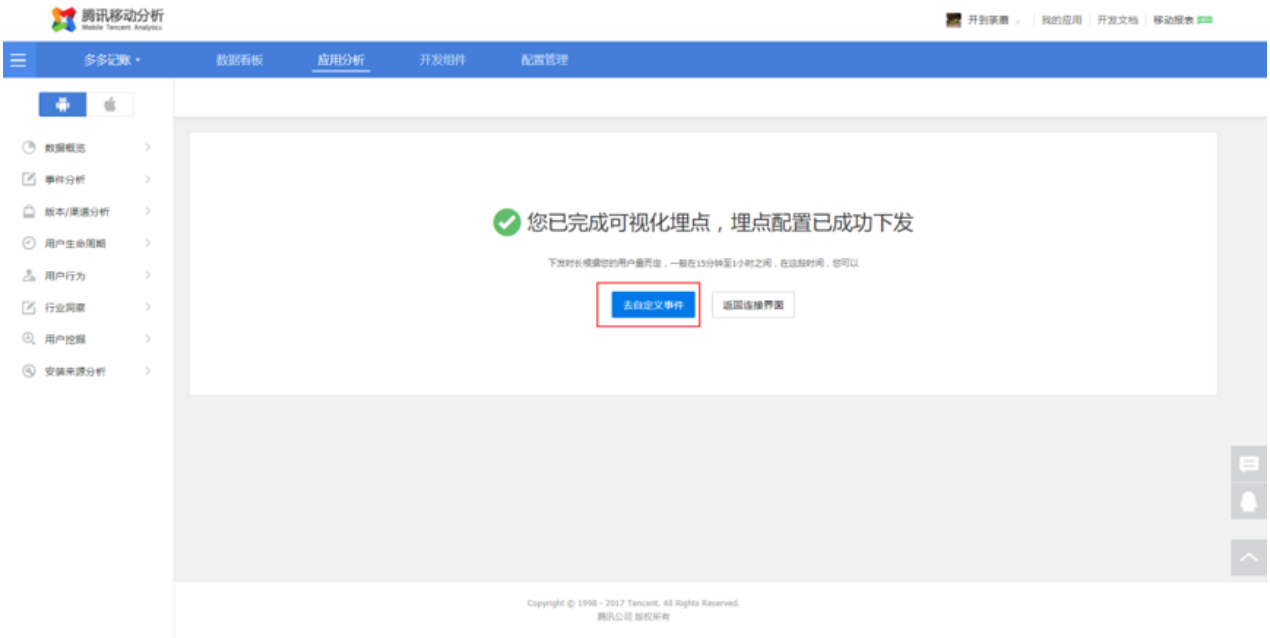
页面上应用 App 中，红色框表示可以埋点的地方，绿色框表示已经操作埋点的地方；左边栏是埋点区的操作日志，当在手机上单击埋点位置，会显示日志记录；右边栏是已埋点的列表，可进行相关操作。

具体埋点示例：

1. 鼠标单击应用左上角的【红色框】，弹出框命名，单击确定后该埋点会出现在后边栏的已埋点列表里。



2. 单击【生效下发】，会弹出确认提示框，单击确定，进入如下页面，埋点成功。



具体视频教学可见：

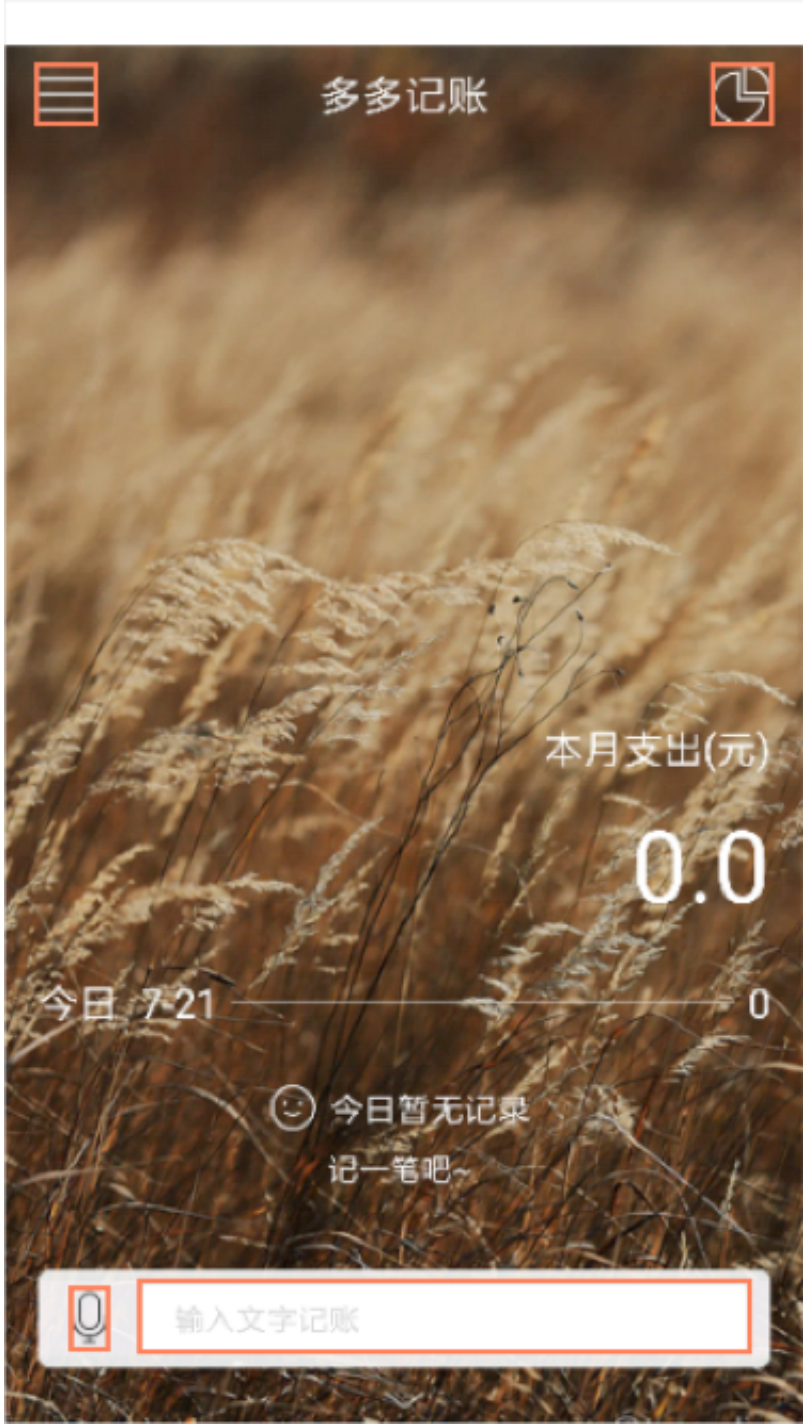
可视化埋点支持元素

Android 可视化埋点支持元素

有 Button、EditText、layout等。

> 注意：
> 要具有 click，selected，text_changed 属性的（如 button，edittext 等）并且值为 true
> 没有被遮盖（透明层也属于遮盖）的。

可埋点元素示例：
图中红框和绿框选中的元素表示该元素可以埋点，LinearLayout 或者是 button、click 都为 true



不可埋点元素示例：

图中十二个小柱子图片和下面的文字不具有 click 属性或者 click 为 false，因此不可埋。

iOS 可视化埋点支持元素

有 UIButton UISwitch 和 UISlider。

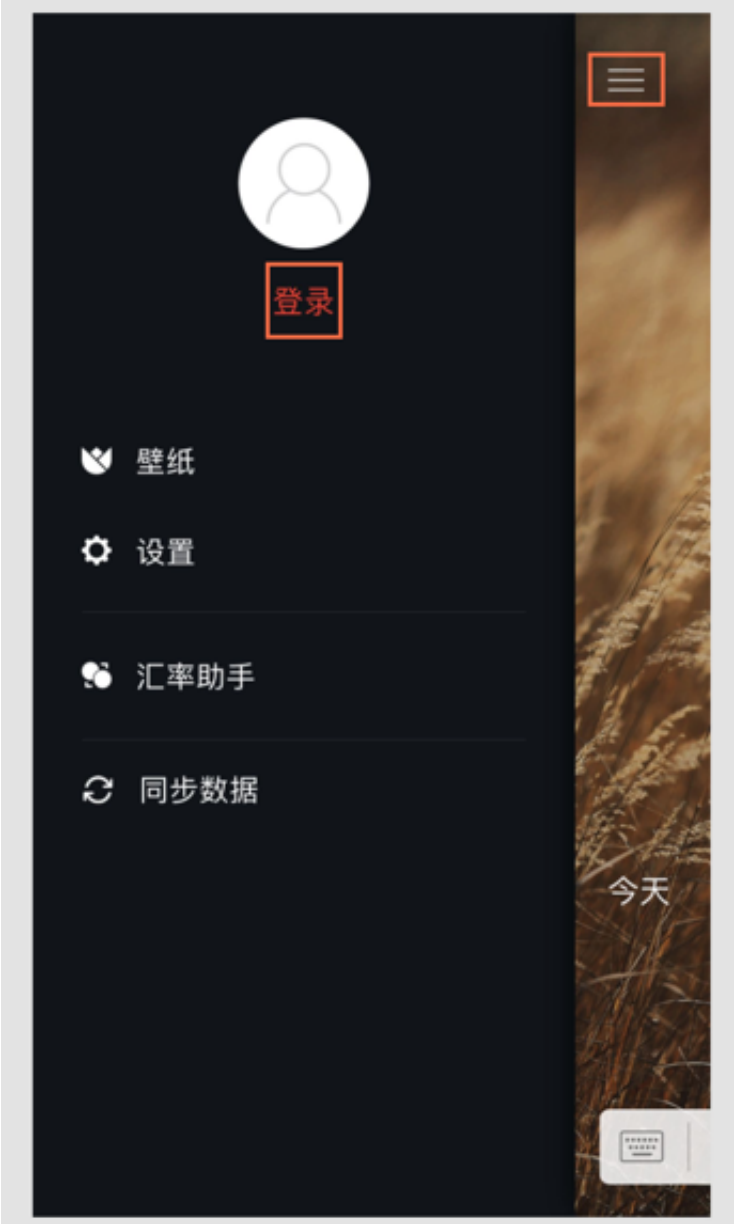
> 注意：
> 如果这些元素在 TableView 或者 CollectionView 中，则这些元素也不可埋点。

可埋点元素示例：

图中红框和绿框选中的元素表示该元素可以埋点，其中四个元素都为 UIButton，并且不在 TableView 或者 CollectionView 中，因此可以埋点。

不可埋点元素示例：

图中登录按钮上面的图片不是 UIButton，UISwitch 和 UISlider 的任何一个，因此不可埋；登录按钮下面的壁纸，设置等按钮虽然是 UIButton，但是他们处在一个 TableView 中，因此不可埋点。


登录
壁纸
设置
汇率助手
同步数据
今天

注意事项

若始终不能连接设备，请排查以下问题：

1. 确定进入了正确的 App 进行无埋点配置；
2. 当前 App 集成的 SDK 是否为最新的 SDK；
3. 确定设备正常接入网络；
4. 是否在应用打开进入主界面后才用手指触屏（加载页/广告页手势无法识别）；
5. 是否四指触屏的时候触发了 App
   页面的单击效果（如果主界面控件过多，可寻找页面元素简单的页面进行连接）；
6. 确定启动方法是否正确，iOS 需要在 App 启动 10
   秒内同时用四指触摸屏幕，持续三秒，若错过，需要杀死App 进程，然后再尝试；
7. 是否接入第三方 SDK 手势与 MTA 无埋点统计启动手势冲突。

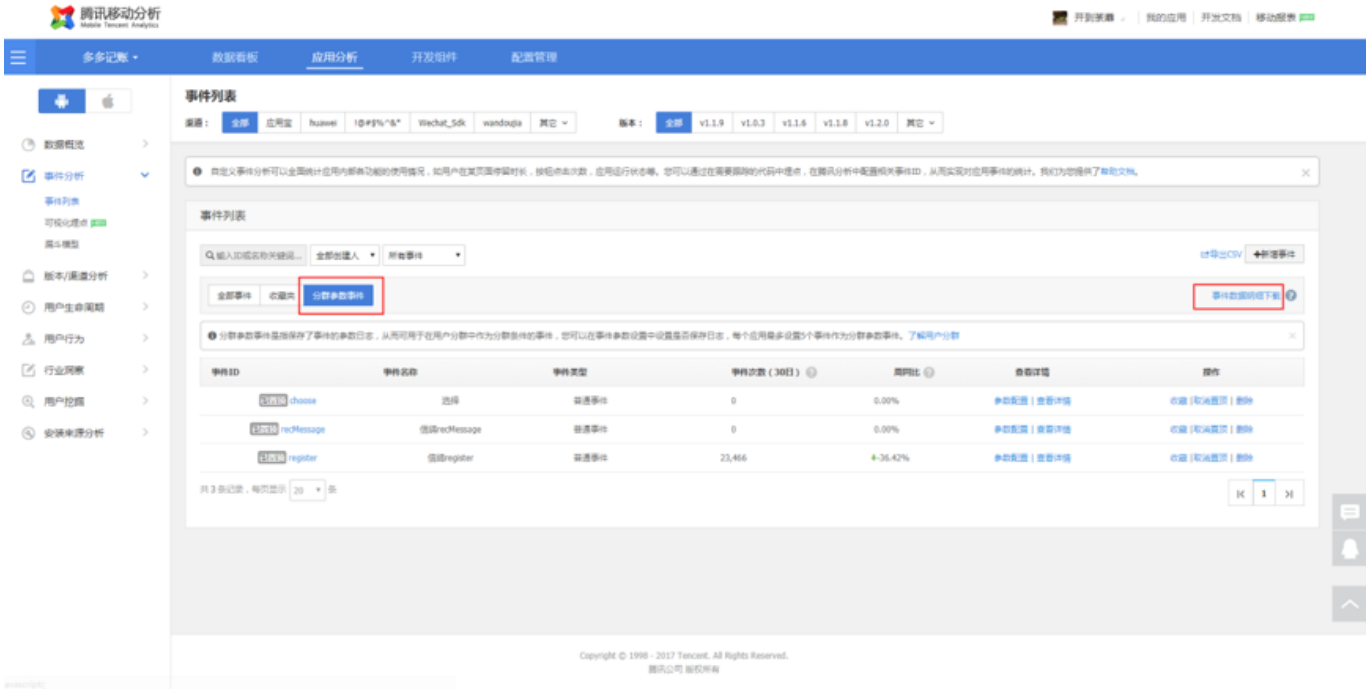
自定义事件明细导出

功能介绍

对用户自定义事件，MTA 平台不但提供基础统计分析，还提供数据明细导出服务，用户可获取明细数据，结合自身业务特性更深度数据分析挖掘数据价值，助力业务发展。
目前，平台提供两种数据导出方式：在线下载、后台推送。

在线下载

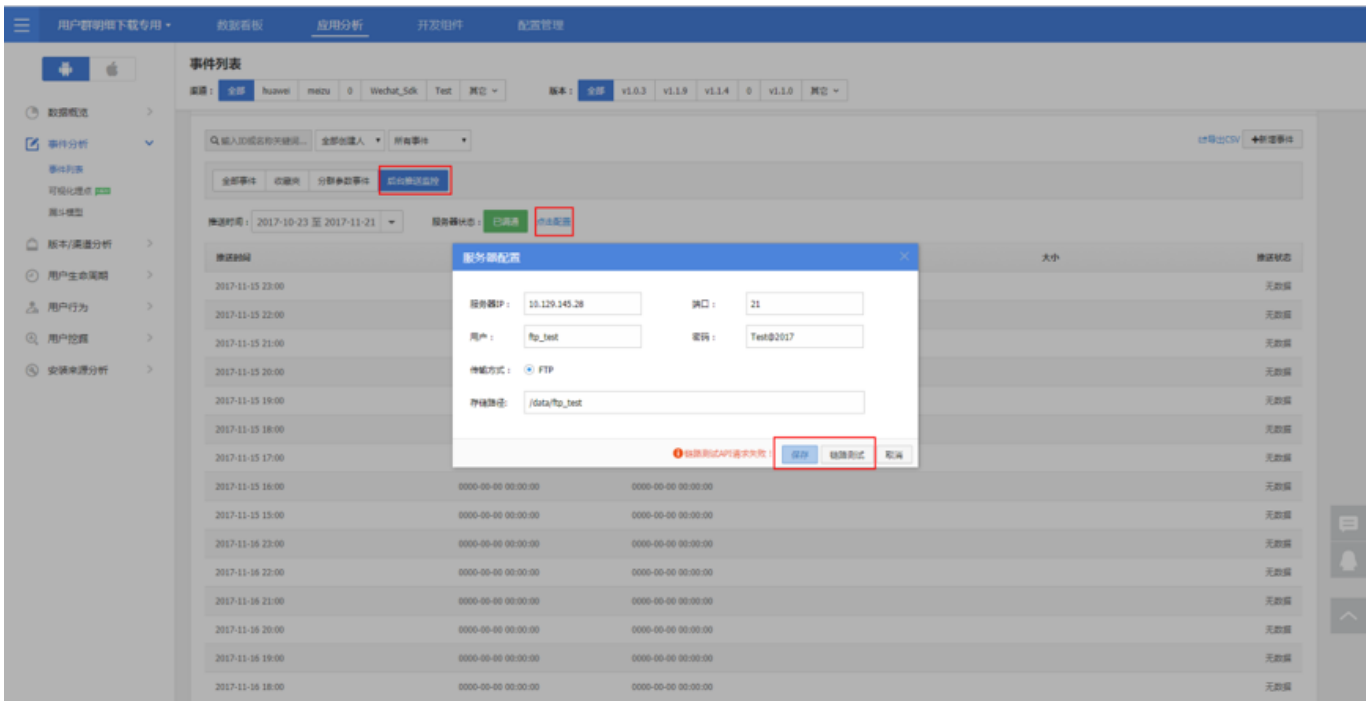
针对分群参数事件，用户可直接在 MTA 页面上，操作下载数据明细，其他事件不支持。
特别说明：分群事件只保留最近 100 万条分群参数事件数据明细，具体页面如下，直接单击即可下载文件。



后台推送

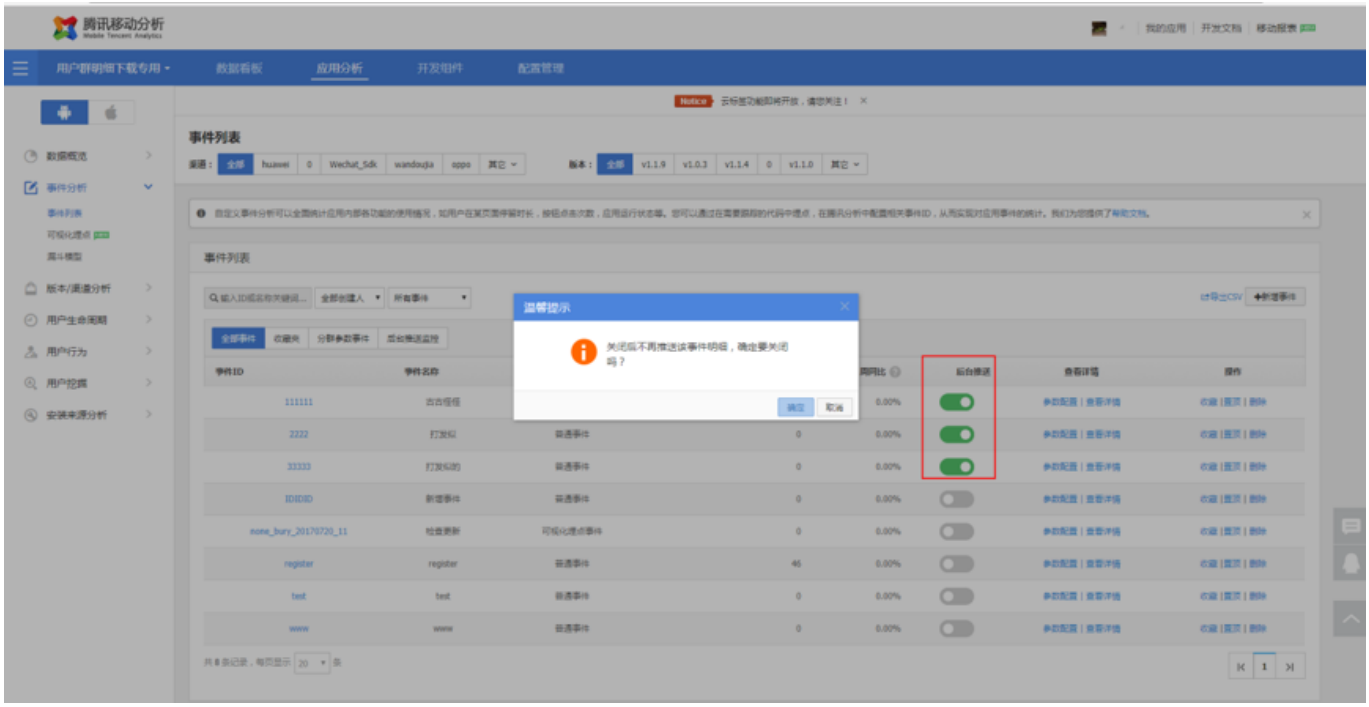
针对所有事件，用户可选择具体事件，平台将该事件数据明细以 FTP 方式推送到用户配置的服务器上。
特别说明：用户需配置服务器信息并保证链路通畅，具体操作如下：

第一步：配置服务器信息，具体如下，用户配置服务器信息后可单击调试链路是否与平台服务器打通。当用户配置的服务器与平台服务器链路打通时，服务器状态会显示【已调通】。



第二步：配置事件导出开关，具体如下，对自定义事件，可以设置后台推送的开关，服务器会把推送状态为【开】的事件，按每小时 1

次把这些事件的数据明细推送到指定服务器，可在【后台推送监控】页面查看推送情况。



数据格式

在线下载 和 后台推送，明细以 jason 串方式导出，具体字段如下：

名称	字段说明
ui	user id，Android 为 IMEI，iOS 为 IFA 或 openUDID
mc	设备的 mac 地址
ts	timestamp，时间戳
idx	event index，事件编号
cui	Custom user id，用户自定义的 ID
av	App version，应用的版本
ch	channel，应用安装（推广）渠道
id	与 ky 一一对应的 int 类型 ID
ei	event id，自定义事件 ID
du	duration，时长
kv	Key-Value，自定义事件 Key-Value 参数

数据上报策略

Android 数据上报策略配置

上报策略基础介绍

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）。

```
StatConfig.setStatSendStrategy(StatReportStrategy.INSTANT);
```

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）。

voidStatConfig.setEnableSmartReporting (boolean isEnable)

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。

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。

iOS 数据上报策略配置

上报策略基础介绍

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 MTA_STRATEGY_APP_LAUNCH）

@property MTAStatReportStrategy reportStrategy

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	MTA_STRATEGY_INSTANT	实时发送，App 每产生一条消息都会发送到服务器。
2	MTA_STRATEGY_ONLY_WIFI	只在 WiFi 状态下发送，非 WiFi 情况缓存到本地。
3	MTA_STRATEGY_BATCH	批量发送，默认当消息数量达到 30 条时发送一次。
4	MTA_STRATEGY_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	MTA_STRATEGY_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	MTA_STRATEGY_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 MTA_STRATEGY_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 MTA_STRATEGY_APP_LAUNCH或MTA_STRATEGY_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）。

@property BOOL smartReporting

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，App 下次启动时会自动生效并存储该策略。

上面 3 种方式的优先级依次递增。例如，WiFi 下转为实时发送会优先于第 1 种方式中选定的任何策略执行，在 Web 界面上配置的策略会覆盖 App 本地已经生效的策略。

数据上报相关的设置

1.设置最大缓存未发送消息个数（默认 1024）。

@property uint32_t 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 MTA_STRATEGY_BATCH 时有效）设置最大批量发送消息个数（默认 30）。

@property uint32_t minBatchReportCount

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）。

@property uint32_t sendPeriodMinutes

App 设置接口

功能介绍

使用这些函数可以动态调整 App 和 SDK 的相关设置。

Android App 接口设置

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.setSessionTimoutMillis(int sessionTimoutMillis)

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.setMaxParallelTimmingEvents(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)

若为 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)

10.支持多进程（默认为 false）
同一个 App 多个进程同时使用 MTA，请参考注意事项中的“多进程支持”。

void StatConfig.setEnableConcurrentProcess(boolean enableConcurrentProcess)

iOS App 接口设置

1.使用 MTAConfig 单例对象属性设置可以动态调整 App 和 SDK 的相关设置，调用形式为：

[[MTAConfig getInstance] setPropertyName:value];

2.会话时长（默认 30s，离开应用 30 秒之后再回来，视为一次新的会话）

@property uint32_t sessionTimeoutSecs

3.消息失败重发次数（默认 3）

@property uint32_t maxSendRetryCount

4.用户自定义时间类型事件的最大并行数量（默认 1024）

@property uint32_t maxParallelTimingEvents

5.设置安装渠道(默认为 appstore)

@property (nonatomic, retain) NSString* channel

6.设置App Key

@property (nonatomic, retain) NSString* appkey

7.设置统计功能开关（默认为 true）

@property BOOL statEnable

如果为 false，则关闭统计功能，不会缓存或上报任何信息，设置 session 内发送消息限制（默认为 0，即无限制）。

@property int32_t maxSessionStatReportCount

若为 0，则不限制 session 内发送消息的个数；若大于 0，每个 session 内发送的消息不会超过此值；若超过了，新产生的消息将会被丢弃。

用户分群

概览

功能概括

用户分群帮助运营、产品人员更好的了解用户，进行针对性运营及优化产品。首先用户分群允许产品运营者根据筛选条件（例如：用户活跃时间段，新用户注册时间，使用设备以及开发者自定义事件的追踪行为）将用户进行分群归类，分群后您将获得该用户群在您 App 中的留存率、用户流失与回流数据以及用户活跃度。

立即体验>>用户分群

独特优势

- 在创建您所需要分析的用户群体时，您不仅可以通过时间、活跃度、用户所用的设备属性来筛选用户，还可以通过自定义事件来筛选用户，例如：您可以筛选出在您的 App 里有购买行为的用户；
- 在创建用户群时您不仅可以通过特定的自定义事件筛选用户，也可以通过漏斗模型来筛选您所需要的用户，例如：您可以筛选出触发了【用户注册】>【加入购物车】>【购买商品】这样特定顺序事件的用户群；
- 用户分群后，您可以查看到该用户群中用户的留存率，用户流失与回流以及用户的活跃度等重要信息。

应用场景

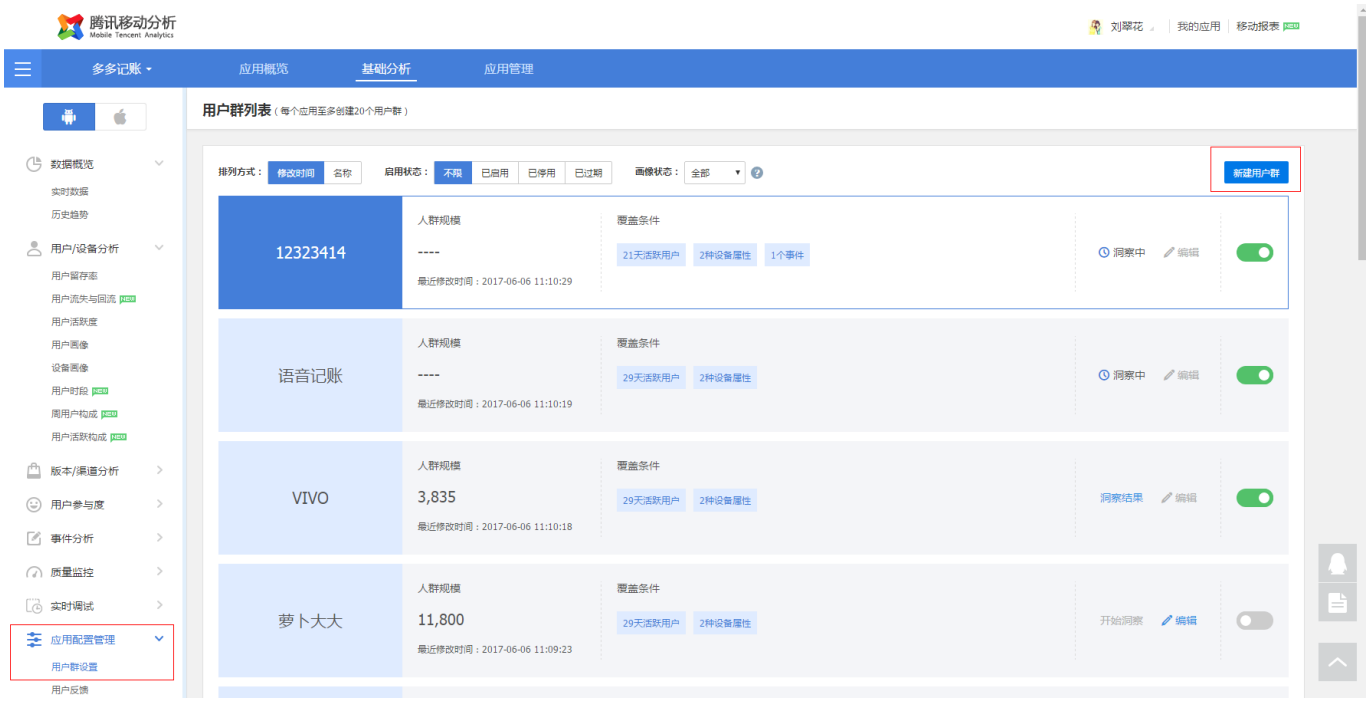
- 帮助运营人员根据用户群体特征制定针对性的运营战略
  运营人员可以根据用户的流失与回流制定相应的运营活动，例如当用户留存率变低，用户流失过多时可以加大运营活动的举办，同时可以通过用户的回流量来判断运营活动是否成功。
- 帮助产品经理根据用户群体特征确定产品迭代优化方向
  同样，产品经理可以将迭代版本中添加的新功能通过打点记录使用该功能的用户数据，并将使用新功能的用户划分为一个用户群，再通过判断这一群体的统计数据分析该功能的可用性。例如，将使用微信的【摇一摇】功能的用户划分为一个用户群，通过统计分析该用户群的流失与回流等数据来判断【摇一摇】这一功能的可用性。

操作指南

要使用用户分群功能，需在 MTA 管理台成功注册自己的App，然后通过以下简单的步骤便可查看您需要的用户分群数据结果：

在管理台中选择功能

前往 MTA 管理台，选择一个 App，同时注意选择 Android 或 iOS 平台。在左侧菜单栏，选择【应用配置管理】下的【用户群设置】功能，然后单击右上角的【新建用户群】按钮。



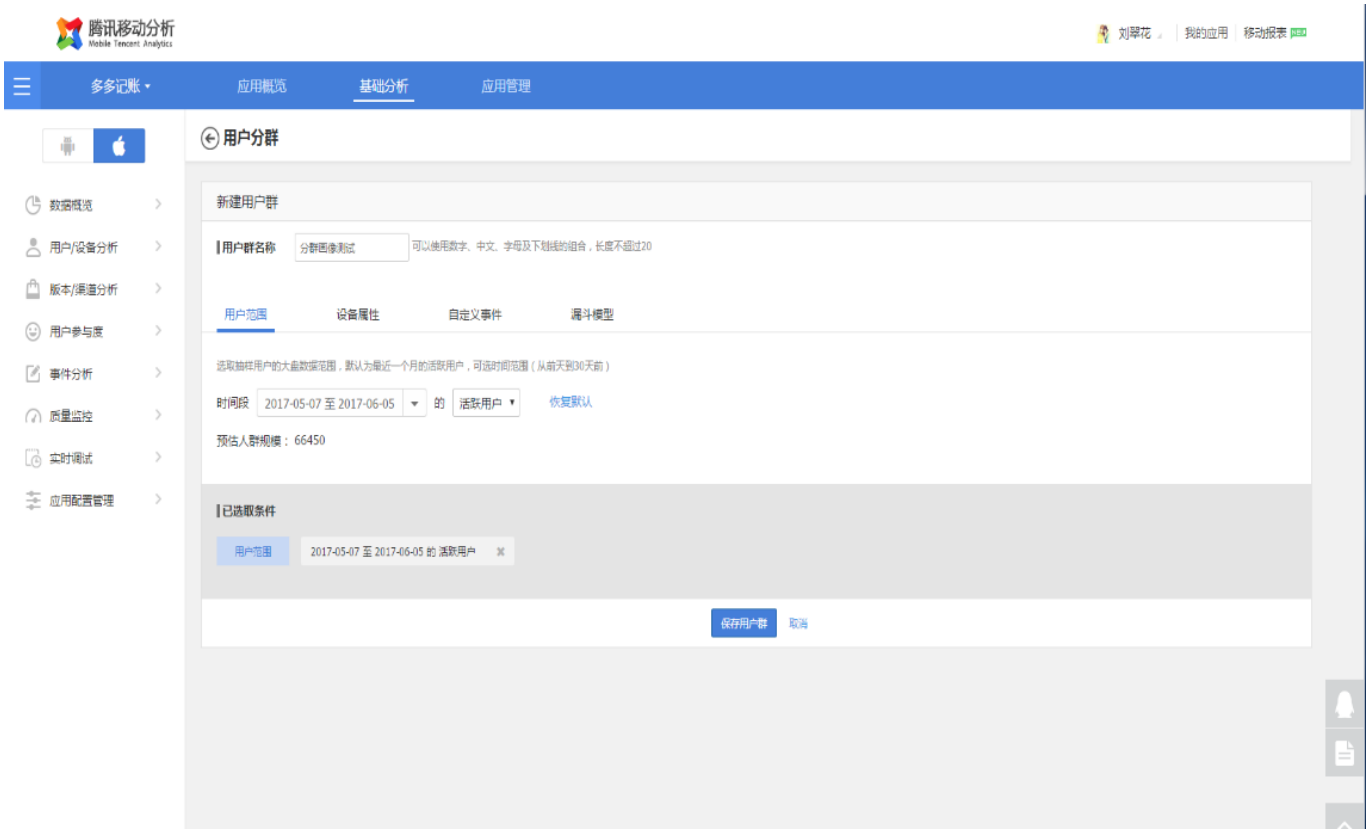
新建用户群

单击新建用户群后，按照您的需求填写下图中创建群资料页面，您可以通过以下的筛选条件来创建您所需的用户群：

1. 【用户范围】条件下，可筛选指定时间段中的活跃用户/新增用户；
2. 【设备属性】条件下，可筛选终端设备机型及版本；
3. 在【自定义事件】条件下，可以筛选出指定时间段下触发特定事件的用户群；
4. 在【漏斗模型】条件下，可筛选出某一时间段下按照指定顺序触发特定事件的用户群体，填写完后单击【保存用户群】。

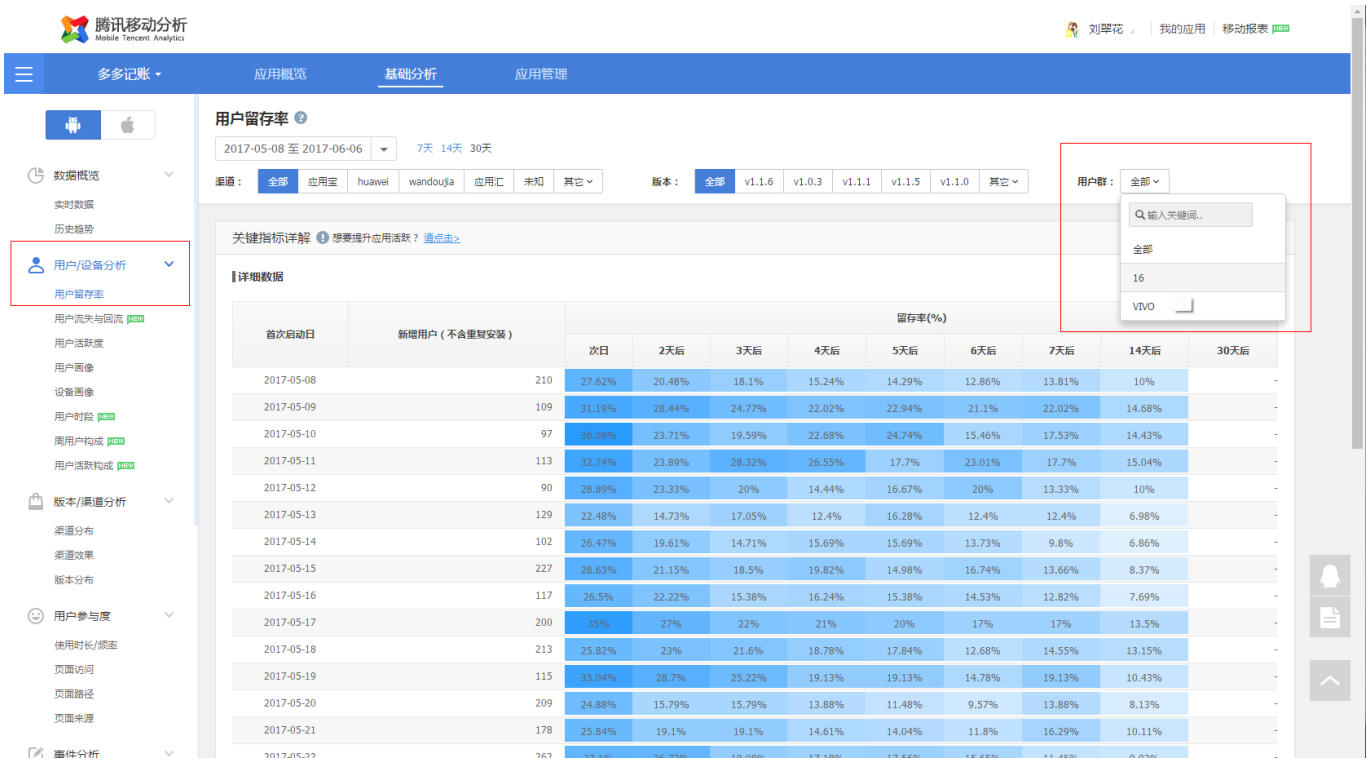
注意：

1. 最多可以创建 20 个用户群；
2. 最多可同时启用 5 个用户群：启用中的用户群，每天统计数据结果，您无法编辑启用中的用户群，需关闭后才可以进行编辑，用户群新建/编辑后，您需手动启用确认后方可开始数据更新；
3. 用户群于 1 个月后到期，到期后将自动转入关闭状态，届时您可以再次启用该用户群 。



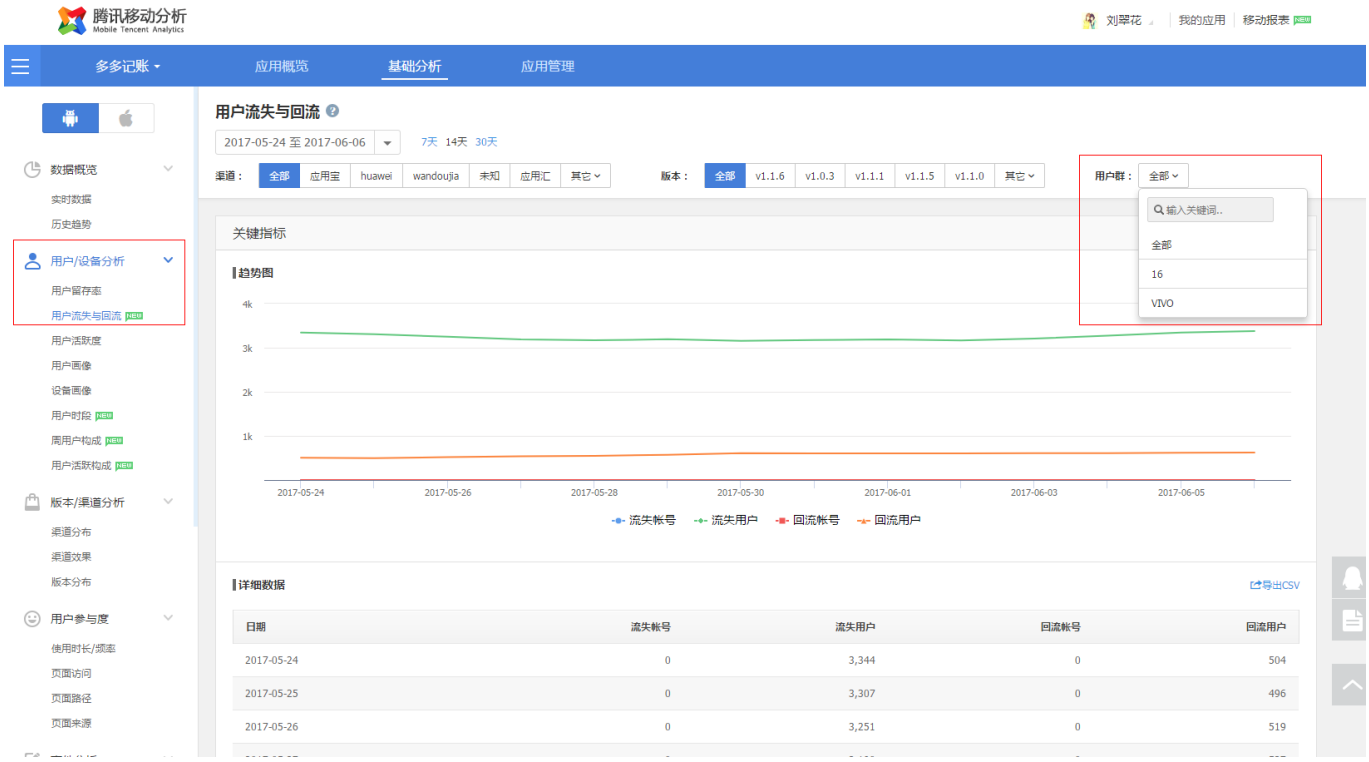
查看用户群的留存率

创建完您需要的用户群后，打开右侧的用户群开关启用用户群，启用后后台会统计每天的数据，单击【用户/设备分析】目录下的【用户留存率】并在右上角（如下图）的用户群中选择您需要查看的用户群名称，用户留存率越低颜色越浅，您也可以选择用数字的方式查看，数据结果可以用.csv
的格式导出，同时，您需要选择时间（您可选择查看 7 天/ 14 天/30天
的留存率）以及用户使用终端的品牌及型号。



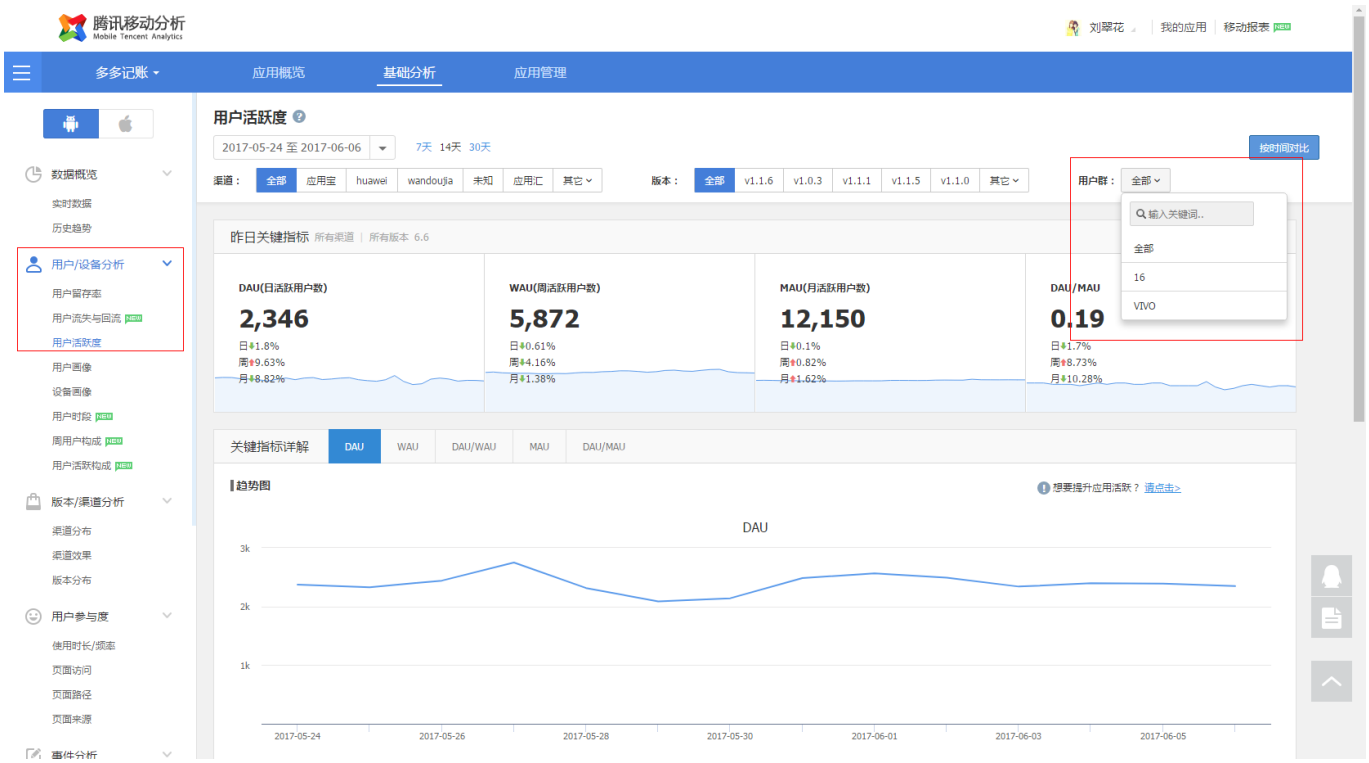
查看用户群的流失与回流

单击【用户/设备分析】目录下的【用户流失与回流】并在右上角（如下图）的用户群中选择您需要查看的用户群名称，您可以在【用户流失与回流】查看到用户流失与回流账号的趋势折线图以及账号的详细信息，数据可以通过 csv 导出，同时，您需要选择时间以及用户使用终端的品牌及型号。



查看用户群的用户活跃度

单击【用户/设备分析】目录下的【用户活跃度】并在右上角（如下图）的用户群中选择您需要查看的用户群名称，您可以在【用户活跃度】中查看到用户活跃度的账号总数、活跃度变化趋势折线图以及活跃账号的详细信息，同时，您需要选择时间以及用户使用终端的品牌及型号。



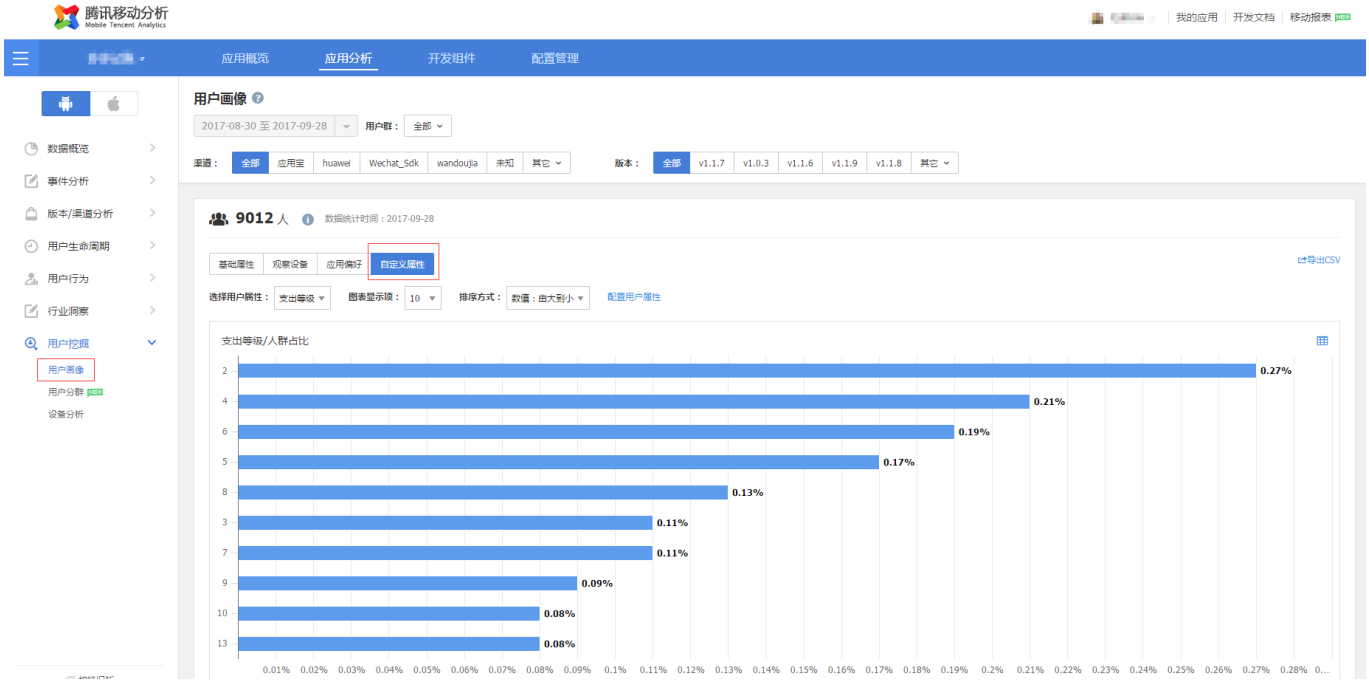
自定义用户属性

功能介绍

您可以为您的用户设置属性标识，比如游戏用户的会员等级、电商用户累计消费金额、常见的 A/B Testing 中的 AB 分组等，并通过管理台查看其属性分布或创建不同属性的用户群，从其活跃程度、留存率、流失、用户画像等全维度进行对比分析。

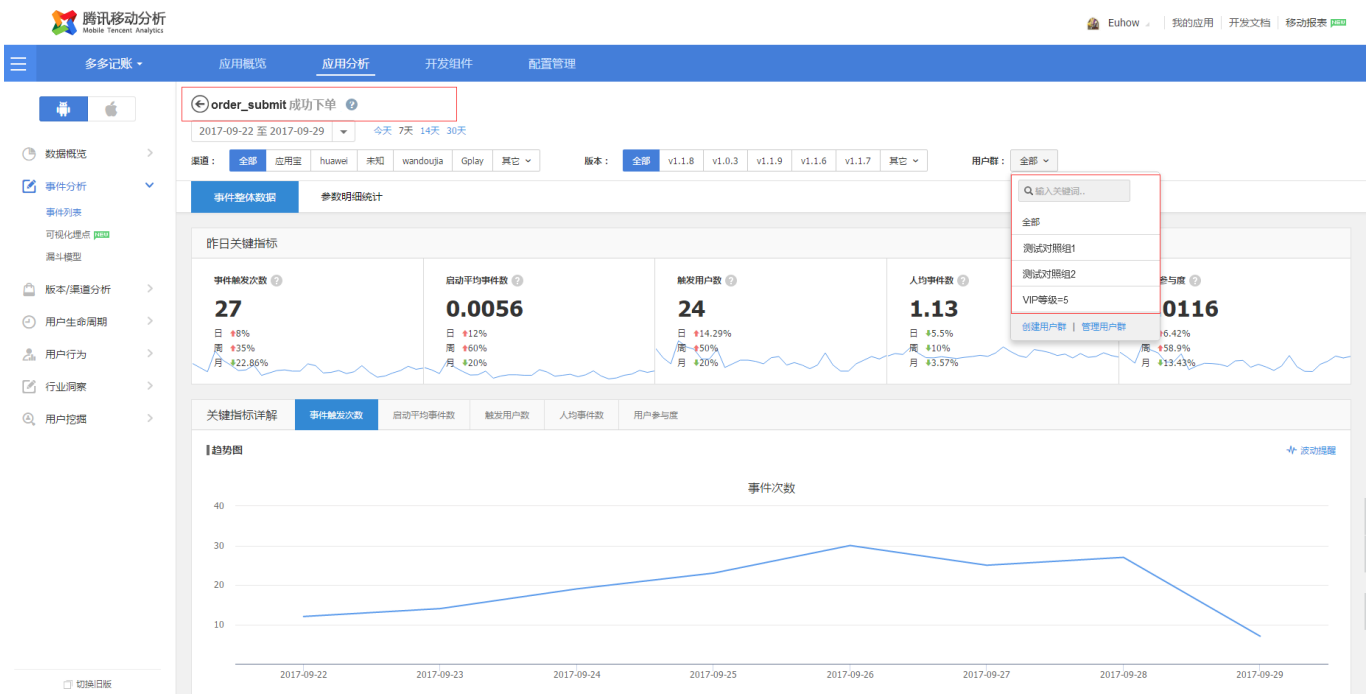
查看用户属性分布

您可以在管理台【应用分析】>【用户挖掘】>【用户画像】>【自定义属性】中查看用户的不同属性分布。



对比不同属性人群的用户行为

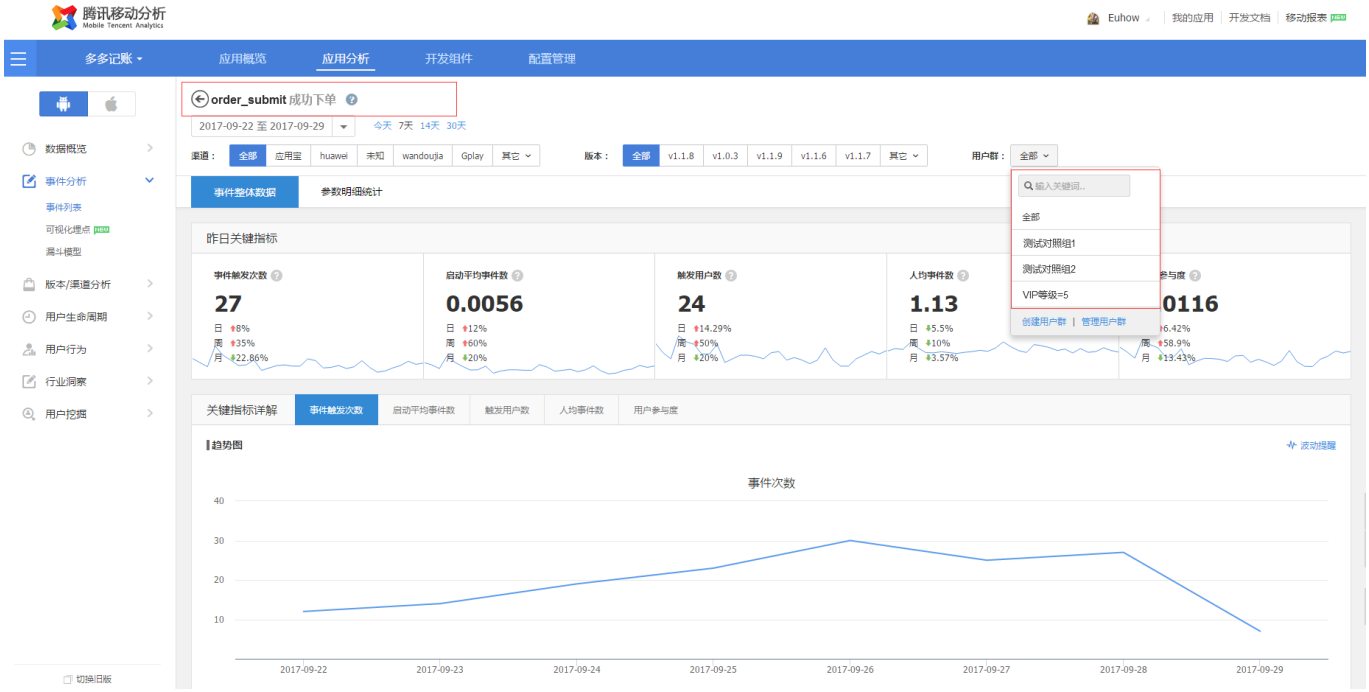
通过用户属性设定不同的用户群，对比他们完成的效果，如设定参照组 1 和 2，两组人群使用不同的下单流程，再对比最终单击下单的次数、用户数区别。


腾讯移动分析
Euhow
我的应用
开发文档
移动报表
多多记账
应用概览
应用分析
开发组件
配置管理
order_submit 成功下单
2017-09-22 至 2017-09-29
今天 7天 14天 30天
渠道：
全部
应用宝
huawei
未知
wandoujia
Gplay
其它
版本：
全部
v1.1.8
v1.0.3
v1.1.9
v1.1.6
v1.1.7
其它
用户群：
全部
数据概览
事件分析
事件列表
可视化埋点
漏斗模型
版本/渠道分析
用户生命周期
用户行为
行业洞察
用户挖掘
事件整体数据
参数明细统计
昨日关键指标
事件触发次数
27
日 8%
周 35%
月 22.86%
启动平均事件数
0.0056
日 12%
周 60%
月 20%
触发用户数
24
日 14.29%
周 50%
月 20%
人均事件数
1.13
日 5.5%
周 10%
月 3.57%
0116
6.42%
58.9%
13.43%
输入关键词..
全部
测试对照组1
测试对照组2
VIP等级=5
创建用户群 | 管理用户群
关键指标详解
事件触发次数
启动平均事件数
触发用户数
人均事件数
用户参与度
趋势图
波动提醒
事件次数
40
30
20
10
2017-09-22
2017-09-23
2017-09-24
2017-09-25
2017-09-26
2017-09-27
2017-09-28
2017-09-29
切换旧版

操作指南

配置应用的用户属性

在您的【应用管理台】>【配置管理】>【配置用户属性】中，配置您的用户属性。



在代码端完成属性上报

Android 上报自定义用户属性：

```
// 封装MTA用户属性参数，key-value格式
JSONObject customProperty = new JSONObject();
customProperty.put("用户属性1", "属性值1");
customProperty.put("用户属性2", "属性值2");
// 调用接口上报
// context：app上下文，customProperty：具体的自定义属性值
StatService.reportCustomProperty(context, customProperty);
```

iOS 上报自定义用户属性：

```
/**
```

支持用户自定义属性

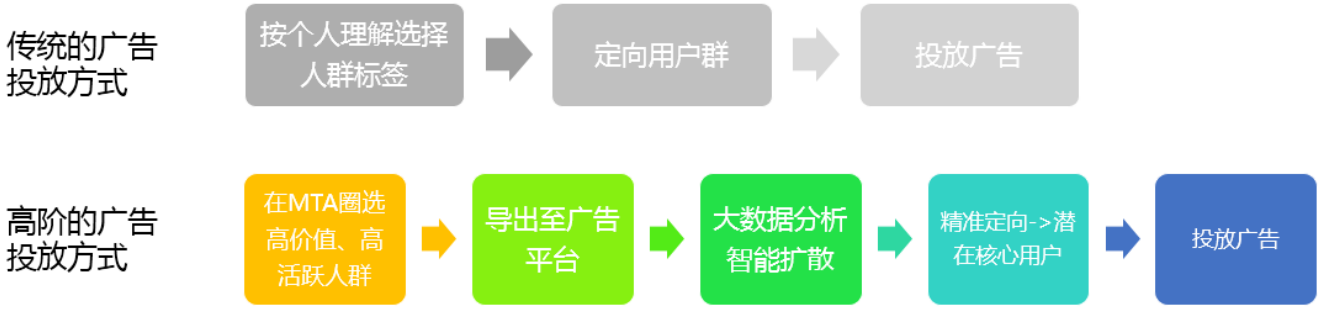
@param kvs key-value形式，例如"用户属性1", "属性值1"
*/
+ (void)setUserProperty:(NSDictionary *)kvs;

广告精准定向

概览

功能介绍

为了帮助移动应用开发者、运营、市场人员最大化的提升推广拉新效果，腾讯移动分析（MTA）提供了新的广告人群定向能力，通过与广告系统的人群扩散能力相结合，帮助您精准定位潜在的核心用户群体，提升广告投资回报率。



普通的广告投放方式：

根据个人主观理解，选择广告系统提供的人群标签，如城市、年龄、性别、兴趣偏好等，基于标签进行广告投放；

高阶的投放方式：规避操作者对产品目标用户理解的偏差，不再受限于广告标签的系统，通过 MTA 用户分群功能，把应用内的高价值、高活跃人群圈选出来，推送至广点通中进行人群扩散，从而得到与核心用户群具有相同特征的用户包，这个用户群对您的产品更感兴趣，且拥有极大的可能会成为你的高价值、高活跃用户（即潜在核心用户），因而 ROI 大大提升。

支持平台

广点通：

腾讯社交广告 是基于腾讯社交网络体系的效果广告平台，包括了 QQ 空间、QQ 客户端、手机 QQ 空间、手机 QQ、微信、QQ
音乐客户端、腾讯新闻客户端等诸多平台广告投放资源，可以在该平台上进行产品推广。

智赢销：

智赢销 是由腾讯公司最新推出的以品牌广告为主的自助营销服务平台，其整合了腾讯视频、腾讯新闻客户端、天天快报等优质媒体资源，可以在该平台上进行产品推广。

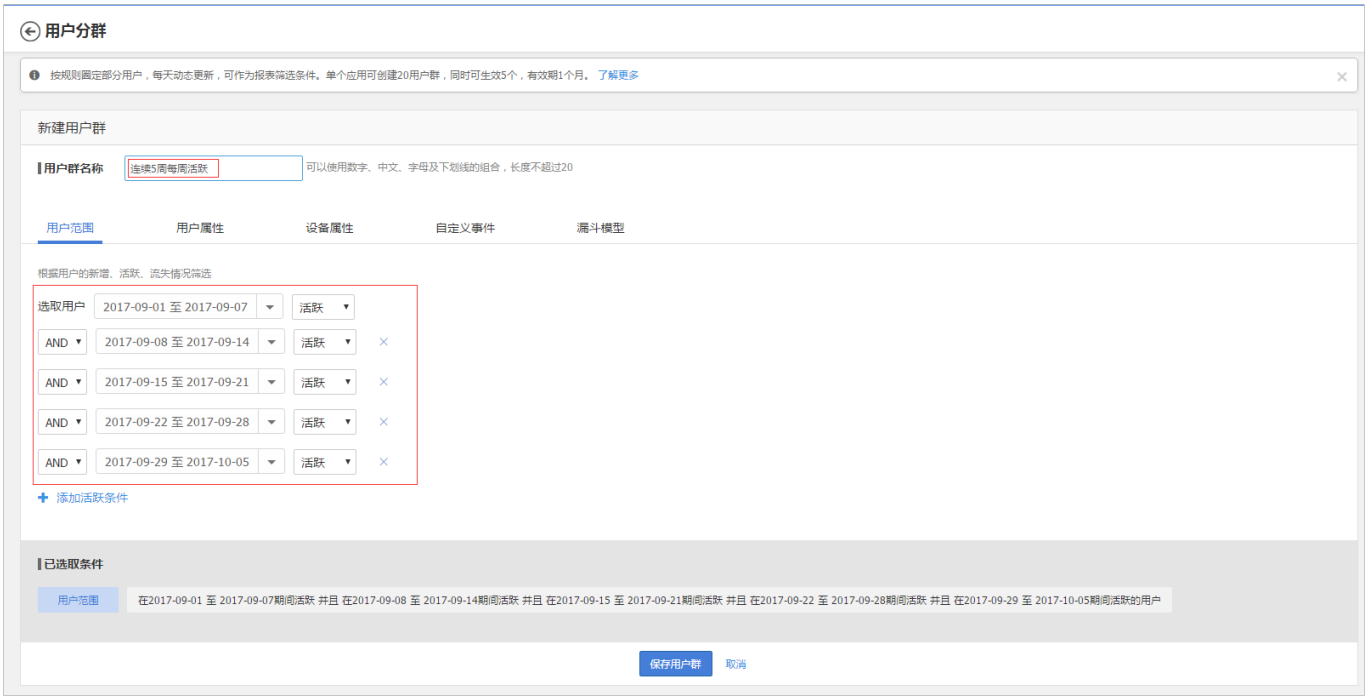
如您有更多平台需支持，请联系客服或发送邮件至 dtsupport@tencent.com

用户群导出至广点通

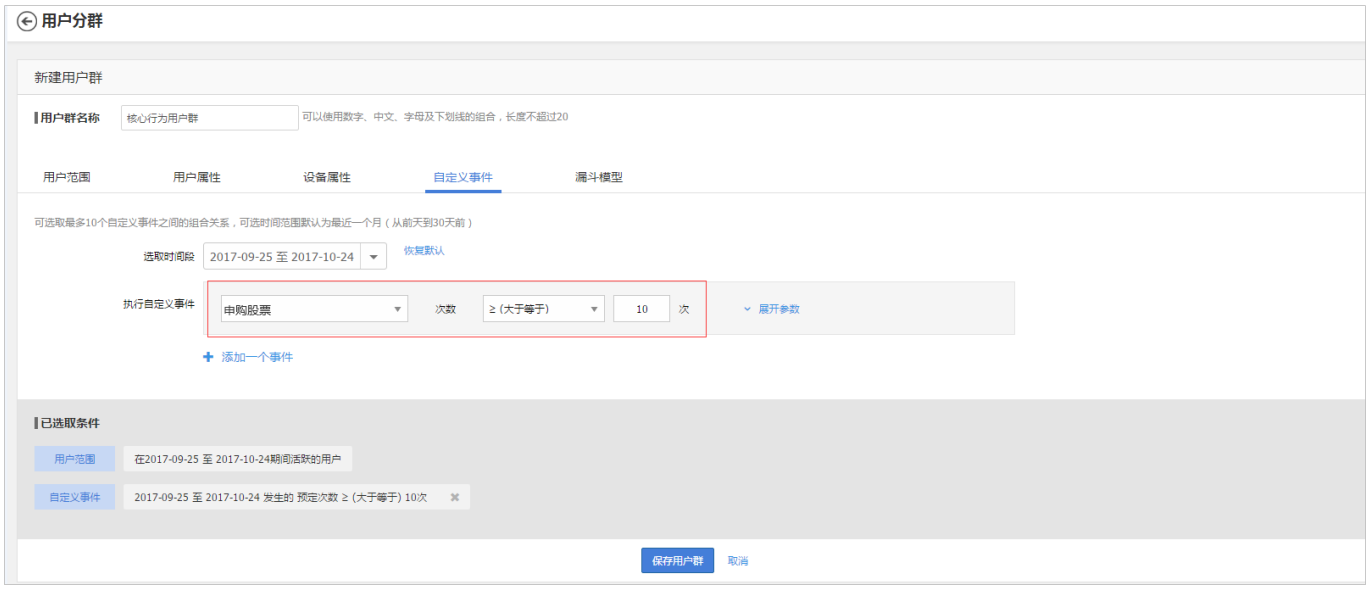
在 MTA 中圈选出应用内的高价值、高活跃用户：

在 MTA 管理台【应用分析】中找到【用户挖掘】>【用户分群】功能，单击【新建用户群】。

根据活跃条件选择高活跃的用户群，比如连续 5 周每周都活跃的用户群体，选择后保存用户群。



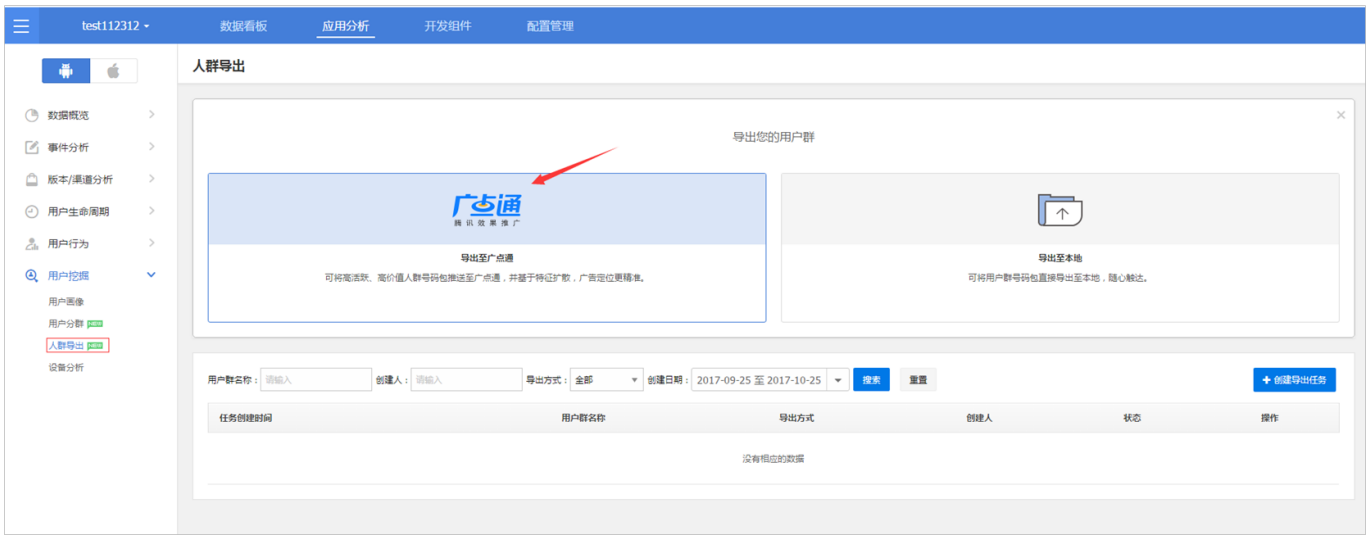
或者选择【自定义事件】，根据用户的最高价值行为次数来分群，如股票类应用认为【申报股票】是高价值行为，则可以通过用户分群筛选出【申报股票】超过 10 次的用户群。



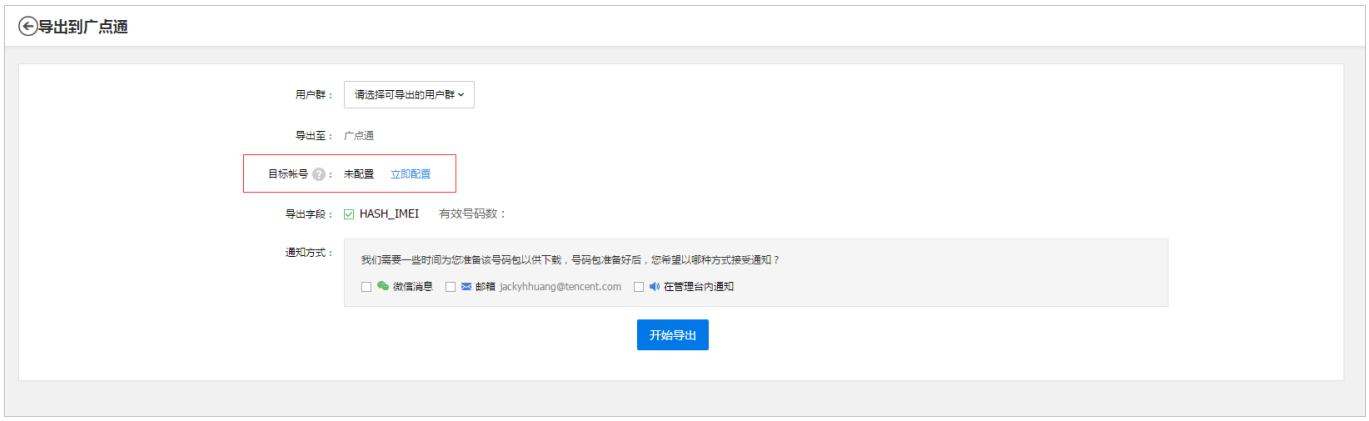
保存用户群后，单击开关启用用户群，用户群会在第二天计算出结果，不用等待计算出结果，先进行广点通账号配置。

配置您的广点通账号

您可以在 MTA 管理台【应用分析】中找到【用户挖掘】>【人群导出【功能，单击选择【导出至广点通】。



在接下来的导出页面中，单击【立即配置】以配置广点通的目标账号。

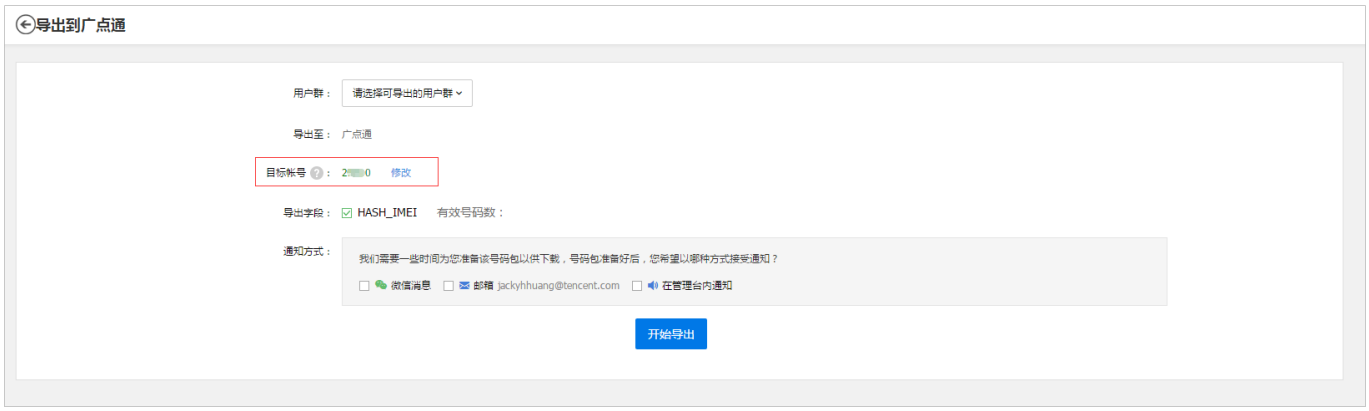


在打开的授权页面中，您需要登录在广点通的 DMP 管理平台拥有访问权限的 QQ 号，并对 MTA 进行授权 。

注意：

1. 若您的业务未开通过广点通账号，请访问 腾讯社交广告
   开通广点通投放平台先开通广点通账号；
2. 若您的业务已开通广点通账号，但是未开通广点通 DMP 管理平台，请访问
   广点通数据管理平台 滚动至页面底部获取联系方式；
3. 若您所在公司都已开通相关账号，但账号不能共享给您，您可以在配置管理中将其 QQ 号加入系统，并由您的同事完成授权操作。

授权完成后确认账号是否配置成功，配置成功后关闭页面（如果已有可导出用户群，可选择用户群并直接导出至广点通）。

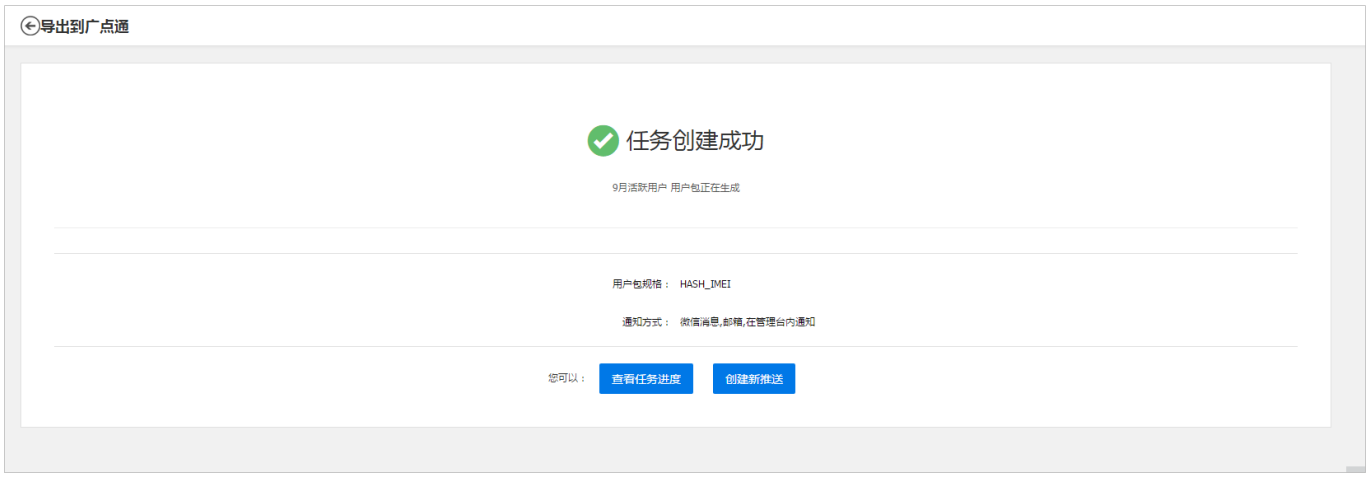


导出人群并在 DMP 系统中进行人群扩散，投放广告

用户群计算完成后，在人群列表对应的人群单击【导出】，选择【导出至广点通】，在接下来的页面配置好用户群以及导出字段，单击【开始导出】。

> 注意：
> 用户群的用户量大于等于 10,000 以上时，人群扩散效果最佳。

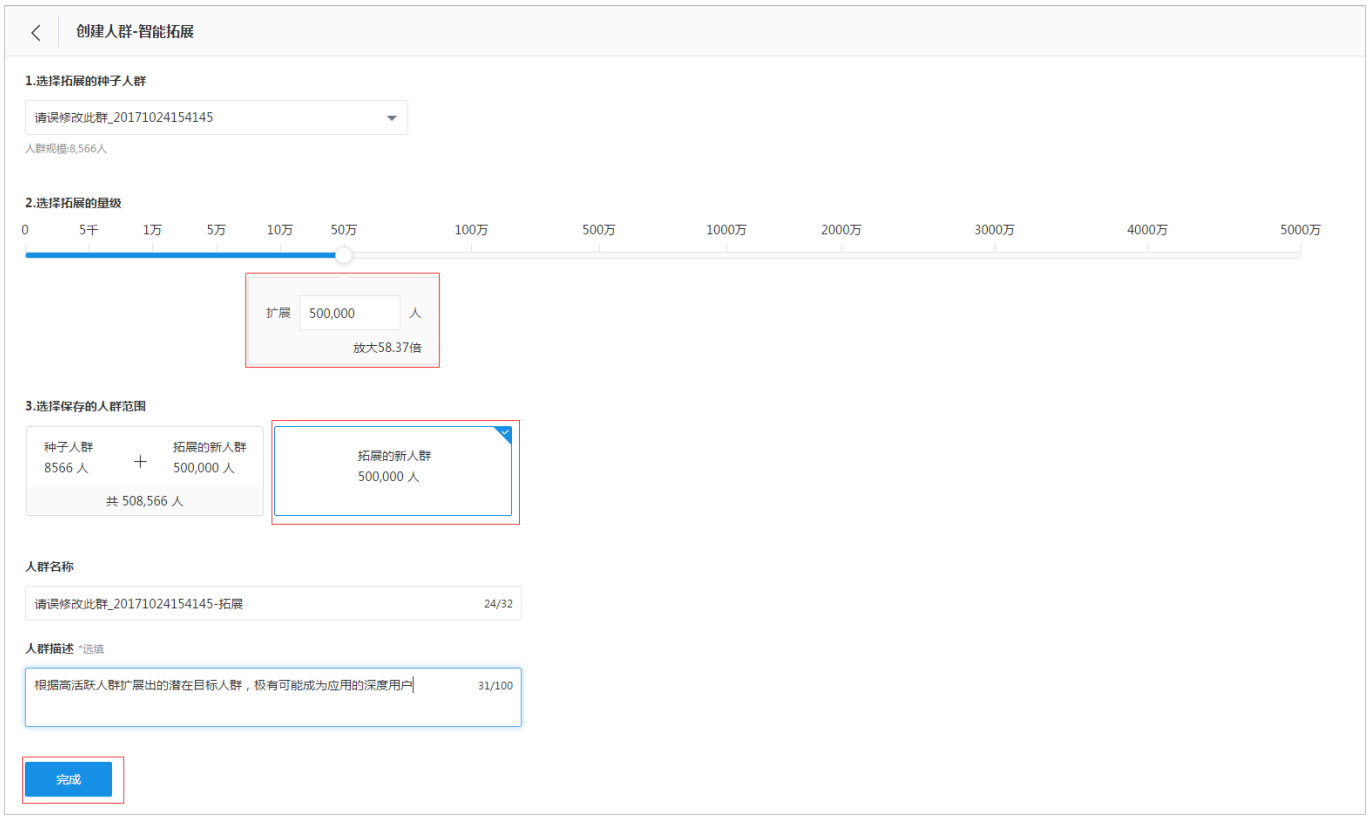
单击【开始导出】后，可以单击【查看任务进度】，人群若成功导出至广点通，您将会收到通知提醒（上一步骤中配置的提醒方式）。



收到通知提醒后，登录广点通数据 平台系统，查看对应的人群状态。

当人群状态可用时，勾选对应人群并选择【操作】>【智能扩展】。

选择扩展的量级，去除已有用户，保存后待人群计算完成，即可在广点通投放广告时选择该人群进行投放。



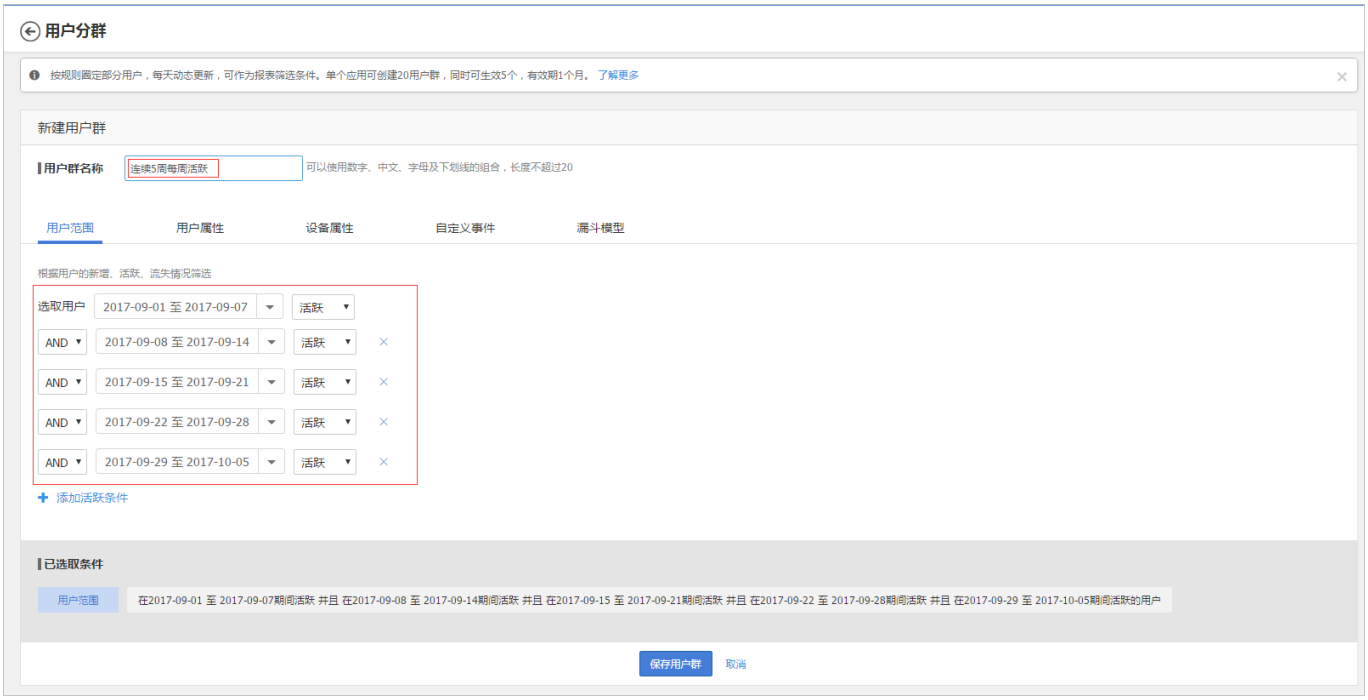
如果您有任何疑问，可在 MTA 官网右下角联系我们的客服人员。

用户群导出至智赢销

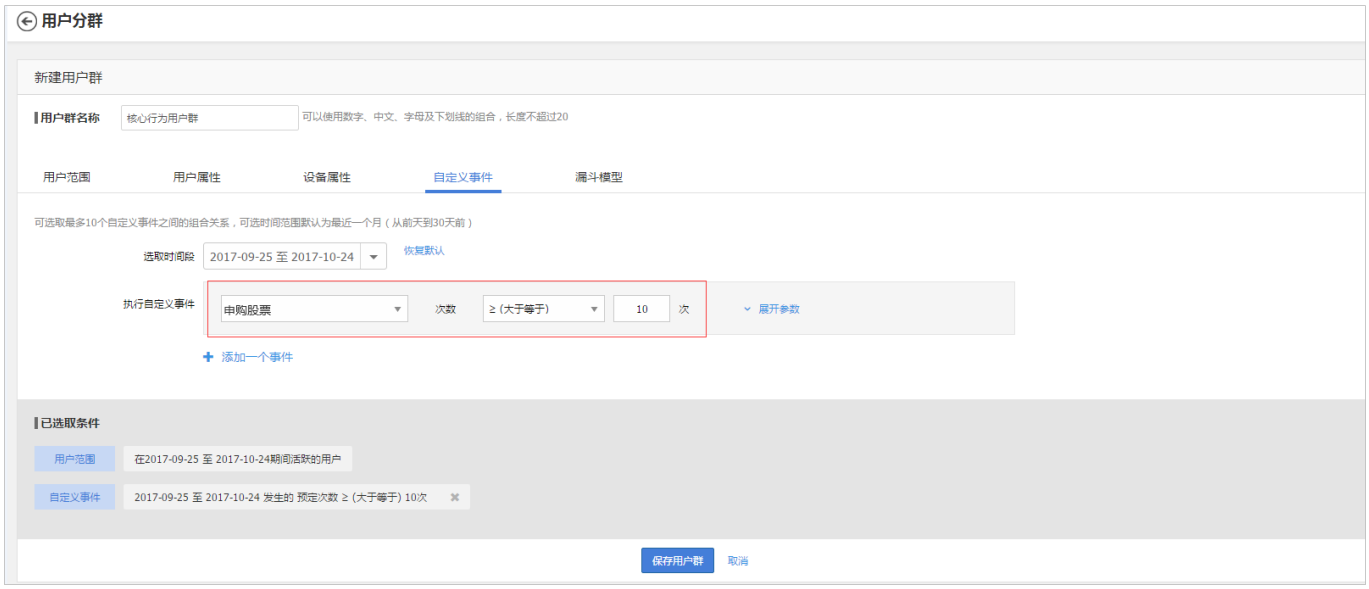
在 MTA 中圈选出应用内的高价值、高活跃用户

在 MTA 管理台【应用分析】中找到【用户挖掘】>【用户分群】功能，单击【新建用户群】。

根据活跃条件选择高活跃的用户群，比如连续 5 周每周都活跃的用户群体，选择后保存用户群。



或者选择【自定义事件】，根据用户的最高价值行为次数来分群，如股票类应用认为【申报股票】是高价值行为，则可以通过用户分群筛选出【申报股票】超过 10 次的用户群：



保存用户群后，单击开关启用用户群，用户群会在第二天计算出结果，不用等待计算出结果，先进行广点通账号配置。

配置您的智赢销账号

您可以在 MTA 管理台【应用分析】中找到【用户挖掘】>【人群导出】功能，单击选择【导出至智赢销】。

在接下来的导出页面中，输入您在智营销中开通了 DMP 平台使用权限的 QQ 号并验证。

注意：

1. 若您的业务未开通过智赢销账号，请访问 智赢销 注册，并按网站提示申请；
2. 若您的业务已开通智赢销账号，但是未开通智赢销 DMP 管理平台，请访问 智赢销 DMP 登录申请；
3. 若您所在公司都已开通相关账号，但账号不能共享给您，您可以在配置管理中将其 QQ 号加入系统，并由您的同事完成配置操作。

配置完成后， 关闭页面（如果已有可导出用户群，可选择用户群并直接导出至智赢销）。

导出人群并在 DMP 系统中进行人群扩散，投放广告

用户群计算完成后，在人群列表对应的人群单击【导出】，选择【导出至智赢销】，在接下来的页面配置好用户群以及导出字段，单击【开始导出】。

注意：

用户群的用户量大于等于 10,000 以上时，人群扩散效果最佳。

单击【开始导出】后，可以单击【查看任务进度】，人群若成功导出至智赢销，您将会收到通知提醒（上一步骤中配置的提醒方式）。

收到通知提醒后，登录 智赢销数据平台系统 查看对应的人群状态（人群规模 ≥10,000 时方可进一步扩散）。

当人群状态为计算完成时，在顶部导航【人群生成】>【人群扩散】中选择【人群扩散】。

选择对应的人群，单击扩散。

拖动滚动条，选择您需要扩散的人群规模（即您希望投放的目标人群规模）。

如果您有任何疑问，可在 MTA 官网右下角联系我们的客服人员。

网速监控

Android网速监控接口集成

网速监控分为两种类型，一种是在 App 本地提供域名和端口列表进行测速，另一种是在前台配置域名列表，由 SDK 或 App 在适当的时候进行测速，所有的网速监控都是异步操作。

本地指定域名端口测速

在 App 本地指定测速的域名和列表，由开发者决定何时进行测速。

```
void StatService.testSpeed(Context ctx,
Map<String, Integer> domainMap)
```

参数说明：
ctx：页面的设备上下文；
domainMap：待测速的域名和端口列表。

```
Map<String, Integer> map = new HashMap<String, Integer>();
map.put("www.qq.com", 80);
map.put("pingma.qq.com", 80);
StatService.testSpeed(ctx, map);
```

前台指定域名测速

开发者在前台配置待监控的域名和端口列表，由服务器下发到 SDK，然后在 App
需要测速的地方调用以下接口，便会对配置的所有域名进行测速监控，通常 SDK 在 App
启动时会主动测速，如果要在特定的地方测速，需要开发者主动调用本接口。

```
void StatService.testSpeed(Context ctx)
```

参数说明：

ctx：页面的设备上下文

> 注意：
> 主动调用本接口产生网络 I/O 可能会影响用户体验，请慎重使用。

iOS网速监控接口

开发者在前台配置待监控的域名和端口列表由服务器下发到 SDK，然后 SDK 在 App 启动时会主动测速，会对配置的所有域名进行测速监控。

> 注意：
> 本功能会产生网络 I/O。

在线参数

功能介绍

开发者在腾讯移动分析网站上设置 Key-Value
值之后，可以通过接口调用动态获取线上最新的参数值，可用于以下场景：

1. 内容更新，文案、价格、应用欢迎语等；
2. 开关控制，广告的开启和关闭；
3. 简单的逻辑控制，比如当满足一定条件之后更改对应的内容展示；
4. App 自动更新。

Android 代码集成

更新机制

用户在前台配置在线参数并不是实时下发的，而是当 SDK 上报会话统计日志时才会更新，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，产生一个会话，便会更新。

集成代码

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
}
```

iOS 代码集成

接口

```
/**
 获取在MTA前端控制台配置的参数
 调用这个函数前需要在MTA前端控制台中‘应用配置管理项’下的‘自定义参数’中配置才能生效
 首次配置或者更改参数配置后，需要3-5分钟才能生效

 @param key 参数的key
 @param v 没取到参数时返回的默认值
 @return 参数的值或者默认值
 */
- (NSString *)getCustomProperty:(NSString *)key default:(NSString *)v;
```

示例

```
[[MTAConfig getInstance] getCustomProperty:@"参数名称" default:nil];
```

注意：

【参数名称】要与 MTA 管理台配置的参数名称一致。

统计接口监控

功能介绍

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。

Android 接口监控代码集成

代码说明

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。

void StatService.reportAppMonitorStat (
Context ctx, StatAppMonitor monitor)
<span style="font-family:'sans serif', tahoma, verdana,
helvetica;font-size:14px;line-height:1.5;">参数： <span
style="font-family:'sans serif', tahoma, verdana, helvetica;font-size:14px;line-height:1.5;">

调用位置

被监控的接口：StatAppMonitor 方法名列表

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，是可容忍的
  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);
```

iOS 接口监控使用指南

1. 接口

```
/**
 接口统计的枚举值
 */
typedef enum {
  /**
  接口调用成功
  */
  MTA_SUCCESS = 0,

  /**
  接口调用失败
  */
  MTA_FAILURE = 1,

  /**
  接口调用出现逻辑错误
  */
  MTA_LOGIC_FAILURE = 2
} MTAAppMonitorErrorType;


/**
 接口统计的数据结构
 */
@interface MTAAppMonitorStat : NSObject

/**
 监控业务接口名
 */
@property (nonatomic, retain) NSString *interface;
```

```
/**
 上传请求包量，单位字节
 */
@property uint32_t requestPackageSize;

/**
 接收应答包量，单位字节
 */
@property uint32_t responsePackageSize;

/**
 消耗的时间，单位毫秒
 */
@property uint64_t consumedMilliseconds;

/**
 业务返回的应答码
 */
@property int32_t returnCode;

/**
 业务返回类型
 */
@property MTAAppMonitorErrorType resultType;

/**
 上报采样率，默认0含义为无采样
 */
@property uint32_t sampling;
@end

/**
 对网络接口的调用情况进行统计
 参数的详细信息请看接口统计数据结构中的相关说明
```

```
 @param stat 接口统计的数据，详情请看接口统计数据结构的相关说明
 */
+ (void)reportAppMonitorStat:(MTAAppMonitorStat *)stat;

/**
 对网络接口的调用情况进行统计
 并指定上报方式
 参数的详细信息请看接口统计数据结构中的相关说明

 @param stat 接口统计的数据，详情请看接口统计数据结构的相关说明
 @param appkey 需要上报的appKey，若传入nil，则上报到启动函数中的appkey
 @param isRealTime 是否实时上报，若传入YES，则忽略全局上报策略实时上报。否则按照全局策略上报。
 */
+ (void)reportAppMonitorStat:(MTAAppMonitorStat *)stat appkey:(NSString *)appkey
isRealTime:(BOOL)isRealTime;
```

2. 示例

```
-(IBAction) clickNormaltButton:(id)sender{
  MTAAppMonitorStat* stat = [[MTAAppMonitorStat alloc] init];
  [stat setInterface:@"interface1"];
  // ...
  [stat setRetsultType: SUCCESS];
  [MTA reportAppMonitorStat:stat];
}
```

反作弊分析

反作弊介绍

什么是作弊？

对应用没有任何兴趣，通过不正当的手段完成虚假用户的新增、活跃等操作。

为什么反作弊？

在过去，由于没有形成对流量作弊的认识，所以只要广告/推广/活动等在预算下正常工作，每个人都不会对流量有所怀疑。但正是这种做法，给予了虚假流量可乘之机。导致供应链上的每个人都受到影响。

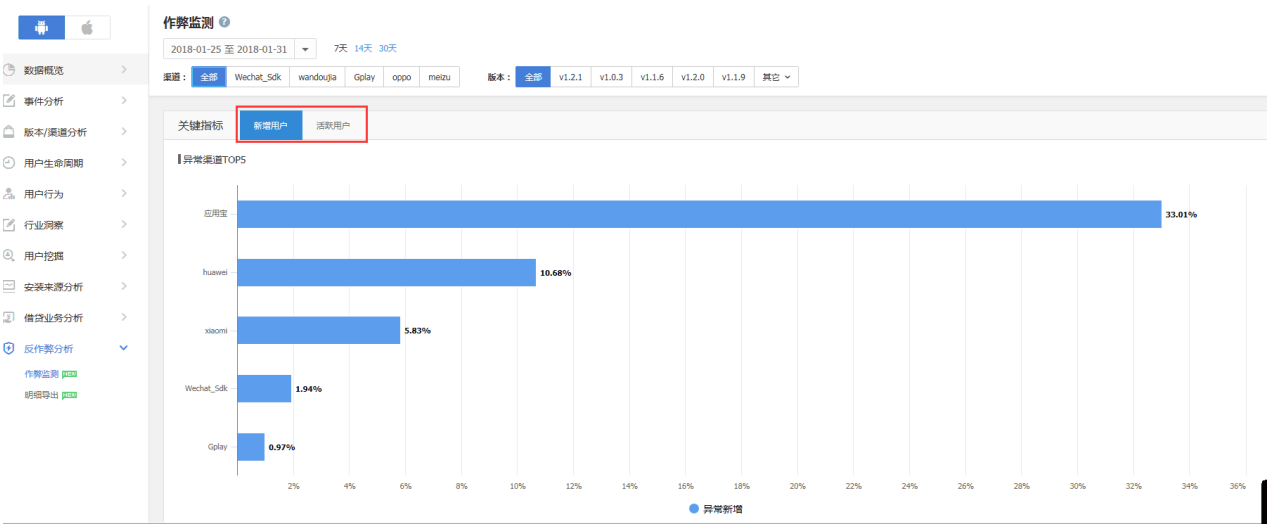
- 造成直接利益损失：由于流量欺诈可产生直接金钱损失
- 造成数据虚假，看到的是虚假的数据，无法还原业务真实面貌，或者丧失新的商机;行业形象、可信度的损

MTA流量反作弊产品介绍

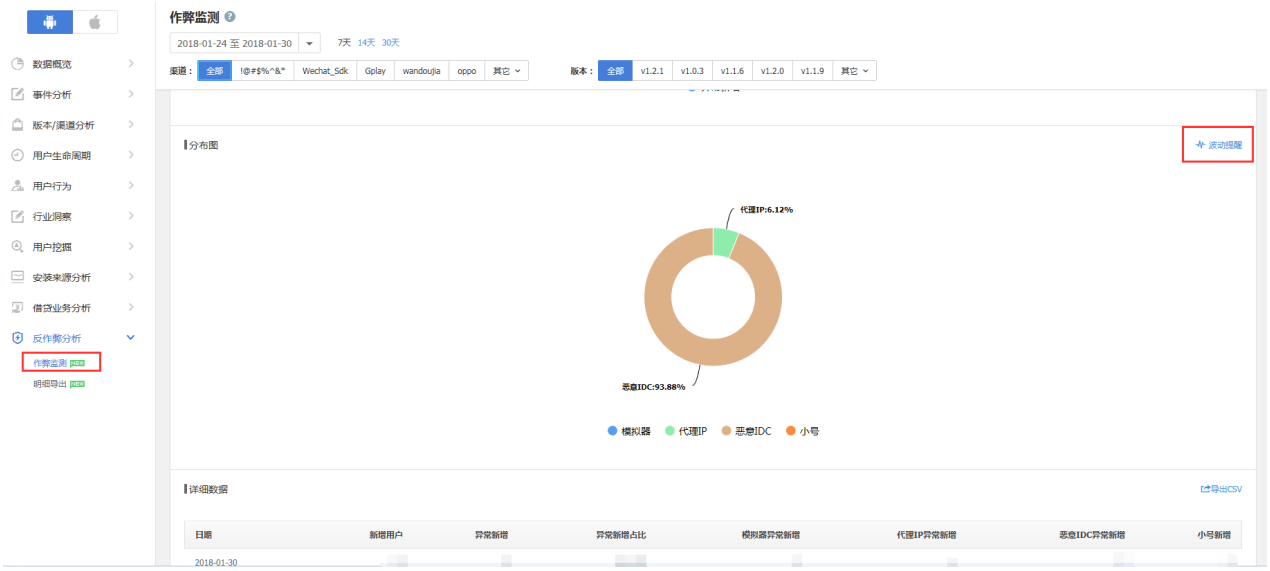
MTA反作弊结合腾讯数据库及专业的反作弊能力，为移动应用提供全面的移动作弊保护。积极检测作弊类型，识别异常渠道，提前预警，从而帮助营销人员及时采取行动。

作弊监测

1. 异常渠道 top 排行：根据作弊规则识别异常渠道，直观展示异常 top5 渠道，帮助运营推广人员评估渠道质量，过滤无效流量。

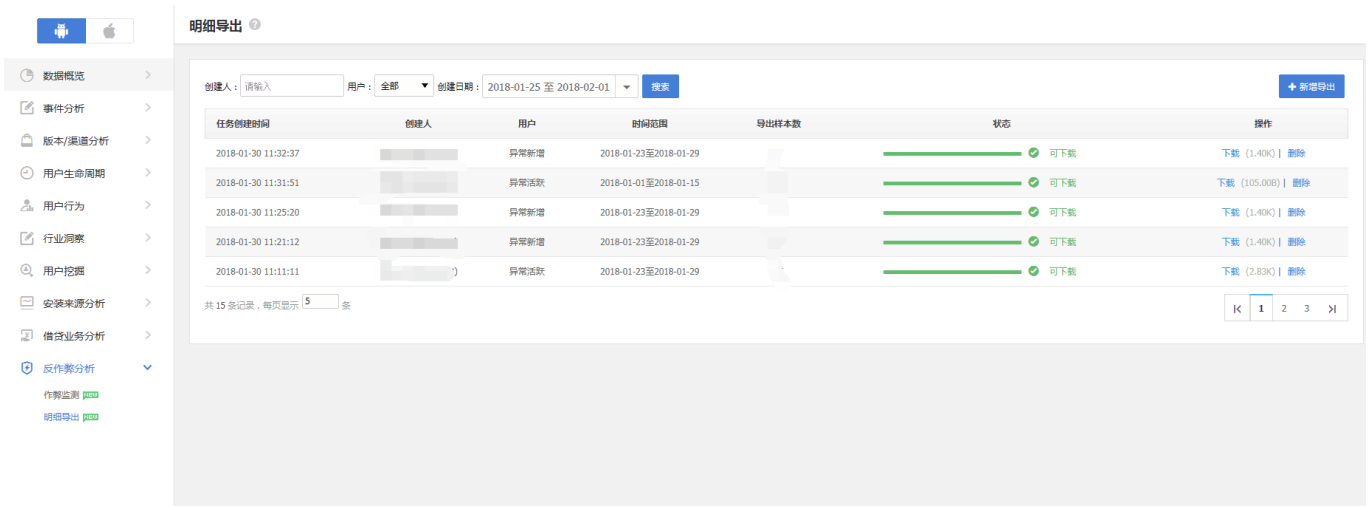


2. 多维度识别异常用户：从小号、模拟器、恶意代理ip、自动机等多个维度识别异常新增/活跃用户；列举具体异常类型，让流量回归真实原貌.



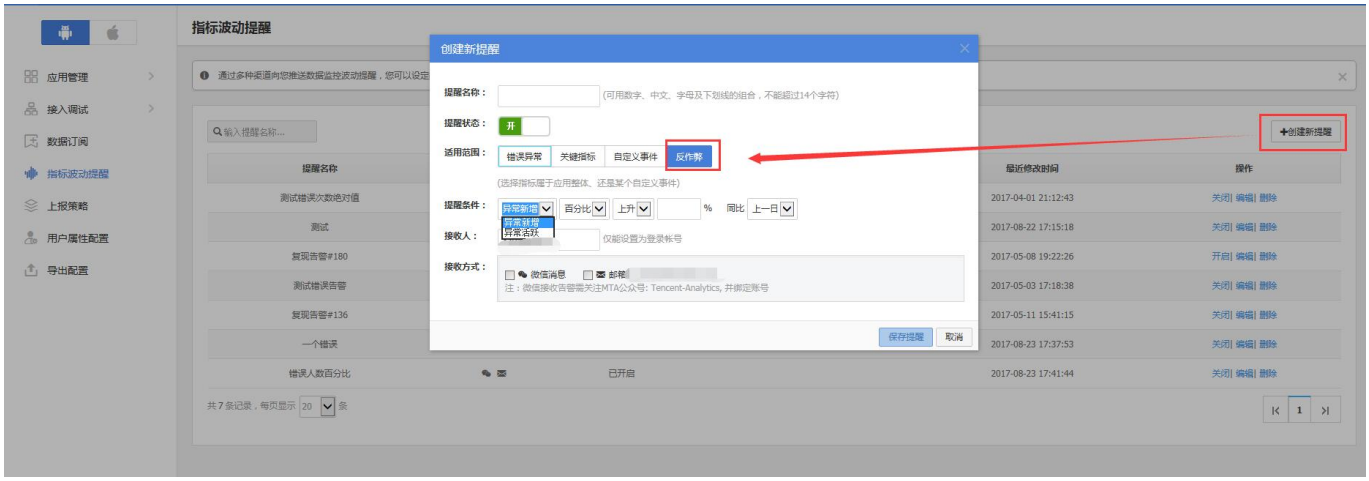
作弊数据导出

支持设备id、ip等数据样本日志下载，以便运营推广人员进行判断，加深反作弊甄别的深度和精细度。



作弊监控告警

点击【作弊监测】中的【波动提醒】，便可对【异常新增】及【异常活跃】设置监控告警。



使用和注意事项

如何使用

注意：

当前反作弊处于内测阶段，如需申请，直接在应用处的

【反作弊分析】>>【作弊监测】，点击立即开通，申请开通即可。

审核通过后，次日即可看到数据

如未集成 sdk，请先集成 sdk。详细操作请参考IOS SDK接入指南 或 Android SDK 接入指南 。

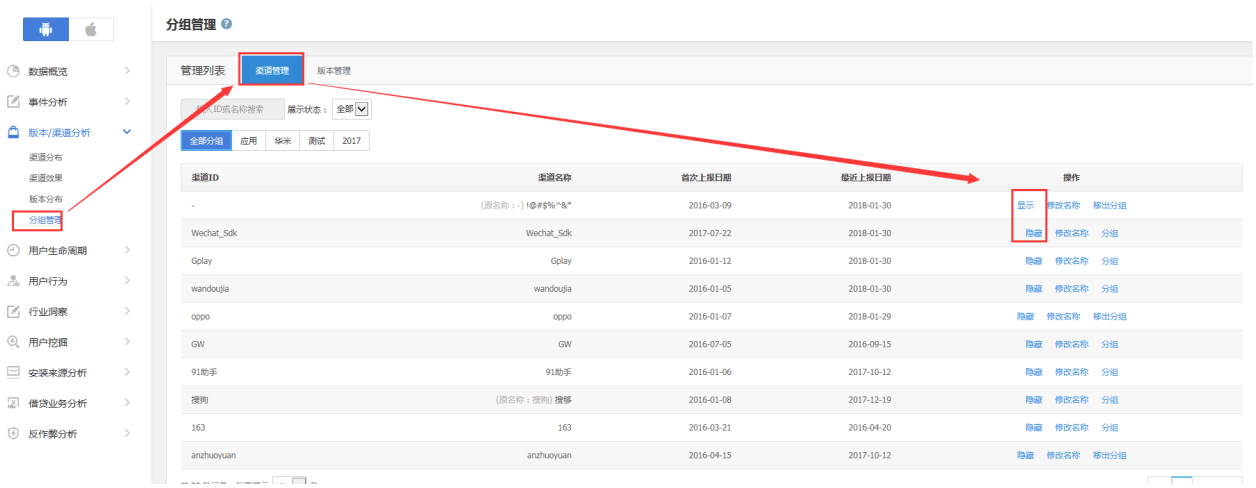
注意事项

1. 不是所有的类型都能查看

   因为异常类型的获取匹配依赖用户账号的上报（尤其是手机号码）。为了保证异常类型的全面及数据的准确性，建议开发者上报账号（点击查看如何上报账号）。

2. 在异常渠道中存在的渠道，可能在报表上的渠道维度上没有

   一般出现这种情况是因为对渠道做了隐藏处理，所以在渠道维度上的维度不是全部的维度。如需同时查看只需取消隐藏即可


分组管理
管理列表
渠道管理
版本管理
展示状态：
全部
全部分组
应用
华米
测试
2017
渠道ID
渠道名称
首次上报日期
最近上报日期
操作
数据概览
事件分析
版本/渠道分析
渠道分布
渠道效果
版本分布
分组管理
用户生命周期
用户行为
行业洞察
用户挖掘
安装来源分析
借贷业务分析
反作弊分析
Wechat_Sdk
Gplay
wandoujia
oppo
GW
91助手
搜狗
163
anzhuoyuan
显示
隐藏
修改名称
移出分组
分组

安装来源分析

功能简介

功能介绍

安装来源分析，通过集成 MTA 的 SDK（Beta SDK 下载链接），可以方便的统计您的推广渠道访问量与点击量，同时可以统计到不同渠道的安装量、转换率等，方便用户评估各渠道的推广效果，从而调整自己的推广计划，详情请见 应用分析。

申请方式：发邮件或者 QQ 来申请使用此功能。

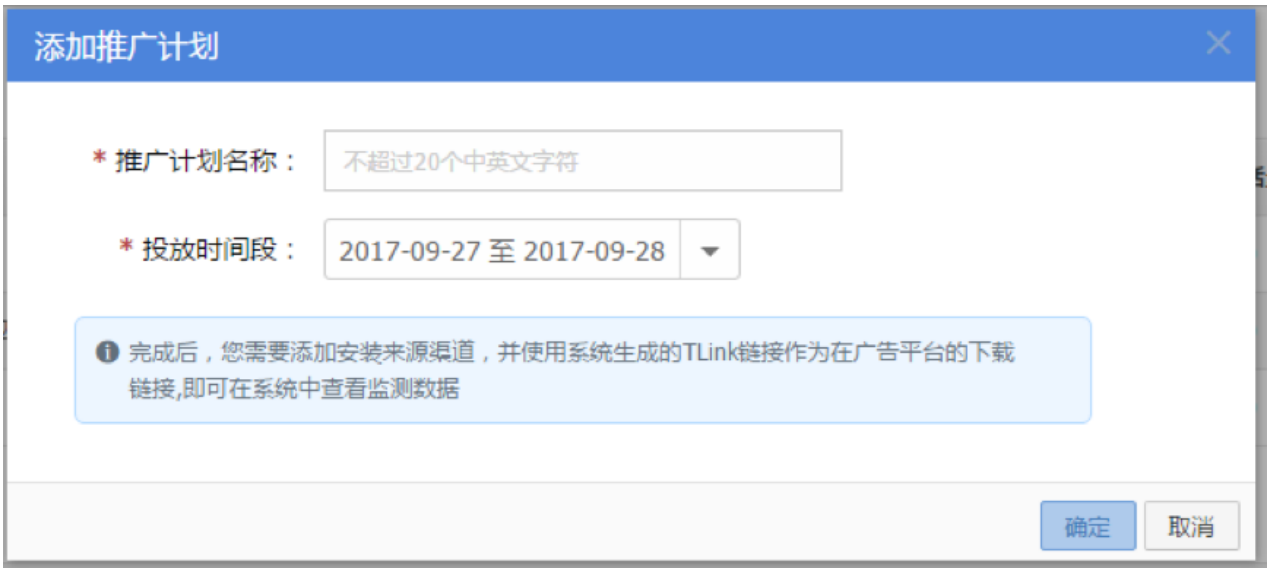
功能特色

1. 支持全平台安装来源分析，无论是 Android 还是 iOS，甚至只是 H5 落地页，都可以统计到用户在每一层的访问，从而全面的统计到推广的安装效果；
2. 支持多种形式的落地链接，包括落地 H5（需要集成 JS SDK ），应用市场(包括 App store 以及应用宝在内的各种安装市场)，直接 APK 下载；
3. 支持实时效果统计，完成集成后，推广效果即刻就可以在实时页面看到，用户可以根据实时效果调整自己的推广策略；
4. 即将推出从广告点击到应用内有效转化的全链路分析。

Tlink 解释：当用户配置完一个推广渠道的时候，MTA 会根据您提供的应用落地链接，生成一个用于分析统计的链接给到您，您需要将此链接配置到对应投放广告商的投放系统上，即可使用 MTA 统计对应广告商的安装转化效果。

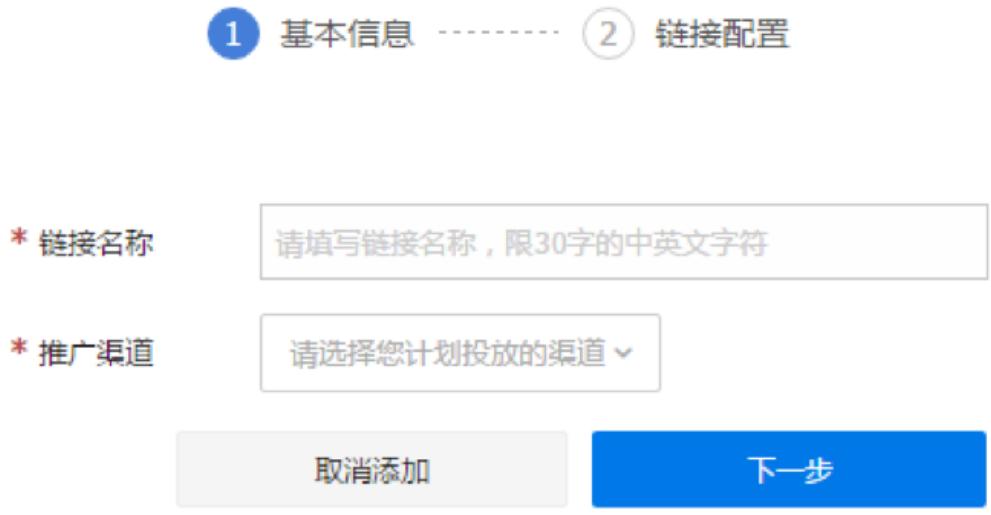
操作指引

1. 集成含有安装来源分析的 SDK （SDK下载）；
2. 创建推广计划 ；



3. 单击推广渠道配置进行相应的配置操作；

4. 配置安装渠道；



注意：

【推广渠道】是指您在推广计划中，计划进行广告投放的平台 MTA

支持多种落地链接，您可以根据您的需要进行对应的配置。

5. 完成配置后，单击 Tlink链接中的【查看】，即可获取 Tlink，用以广告投放。

技术原理

iOS

iOS 通过连接参数区分投放渠道标识，通过 iOS 的共享存储机制来传递渠道标识和用户标识，从而串联 H5 的下载 和 Native 的安装/打开；

Android

Android 是通过加工后的短链接来收集渠道标识并上报，利用 User-Agent+IP 的方式来标识设备，从而串联 H5 的下载 和 Native 的安装/打开 。

风险说明：应用在 Android 和 iOS 平台的下载链接经 MTA 加工形成 Tlink，Tlink 依赖腾讯的服务器以稳定运转，因自然灾害等不可抗力因素导致服务器出现故障，由此影响 Tlink 无法正常跳转的风险无法完全避免。

Android 使用文档

如何启用 TLink 功能

1. 【必选】要在 MTA 前台开通并配置相关的推广计划；
2. 【必选】在 App 的入口处，一般为 Application 或 MainActivity 的 onCreate() 调用“StatConfig.setTLinkStatus(true);”开启TLink功能；
3. 【可选】若有接入腾讯 TBS 浏览服务 SDK，请在主线程调用“QbSdk.initX5Environment()”方法后面添加“StatConfig.invokeTBSSdkOnUiThread(context);”。

Android落地页 JS SDK 配置

以下为嵌入落地页的 JS SDK 格式，必须填写替换参数：
$download_btn_id：下载按钮的 ID
$app_key：MTA 管理台中 Android 的 APP KEY

```
<script>
var _mta_btn_id = '$download_btn_id';
 (function() {
  var mta = document.createElement("script");
  mta.src = "//pingjs.qq.com/mta/channel_stats.js?v1";
  mta.setAttribute("name", "MTA_CHANNEL");
  mta.setAttribute("app_key", "$app_key");
  var s = document.getElementsByTagName("script")[0];
  s.parentNode.insertBefore(mta, s);
})();
</script>
```

使用示例：

```
<a id="download_app">下载应用</a>

<script>
var _mta_btn_id = 'download_app';
```

```
(function() {
  var mta = document.createElement("script");
  mta.src = "//pingjs.qq.com/mta/channel_stats.js?v1";
  mta.setAttribute("name", "MTA_CHANNEL");
  mta.setAttribute("app_key", "IB7ZRJ6V8S1T");
  var s = document.getElementsByTagName("script")[0];
  s.parentNode.insertBefore(mta, s);
})();
</script>
```

iOS 使用文档

工程配置

1. 统计安装来源（URL Scheme）
   XCode 中的 URL Types 中增加一条 URL Scheme 配置，Role 是 Viewer，URL Schemes 的配置后续在 JS SDK 的初始化会用到。
2. 如果已安装 App，直接打开（非必需，通过 Universal Links 技术）。
   i. 首先你需要有一个 https 的域名，例如 domain.com；
   ii. Uninversal Links 需要的 json 文件：apple-app-site-association，可以从 MTA 管理台生成；
   iii. 把 apple-app-site-association 上传到 domain.com 根目录（iOS 系统会自动从

   https://domain.com/apple-app-site-association

   进行访问）；
   iiii. XCode 的 capabilities 增加 Domains 的配置，例如 applinks:domain.com。
3. 下载页面的修改
   具体请参考 MTA 管理台中关于 JS SDK 的说明。
4. 备注
   因为用到了 keychain，如果遇到相关编译不过的问题，请在项目中引用 Security.framework。

接口调用

1.AppDelegate 中的改动
在 MTA 的初始化之后增加[Installtracker getInstance]

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
  // Override point for customization after application launch.

  [[MTAConfig getInstance] setSmartReporting:YES];
  [[MTAConfig getInstance] setReportStrategy:MTA_STRATEGY_INSTANT];

  [[MTAConfig getInstance] setDebugEnable:YES];
```

```
  [MTA startWithAppkey:@"I2E3KXDU1E2W"];

  [Installtracker getInstance];

  return YES;
}
```

在 handleOpenURL 中增加调用

```
- (BOOL)application:(UIApplication *)application handleOpenURL:(NSURL *)url{

  [[Installtracker getInstance] handleOpenURL:url];

  return true;
}
```

通用链接，如果 App 已经安装，直接打开

```
- (BOOL)application:(UIApplication *)application continueUserActivity:(NSUserActivity *)userActivity
restorationHandler:(void (^)(NSArray * _Nullable))restorationHandler
{
  BOOL result = [[Installtracker getInstance] checkIsFromMTARefer:userActivity];
  return result;
}
```

2.在 App 进入的第一个 ViewController 的修改

viewDidLoad 中添加以下代码

```
- (void)viewDidLoad {
  [super viewDidLoad];
```

```
  [[Installtracker getInstance] startByViewDidload];

}
```

3.如果有自己的中间页，不使用 MTA 管理台生成的话，需要单独接入 JS SDK
并设置中间页的地址，didFinishLaunchingWithOptions 的初始化修改如下：

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary
*)launchOptions {
  // Override point for customization after application launch.

  [[MTAConfig getInstance] setSmartReporting:YES];
  [[MTAConfig getInstance] setReportStrategy:MTA_STRATEGY_INSTANT];

  [[MTAConfig getInstance] setDebugEnable:YES];

  [MTA startWithAppkey:@"I2E3KXDU1E2W"];

  [[Installtracker getInstance] setChannelUrl:@"http://domain.com/test/download.html"];

  return YES;
}
```

说明：
i. 注意

http://domain.com/test/download.html

这代表着你投放的网址可能是

http://domain.com/test/download.html?ADTAG=youradtag

、

http://domain.com/test/download.html?ADTAG=youradtag2

、

http://domain.com/test/download.html?ADTAG=youradtag3

等。
ii. 请替换

http://domain.com/test/download.html

为实际的中间页地址。
iii. JS SDK 的使用参考 MTA 管理台相关页面。

iOS 落地页 JS SDK 配置

以下为嵌入落地页的 JS SDK 格式，必须填写替换参数：
$download_btn_id：下载按钮的 ID
$app_key：MTA 管理台中 iOS 的 APP KEY
$URL_Scheme：ios app的URL Scheme

```
<script>
var _mta_btn_id = '$download_btn_id';
 (function() {
  var mta = document.createElement("script");
  mta.src = "//pingjs.qq.com/mta/channel_stats.js?v1";
  mta.setAttribute("name", "MTA_CHANNEL");
  mta.setAttribute("app_key", "$app_key");
  mta.setAttribute("app_flag", "$URL_Scheme");
```

```
  var s = document.getElementsByTagName("script")[0];
  s.parentNode.insertBefore(mta, s);
})();
</script>
```

使用示例：

```
<a id="download_app">下载应用</a>

<script>
var _mta_btn_id = 'download_app';
(function() {
  var mta = document.createElement("script");
  mta.src = "//pingjs.qq.com/mta/channel_stats.js?v1";
  mta.setAttribute("name", "MTA_CHANNEL");
  mta.setAttribute("app_key", "IB7ZRJ6V8S1T");
  mta.setAttribute("app_flag", "mtaApp");
  var s = document.getElementsByTagName("script")[0];
  s.parentNode.insertBefore(mta, s);
})();
</script>
```

单链接双平台JS_SDK配置方式

如果您需要在微信、朋友圈、二维码、短信等场景下进行推广，且推广地址为活动页面，可以使用以下方案：

在落地页（活动页面）嵌入 JS SDK 时，可以在代码中根据当前访问页面的设备系统(参考 HTTP Useragent)将 $adtag 设置成您在 MTA 获得的 TLink 中的后缀 ADTAG 值，同时将 $app_key 也设置成在 MTA 中获得的不同平台的 Appkey 即可。

```
<script>
var _mta_btn_id = '$download_btn_id';
 var _mta_adtag = '$adtag';
 (function() {
  var mta = document.createElement("script");
  mta.src = "//pingjs.qq.com/mta/channel_stats.js?v1";
  mta.setAttribute("name", "MTA_CHANNEL");
  mta.setAttribute("app_key", "$app_key");
  mta.setAttribute("app_flag", "$URL_Scheme"); (Android不需要该行)
  var s = document.getElementsByTagName("script")[0];
  s.parentNode.insertBefore(mta, s);
  })();
</script>
```

使用示例：

在 MTA 管理台 Android 和 iOS 中各配置两个 Tlink：

- iOS:

  http://xxxxx.com?ADTAG=111

- Android:

  http://xxxx.com?ADTAG=222

其中 iOS 的 Appkey 为 AAAAAAAA，Android 的 Appkey 为 BBBBBBB（见管理台->配置管理->应用信息页面） 则在活动页面中判断，若当前设备为 iOS，将 $adtag 设置成 111，同时 $app_key 设置为 AAAAAAAA；若当前设备为 Android, 将 $adtag 设置成 222，同时 $app_key 设置为BBBBBBB。

（如果需要多个链接跳转同一个落地页，同时每个链接都需要统计 Android 和 iOS，则可以在跳转至落地页前将 ADTAG 附在 URL 的参数中）

行业解决方案

借贷业务分析

借贷业务分析介绍

借贷业务分析是腾讯移动分析为互联网金融行业量身定制了一套专业的免费的行业数据解决方案，旨在助力金融互联网不断创新，打造智慧金融。

借贷业务分析介绍

借贷业务分析是MTA扎根借贷行业特点，为互联网金融行业量身定制的行业数据解决方案，旨在帮助用户降低获客成本、提升转化效率，推动业务实现健康增长。

点击体验>>

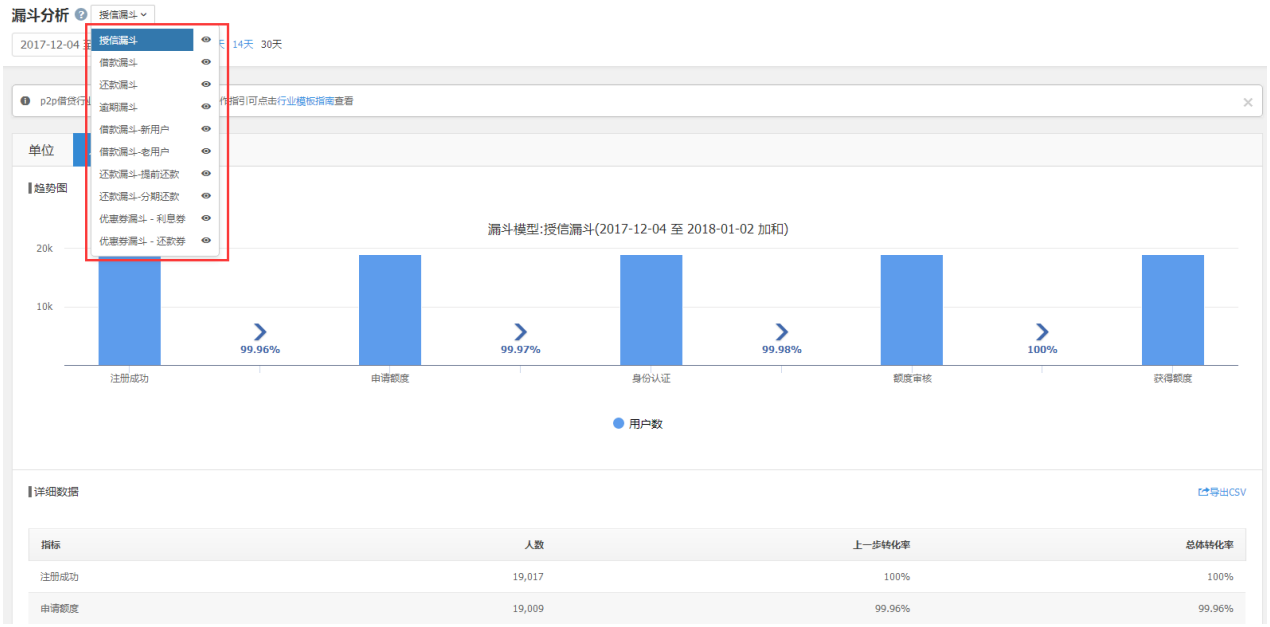
借贷行业特点	特点说明
更关注与业绩相关的业务数据	注册用户对于借贷应用来说是一个小开始，一般以开户用户/借款用户来计算ROI，以成功借款金额制定KPI。
获客成本高于其他行业; 有效（到开户借款）转化率低于其他行业	借贷行业平均上千元每人，其他行业几百; 转化率如果提高1%，将增加几十万（甚至几百万）的贷款金额
业务风险	其他行业，用户交易完后无需再关注本次交易情况，但借贷交易用户存在风险，用户借款后可能会不还款或者用户是中介机构套现，完成借款后需要关注是否有正常还款、逾期率等
消费状态连贯，但反馈链条长	借款用户均有开户、借款、还款状态。但开户、借款均需风控审核，使用到成功借款时间差较长（一般为2-7天）；而应用想了解用户还款、逾期情况需在借款30天后才可了解
使用频率低，较难了解用户的特征	贷款行业的用户使用产品的目的性很强，缺钱的时候才会使用产品，借完款之后没有再次借款需求基本上不会再来（提供自动扣款服务频率更低）

针对以上所述的特点，MTA在下表依次排列了各种解决方案。

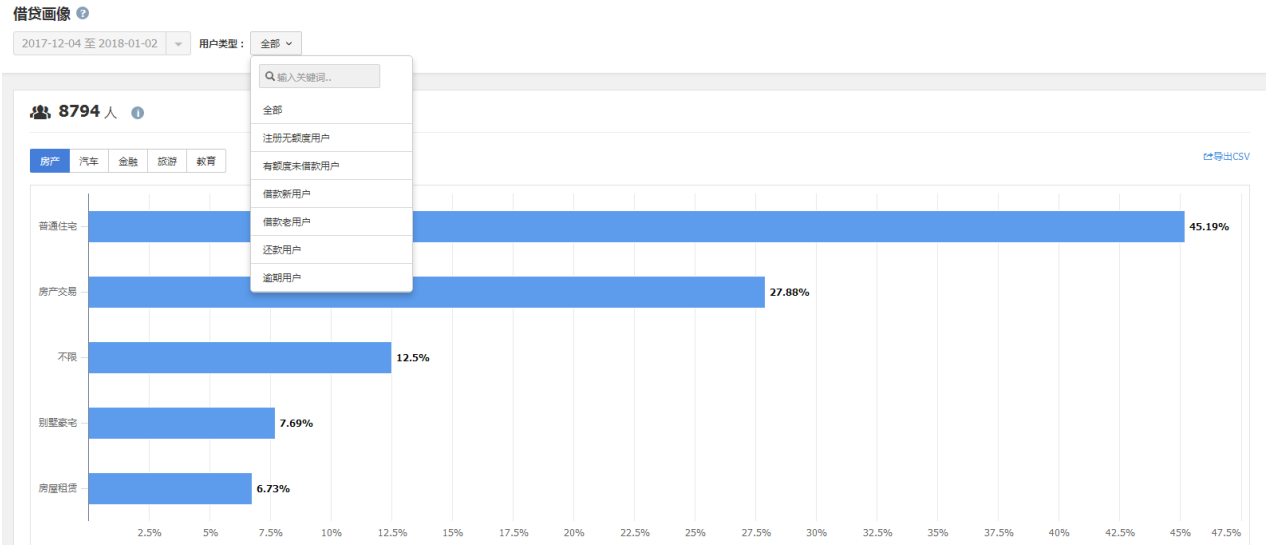
MTA借贷行业解决方案	方案说明

MTA借贷行业解决方案	方案说明
精心选取与借贷业务KPI/ROI相关的关键指标	从整体业绩增长目标出发，帮助业务运营者关注围绕目标的重要数据指标，关注整个借款产品层面的业绩健康，保持一定的业务大局观，以推动业绩增长为目的进行运营。
获客成本高转化率低	打磨10个借贷业务漏斗，包含重要的业务转化点，帮助用户快速了解定位问题，针对问题寻找优化方案，提高转化率，降低获客成本。
业绩与风控指标数据全面兼顾	完整的还原一个用户借款时的真实步骤场景。运营可直观关注风险相关数据（审核率/逾期率），了解风险情况，协助优化风控不合理的规则。风控也能关注业绩数据，结合业务用户制定较合理的风控规则（避免挡住了坏人同时影响了业绩的规则），形成密切的业务闭环。通过数据促进整个借贷业务的健康发展
用户各状态（开户、借款、还款、逾期）数据均支持查看下载	且漏斗支持30天跨天去重，在业务数据表中可结合反馈周期（如7天、30天）同时查看多个状态的数据情况，一目了然的了解近期业务完整链条数据情况
借贷行业垂直画像	为借贷行业应用提供39个画像标签，与业务数据耦合（如注册无额度、有额度未借款等），协助应用了解用户特征，可结合借贷画像设计更适合人群特征的借款产品、营销活动或制定更符合人群特征的风控规则

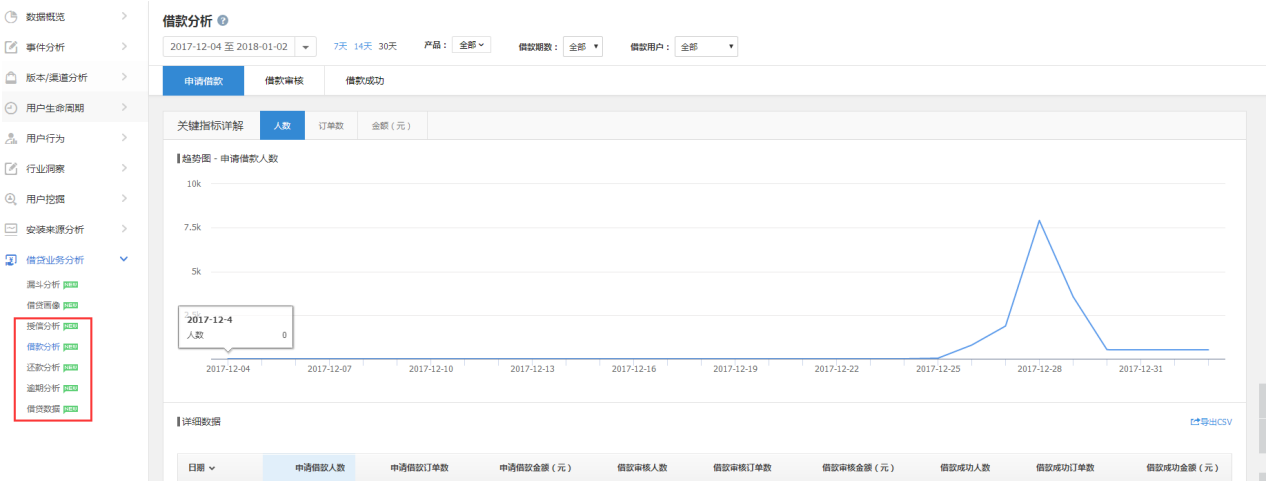
- 打磨10个借贷业务漏斗



- 为借贷行业提供39个画像标签，与业务数据耦合



- 用户各状态（开户、借款、还款、逾期）数据均支持查看下载



如何使用

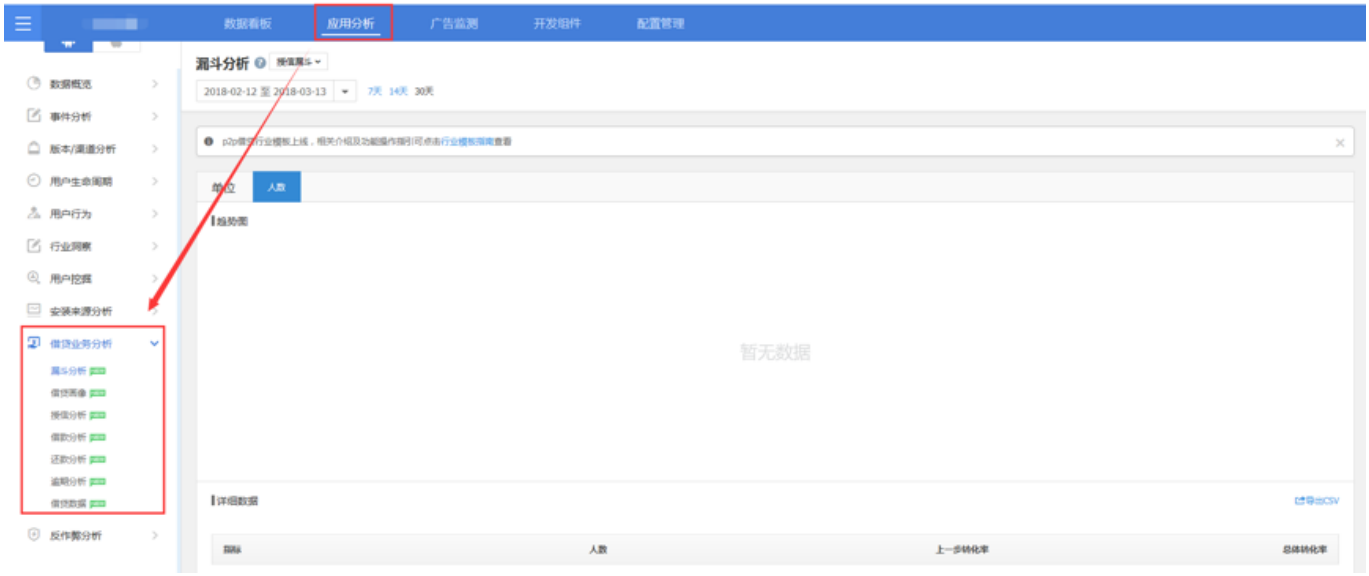
新增应用

在申请时新增应用时，服务类型勾选【 借贷专用统计】即可

已存在应用

在管理台处的卡片，点击【+借贷】图标，即可在移动分析处看到【借贷业务分析】的页面

添加后的前端页面展示如下：

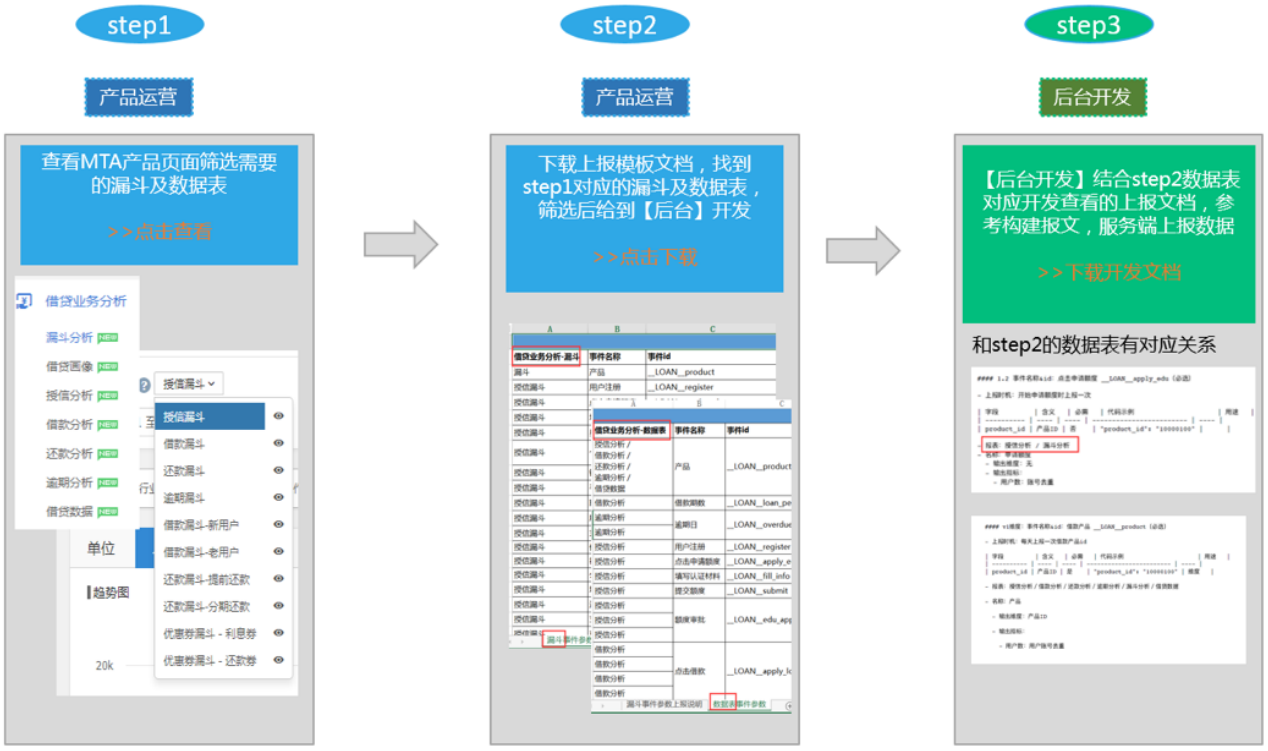


如暂时无统计需求，管理员可在管理台点击，统计即暂停

注意：

请谨慎操作，删除后行业数据停止统计，相关页面将消失

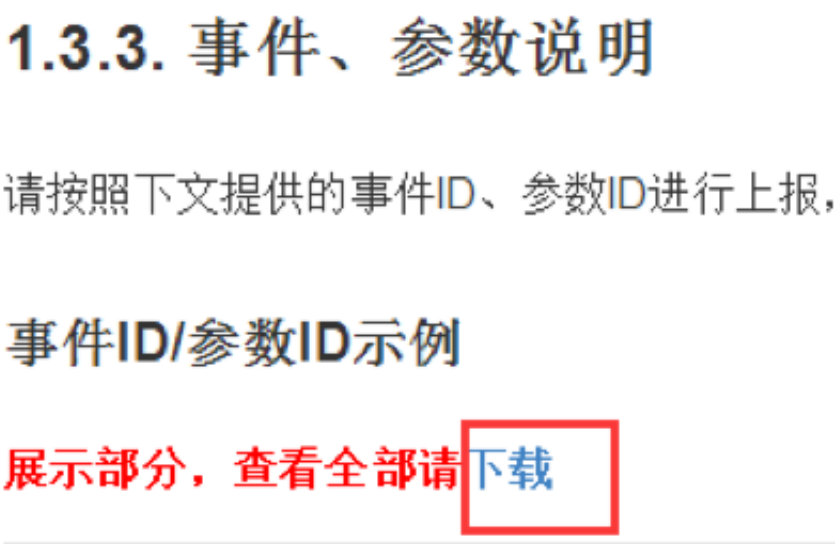
如何查看借贷业务分析的数据



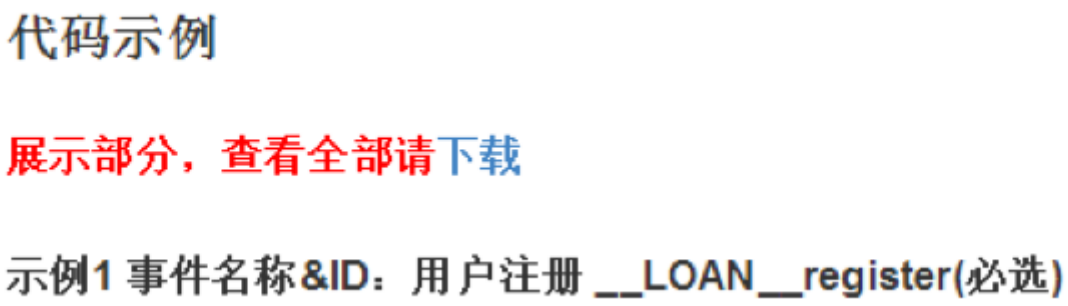
上图链接说明如下：

点击查看

点击下载：同如图下载链接



下载开发文档：同如图下载链接



Tips:

1. 建议后台上报，不建议前端上报，因为前端上报会暴露appkey等，容易让别人拿到伪造上报。
2. 产品运营需要对照数据产品页面，先筛选出对应的数据表，然后再在下载的文档中选出对应漏斗、数据表及其对应的事件与数据指标。
3. 数据指标和事件名称是有一些出入，主要考虑是数据指标主要和运营产品常用的概念对齐，事件名称是方便技术同学进行埋点的理解。

上报数据

上报方式

服务器端上报日志（仅支持“自定义事件”类型的数据），参考下文中的构建报文

> 注意：
> 当前仅支持HTTP协议的上报接口，参考下文中的上报到MTA

构建报文

构建JSON格式的日志，可包含表格中的字段。其中标记为必选的字段必须上报，否则日志无效。

用JSON数组的方式上报多条日志，单条日志也需要加 "[...]" 标记。推荐多条日志合并上报，提高传输效率。

字段ID	含义	是否必选	示例
ky	(系统字段)MTA分配的APP KEY	必选	"if8888mta"
ts	(系统字段)当前UNIX时间戳	必选	1512057600
et	(系统字段)日志类型	必选(填1000)	1000
cui	账号ID	必选	"13823965426"
ei	事件ID	必选	"__LOAN__bind_card"
kv	事件参数(JSON格式)	可选	{"status": 1, "product_id": "1000234"}

报文示例

```
[
 {
 "ky": "if8888mta",
 "ts": 1512057600,
 "et": 1000,
```

"cui": "13823965426",
"ei": "__LOAN__bind_card",
"kv": {"status": 1, "product_id": "1000234"}
}, {
"ky": "if8888mta",
"ts": 1512057600,
"et": 1000,
"cui": "13823965426",
"ei": "__LOAN__card_confirm",
"kv": {"product_id": "1000234"}
}
]

上报到MTA

接口类型：HTTP POST

编码支持：UTF-8

上报示例

```
curl -d '[{"ky": "if8888mta","ts": 1512057600,"et": 1000,"cui":
 "13823965426","ei": "__LOAN__bind_card","kv": {"status": 1, "product_id":
"1000234"}}]' http://pingma.qq.com/mstat/submit
```

事件、参数说明

注意：

请按照下文提供的事件ID、参数ID进行上报，且务必参照上报代码示例进行上报，否则日志无效

事件ID/参数ID示例

展示部分，查看全部请下载

漏斗	事件名称	事件id	是否必选	上报时机	事件参数id	参数含义	是否必选	数据指标
漏斗	产品	__LOAN__product	必选	每天上报一次借款产品id	product_id	产品id	必选	产品维度
授信漏斗	用户注册	__LOAN__register	必选	用户注册完成时上报一次	status	注册状态	必选	注册成功
授信漏斗	点击申请额度	__LOAN__apply_edu	必选	用户点击申请额度按钮时上报一次	product_id	产品id	可选	申请额度
授信漏斗	填写认证材料	__LOAN__fill_info	必选	用户开始填写额度申请认证资料时上报一次	product_id	产品id	可选	授信认证
授信漏斗	提交额度	__LOAN__submit	必选	用户点击“提交审核申请”按钮时上报一次	product_id	产品id	可选	额度审核

代码示例

展示部分，查看全部请下载

示例1

事件名称&ID：用户注册__LOAN__register(必选)

- 上报时机：注册完成时上报一次

字段	含义	必需	代码示例	用途
status	注册状态	是	"status":1	过滤（1：“成功”；

字段	含义	必需	代码示例	用途
				0：“失败”）

- 报表：授信分析/漏斗分析
- 名称：注册成功
- 输出维度：无
    - 输出指标
        - 用户数：用户账号去重

示例2

事件名称&ID：点击申请额度__LOAN__apply_edu(必选)

- 上报时机：开始申请额度时上报一次

字段	含义	必需	代码示例	用途
product_id	产品ID	否	"product_id":"10000100"	维度

- 报表：授信分析/漏斗分析
- 名称：申请额度
- 输出维度：无
    - 输出指标：
        - 用户数：用户账号去重

接口返回

返回报文为JSON格式，包含ret和msg两个字段

- ret字段：0表示接收成功，其他表示接收失败
- msg字段：如接收失败，这里存放出错原因（如 invalid appkey）

错误码说明

错误码	错误码含义
invalid appkey	错误的appkey
Json parse error	Json解析错误
Gzip error	Gzip解压失败

接收成功并不意味着日志被成功处理。如果缺失事件参数说明中的必选字段，日志仍然可能无效。

上报接收成功后次日就可查看借贷分析的数据

其他说明

重点借贷指标概念

新/老用户

以借款成功作为判断新用户的标准；一个用户一旦借款成功后，再次申请借款为老用户

订单数

借款时生成的订单，根据订单id（订单号）进行统计

账单数

借款订单分几期，一般就生成几个账单id（账单号），根据账单id进行统计；还款时根据账单进行还款（借款期数若为不分期，按照1个账单上传）

最后还款日

账单确定的最后还款的日期；用户首次借款后生成，日期固定（如每月的15号），与用户的账号绑定。最后还款日后还未还款的定义为逾期

逾期日

当前日期-最后还款日，逾期日超过90天定义为坏账。只取180天

逾期金额

借款人从逾期账单开始尚未还款的全部剩余金额

广告效果监测

概览

功能介绍

MTA 广告效果监测（原安装来源分析），通过集成 MTA SDK，在不同的广告中使用 MTA 的监测代码，可以帮助用户统计 App 来自不同渠道的广告效果，同时可以分析不同来源用户的后续应用内行为，优化广告投放效果，提升广告投资回报率！

功能目前处于内测试用中，如需申请，请发送邮件至 dtsupport@tencent.com ，告诉我们您的应用名称、APPID、联系人、联系方式，方便我们快速为您开通！

快速一览

SDK集成说明

功能操作说明

效果指标一览

广告平台配置方法

常见问题FAQ

指标描述

指标名	描述
曝光	所选时间段广告被曝光的次数
点击(PV)	所选时间段广告被点击的次数
去重点击（UV）	所选时间段点击广告的唯一设备数
激活	所选时间段内点击广告或在激活归因窗口期内激活的设备数，激活设备指历史首次安装并打开应用的设备
激活转化率	所选时间维度下，激活设备数/去重点击的设备数(UV) * 100%
注册率	所选时间端内，注册设备数/激活设备数 * 100%
注册	所选时间段内激活并注册了的设备数
DAU	所选时间段最后一天（不包括当日），已激活且在此日

指标名	描述
	仍然活跃的设备数
WAU	所选时间段最后一周（未满一周仍计算），已激活且在此月仍然活跃的设备数
MAU	所选时间段最后一月（未满一月仍计算），已激活且在此月仍然活跃的设备数
次日留存	激活设备在激活第2日仍在使用该应用的用户为留存用户，这些用户占激活设备的比例即为此日留存率；此处次日留存为所选时间段内平均次日留存
3日留存	激活设备在激活第3日仍在使用该应用的用户为留存用户，这些用户占激活设备的比例即为3日留存率，此处3日留存为所选时间段内平均 3 日留存
7日留存	激活设备在激活第7日仍在使用该应用的用户为留存用户，这些用户占激活设备的比例即为7日留存率，此处7日留存为所选时间段内平均 7 日留存
30日留存	激活设备在激活第30日仍在使用该应用的用户为留存用户，这些用户占激活设备的比例即为30日留存率，此处30日留存为所选时间段内平均30日留存
推广消费(元)	当前活动/渠道/活动组的总花费
单用户激活成本	推广消费/活动激活用户
付费设备	所选时间段内，激活、完成注册且产生了付费的设备数
付费率	所选时间段内，激活、完成注册且付费的设备数/激活设备数
LTV(0日、7日、15日、30日)	所选时间范围内，新激活的用户中N日内累计产生的收入/激活用户数,所选时间范围内的激活用户激活未满N日的不纳入计算(收入单位存在多币种情况下,不自动换算,分币种统计显示)
事件X(设备数)	所选时间段内，激活并触发了事件的设备数
事件X(次数)	所选时间段内，激活的设备触发了事件的次数

配置说明

配置说明

适用范围：

本文档提供关于使用MTA广告效果监测在投放广告时，在不同广告平台需进行的操作说明。

如果您对使用的平台对接方式有疑问，请咨询 dtsupport@tencent.com;

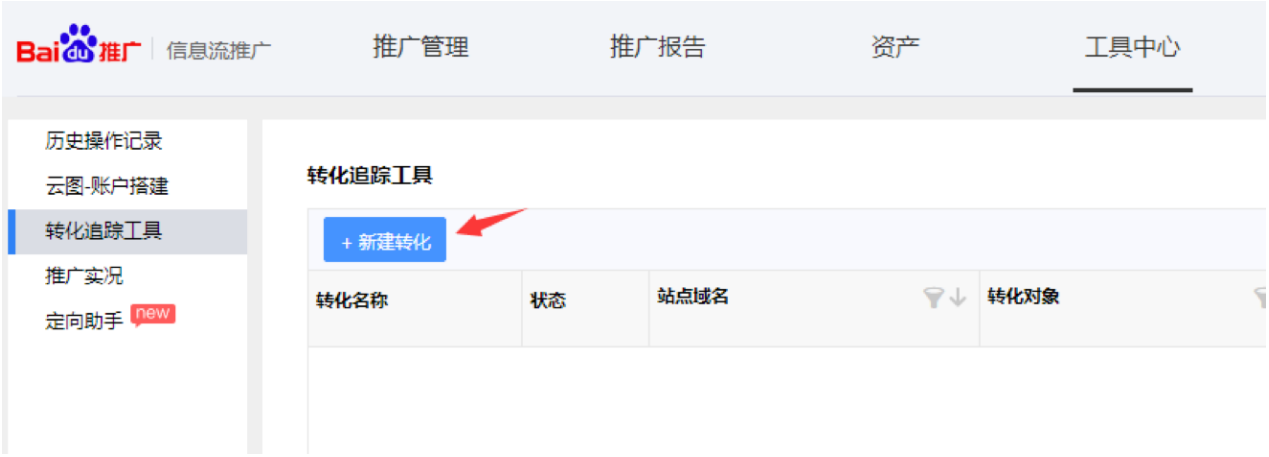
平台列表：

- 百度信息流
- 今日头条

百度信息流

百度信息流推广配置

1. 在百度推广管理台进入【 信息流推广 】>> 【 工具中心 】>> 【 转化追踪工具 】，点击【 新建转化 】



2. 选择【 应用下载 】>>【 立即使用 】

3. 按步骤执行操作，完成后点击【提交】

注意：

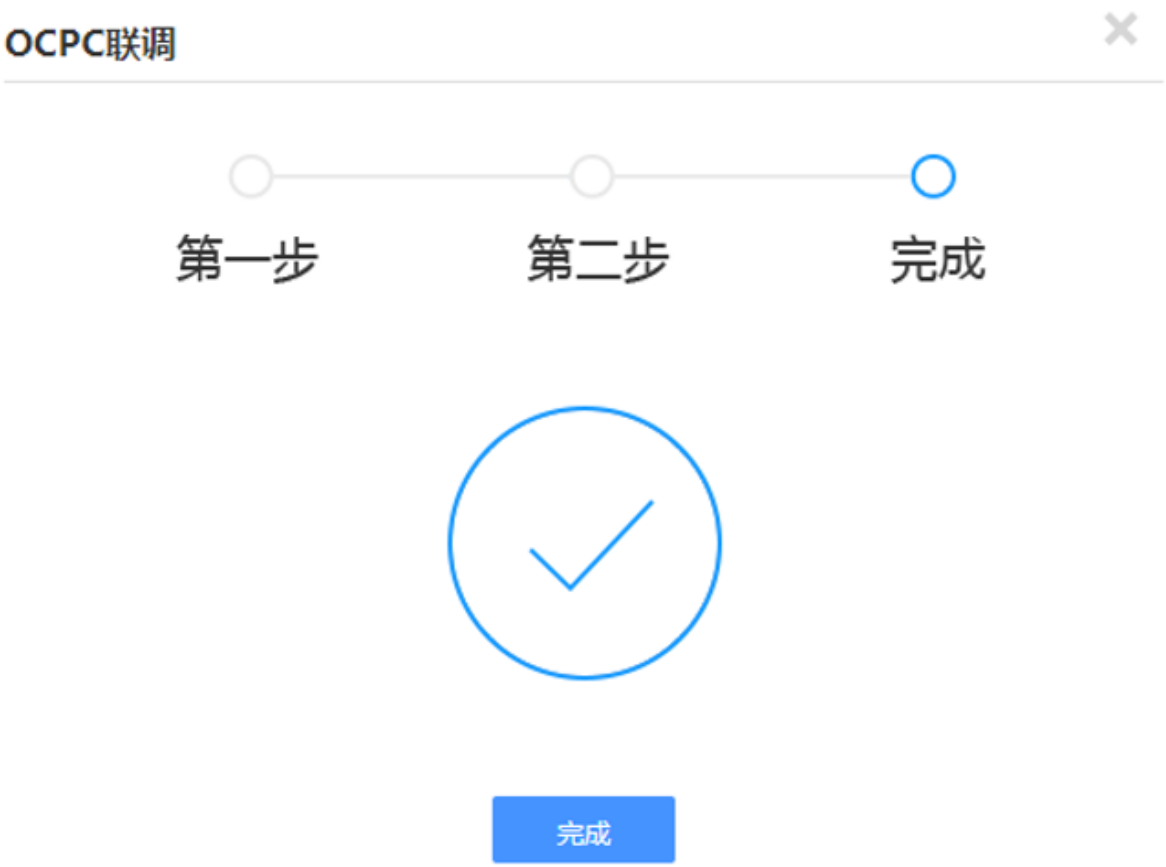
需将 akey 填写至 MTA 管理台


APP下载
点击查看API接口帮助文档
转化名称*：
test
1. 输入转化名称
App类型*：
iOS
Android
下载URL*：
https://itunes.apple.com/cn/app/id12345
2.输入应用下载地址
转化类型：
激活
转化方案：
API
监测地址*：
http://www.mamaiwo.com
3. 输入MTA提供的监测地址
akey：
4.将akey复制并配置在MTA的新建推广单元页面中
提交
返回

4. 进行测试


OCPC联调
第一步
第二步
完成
按照以下步骤进行测试
1、请使用最新版百度推广APP扫描二维码下载您的应用；
2、请在下载并激活您的应用后，点击下一步（确保您使用的手机是第一次激活您的应用）；
3、下载应用可能会消耗大量的流量，建议在wifi环境下进行联调。
下一步


OCPC联调
第一步
第二步
完成
正在联调，请稍候...


OCPC联调
第一步
第二步
完成
完成

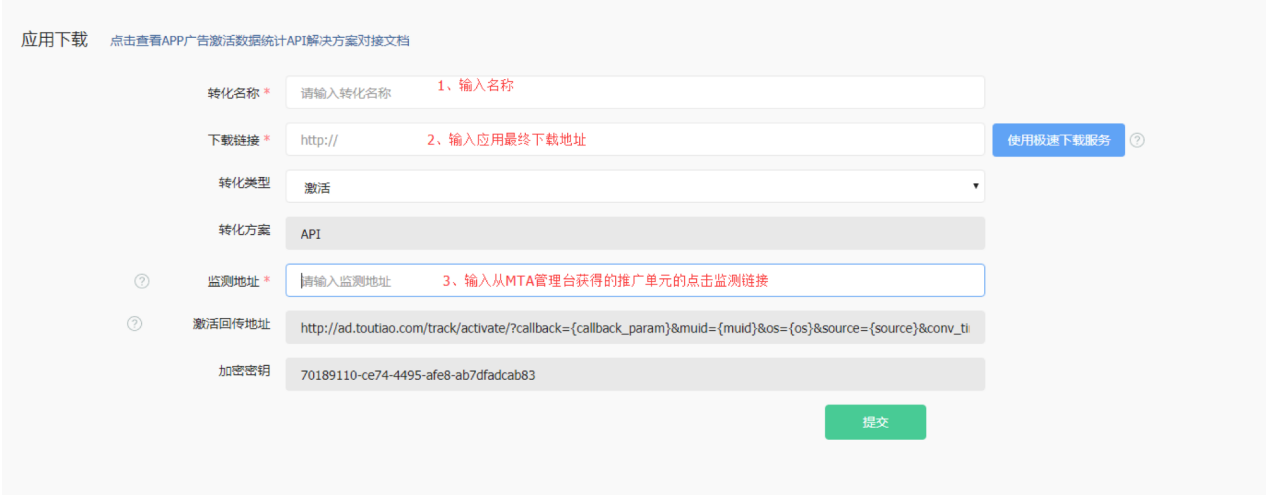
今日头条

今日头条推广配置

1. 进入今日头条管理台【工具箱】，找到【优化辅助】中的【转化跟踪】

2. 点击选择【应用下载 API】

3. 按步骤依次输入，其中监测地址填写 MTA 地址，输入完后点击【提交】按钮提交



4. 提交成功后，若该应用第一次在头条转化统计，点击【开始联调】，并按步骤进行联调直至联调成功

5. 在新建广告计划时，选择已经创建的转化名称，开始投放广告

6. 投放广告后，返回 MTA 广告效果追踪管理台查看统计数据

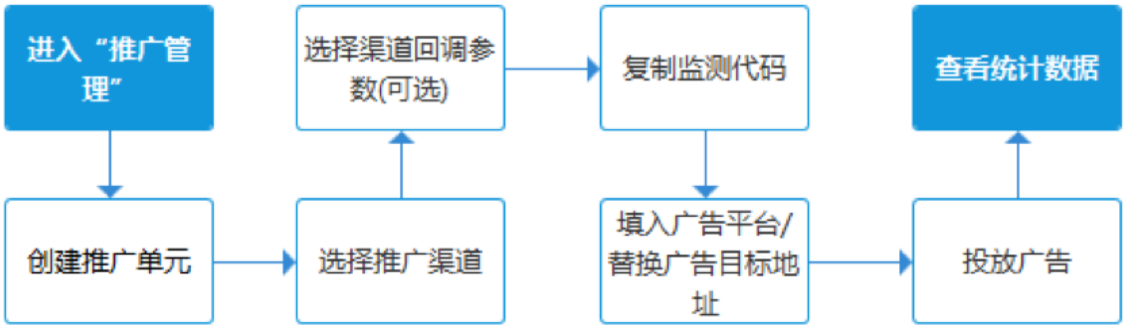
操作说明

功能介绍

MTA广告效果监测（即原安装来源分析），通过集成MTA SDK，在不同的广告中使用MTA的监测代码，可以帮助您：

1. 统计您的App来自不同推广渠道的曝光、点击、激活数据。
2. 按渠道或具体的广告，分析来自不同来源的用户在应用内行为，如注册、付费等。

快速使用流程：



集成SDK

参照 广告效果分析-SDK集成说明。

配置推广单元

MTA 通过推广单元来为您监测每一个广告的效果，通过配置推广单元，您可以获得对应的监测链接(监测代码)，将其配置到投放平台即可开始统计。

配置步骤：

1. 在【推广管理】,【推广单元管理】页面，点击【创建推广单元】

2. 填写广告被点击后的目标地址，可以是应用商店下载地址或者推广活动网页
3. 选择广告所属渠道，如果该广告不是在常见推广渠道投放，您也可以自定义渠道
4. 选择其他配置后，复制生成的链接地址。

投放广告

若您在配置推广单元时，选择的渠道是标准渠道，可将复制的链接地址粘贴在对应广告平台的第三方监测链接填写处，并投放广告；若您选择的自定义渠道，可将链接地址复制并替换广告跳转的地址，即可进行统计：

- 标准渠道 指与 MTA 完成 API 对接的渠道，对接后 MTA 将按渠道提供的设备号进行统计，此种方式统计最准确；若您投放的渠道属于 App 常见投放渠道，可与我们联系，由我们联系并完成对接；

- 自定义渠道 适用于自有推广页面或未完成API对接的渠道，该情况下，MTA 主要通过业内通用 UA + IP 方式进行设备模糊匹配，该种模式统计数据仍可供参考对比以及渠道质量对比分析。

若有任何疑问，可参考左侧“帮助文档”或联系 dtsupport@tencent.com。

查看效果

概览

您可以在此看到所有的推广数据，同时可以将汇总推广流量和自然流量进行流量质量的对比。

推广效果

您可以在管理台左侧菜单“推广效果”中查看推广的详细数据，MTA 提供非常丰富的指标对不同推广单元进行效果统计，您可以在表格右上方找到【自定义指标】的入口，选择不同的指标来查看；与此同时，您可以在表格上方切换不同统计维度来查看数据，如通过【推广渠道】维度查看，可对比不同渠道统计效果；

单元详情

在“推广效果”中，点击某一个推广单元可进入详情，查看该推广单元实时/每日的指标效果，还可以添加对比两个不同的推广单元，比较效果差异；

SDK集成说明

SDK集成说明

要使用 MTA 广告效果监测与分析，您需要先在 App中集成 MTA 的 SDK，集成步骤如下：

1. 集成 MTA 的基础 SDK （Android快速集成入口 | iOS快速集成入口），如果您已集成过 MTA 的基础 SDK ，可跳过该步骤；
2. 集成广告监测模块（Android集成代码 | iOS集成代码），并上报广告监测标准事件；
3. 若您的推广目标是 H5 页面（简称“落地页”），最终通过落地页下载应用统计，您可以在落地页嵌入 MTA 的 JS
   代
   码，
   可以做到
   一个落地页多渠道、多平台的统计，适用于微信内、分享、地推、二维码等各场景（Android集成配置 | iOS集成配置 | 落地页多渠道多平台配置）

模块集成指南

Android JS SDK配置

Android JS SDK配置

工程配置

1. 【必选】要在 MTA 前台开通并配置相关的推广计划；
2. 【必选】在 App 的入口处，一般为 Application 或 MainActivity 的 onCreate() 调用

   StatConfig.setTLinkStatus(true);

3. 【可选】若有接入腾讯 TBS 浏览服务 SDK ，请在主线程调用

   QbSdk.initX5Environment()

   方法后面添加

   StatConfig.invokeTBSSdkOnUiThread(context);

落地页 JS 配置

以下为嵌入落地页的 js sdk 格式，必须填写替换参数：

参数名	参数描述
$download_btn_id	下载按钮的id
$app_key	MTA 管理台中 Android 的 App KEY

```
<script>
var _mta_btn_id = '$download_btn_id';
 (function() {
  var mta = document.createElement("script");
  mta.src = "//pingjs.qq.com/mta/channel_stats.js?v1";
```

```
  mta.setAttribute("name", "MTA_CHANNEL");
  mta.setAttribute("app_key", "$app_key");
  var s = document.getElementsByTagName("script")[0];
  s.parentNode.insertBefore(mta, s);
})();
</script>
```

使用示例

```
<a id="download_app">下载应用</a>

<script>
var _mta_btn_id = 'download_app';
(function() {
  var mta = document.createElement("script");
  mta.src = "//pingjs.qq.com/mta/channel_stats.js?v1";
  mta.setAttribute("name", "MTA_CHANNEL");
  mta.setAttribute("app_key", "IB7ZRJ6V8S1T");
  var s = document.getElementsByTagName("script")[0];
  s.parentNode.insertBefore(mta, s);
})();
</script>
```

Android SDK广告监测模块集成指南

Android SDK广告监测模块集成指南

接口调用

注册事件

通过上报付费事件，您可以统计到每一次投放的注册转换率等标准监测指标，也能向渠道回传以获得广告投放优化。

接口说明：

tatService.trackRegAccountEvent(Context context,String user ，AccountType type);

参数说明：

参数名	是否必要	参数描述
context	是	当前上下文
user	是	用户名，最多64个字符
type	是	目前支持手机号(" mobile ")、邮箱(" mail ")、微信ID(" WX")、QQ号("QQ"). 如：StatConfig.AccountType.WX

付费事件

通过上报付费事件，您可以统计到每一次投放的LTV值从而衡量您的投资回报率。

接口说明：

StatService.trackPayEvent(Context context, String cy, String id, double money, CurrencyType type);

参数说明：

参数名	是否必要	参数描述
context	是	当前上下文
cy	是	支付方式（最多64个字符）.如（"wx"、"Alipay"）等
id	是	订单号等（最多64个字符）.如（"888888999955542"）等
money	是	金额.如（88.8）
type	是	目前支持两种：CNY人民币("CNY")、USD美金("USD")。如StatConfig.CurrencyType.CNY

其他自定义事件

若您希望上报其他自定义的用户行为，您可以参考MTA自定义事件： 自定义事件介绍 | Android接口 | iOS接口

iOS JS SDK配置

iOS JS SDK配置

工程配置

1. 统计安装来源(URL Scheme)
XCode 中的 URL Types 中增加一条 URL Scheme 配置，Role 是 Viewer 。URL Schemes 的配置后续在 JS SDK 的初始化会用到。
2. 如果已安装 App ，直接打开(非必需，通过 Universal Links 技术)
i. 首先你需要有一个https的域名，例如domain.com。
ii. Uninversal Links 需要的 json 文件的【 apple-app-site-association 】，可以从MTA管理台生成。
iii.
把apple-app-site-association上传到domain.com根目录(iOS系统会自动从https://domain.com/apple-app-site-association进行访问)
iiii. XCode的capabilities增加Domains的配置，例如applinks:domain.com
3.下载页面的修改
请参考MTA管理台中关于JS SDK的说明。

> 注意：
> 因为用到了keychain，如果遇到相关编译不过的问题，请在项目中引用Security.framework。

接口调用

AppDelegate中的改动

1. 在MTA的初始化之后增加[ADTracker getInstance]

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
// Override point for customization after application launch.
```

```
[[MTAConfig getInstance] setSmartReporting:YES];
[[MTAConfig getInstance] setReportStrategy:MTA_STRATEGY_INSTANT];
[[MTAConfig getInstance] setDebugEnable:YES];
[MTA startWithAppkey:@"I2E3KXDU1E2W"];
[ADTracker getInstance];
return YES;
}
```

2. 在 handleOpenURL 中增加调用

```
- (BOOL)application:(UIApplication *)application handleOpenURL:(NSURL *)url{
[[ADTracker getInstance] handleOpenURL:url];
return true;
}
```

3. 在 App 进入的第一个 ViewController 的修改
   viewDidLoad中添加以下代码

```
(void)viewDidLoad {
[super viewDidLoad];
[[ADTracker getInstance] startByViewDidload];
}
```

4. 如果有自己的中间页，不使用 MTA 管理台生成的话。需要单独接入JS SDK，并设置中间页的地址.
   didFinishLaunchingWithOptions 的初始化修改如下

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary
*)launchOptions {
  // Override point for customization after application launch.
  [[MTAConfig getInstance] setSmartReporting:YES];
```

```
  [[MTAConfig getInstance] setReportStrategy:MTA_STRATEGY_INSTANT];
  [[MTAConfig getInstance] setDebugEnable:YES];
  [MTA startWithAppkey:@"I2E3KXDU1E2W"];
  [[ADTracker getInstance] setChannelUrl:@"http://domain.com/test/download.html"];
  return YES;
}
```

注意：

1. http://domain.com/test/download.html 这代表着你投放的网址可能是 http://domain.com/test/download.html?ADTAG=youradtag ; http://domain.com/test/download.html?ADTAG=youradtag2; http://domain.com/test/download.html?ADTAG=youradtag3 等等。
2. 请替换 http://domain.com/test/download.html 为实际的中间页地址。
3. JS SDK的使用参考MTA管理台相关页面。

落地页JS配置

以下为嵌入落地页的 js sdk 格式，必须填写替换参数：

参数名	参数描述
$download_btn_id	下载按钮的 id
$app_key	MTA 管理台中 iOS 的 App KEY
$URL_Scheme	iOS App 的 URL Scheme

```
<script>
var _mta_btn_id = '$download_btn_id';
 (function() {
  var mta = document.createElement("script");
  mta.src = "//pingjs.qq.com/mta/channel_stats.js?v1";
```

```
  mta.setAttribute("name", "MTA_CHANNEL");
  mta.setAttribute("app_key", "$app_key");
  mta.setAttribute("app_flag", "$URL_Scheme");
  var s = document.getElementsByTagName("script")[0];
  s.parentNode.insertBefore(mta, s);
})();
</script>
```

使用示例

```
<a id="download_app">下载应用</a>
<script>
var _mta_btn_id = 'download_app';
(function() {
  var mta = document.createElement("script");
  mta.src = "//pingjs.qq.com/mta/channel_stats.js?v1";
  mta.setAttribute("name", "MTA_CHANNEL");
  mta.setAttribute("app_key", "IB7ZRJ6V8S1T");
  mta.setAttribute("app_flag", "mtaApp");
  var s = document.getElementsByTagName("script")[0];
  s.parentNode.insertBefore(mta, s);
})();
</script>
```

iOS SDK广告监测模块集成指南

iOS SDK广告监测模块集成指南

工程配置

引入 adtracker 文件夹下的 .a 文件和头文件,以及 idfa 文件夹下的 libidfa.a 到 Xcode 工程之中。

接口调用

> 注意：
> 激活后的活跃事件的上报, 请在主线程中调用。

注册事件

通过上报付费事件，您可以统计到每一次投放的注册转换率等标准监测指标，也能向渠道回传以获得广告投放优化

接口说明

(void)trackRegAccountEvent:(MTAADAccountType)accountType account:(NSString *)account

参数说明

参数名	是否必要	参数描述
accountType	是	账号类型
account	是	具体的账号

付费事件

通过上报付费事件，您可以统计到每一次投放的注册转换率等标准监测指标，也能向渠道回传以获得广告投放优化。

接口说明

(void)trackUserPayEvent:(MTAADPayMoneyType)moneyType orderID:(NSString *)orderID
payNum:(double)payNum
payType:(NSString *)payType;

参数说明

参数名	是否必要	参数描述
moneyType	是	货币种类，支持两种：人民币、美金
orderID	是	订单ID， 或交易流水号
payNum	是	订单金额
payType	是	支付类型，如微信、支付宝、银联等

其他自定义事件

若您希望上报其他自定义的用户行为，您可以参考MTA自定义事件： 自定义事件介绍 | Android接口 | iOS接口

落地页多渠道多平台配置

落地页多渠道多平台配置

若您需要在微信、朋友圈、二维码、短信等场景下进行推广，且推广地址为活动页面，可以使用以下方案：
在落地页（活动页面）嵌入 JS SDK 时，可以在代码中根据当前访问页面的设备系统(参考 HTTP Useragent)将$adtag设置成您在 MTA 获得的 TLink 中的后缀 ADTAG值，同时将 $app_key 也设置成在 MTA 中获得的不同平台的 Appkey 即可。

```
<script>
var _mta_btn_id = '$download_btn_id';
 var _mta_adtag = '$adtag';
 (function() {
  var mta = document.createElement("script");
  mta.src = "//pingjs.qq.com/mta/channel_stats.js?v1";
  mta.setAttribute("name", "MTA_CHANNEL");
  mta.setAttribute("app_key", "$app_key");
  mta.setAttribute("app_flag", "$URL_Scheme"); (Android不需要该行)
  var s = document.getElementsByTagName("script")[0];
  s.parentNode.insertBefore(mta, s);
  })();
</script>
```

使用示例

在MTA管理台Android和iOS中各配置两个推广链接： iOS: http://xxxxx.com?ADTAG=111
Android: http://xxxx.com?ADTAG=222
其中iOS的 Appkey 为 AAAAAAAA，Android 的 Appkey 为 BBBBBBB（见管理台 -> 配置管理 -> 应用信息页面） 则在活动页面中判断，若当前设备为 iOS，将 $adtag 设置成 111，同时 $app_key 设置为 AAAAAAAA；若当前设备为Android, 将 $adtag 设置成 222，同时 $app_key 设置为 BBBBBBB。

> 注意:
> 如果需要多个链接跳转同一个落地页，同时每个链接都需要统计Android和iOS，则可以在跳转至落地页前将ADTAG附在URL的参数中