广点通移动 APP 激活数据统计

API 上报方案对接文档（API 方案二）

2015-01

目录

激活数是 APP 广告主衡量转化效果的重要指标。为了更加方便、准确的统计到广告主所推广的移动 APP 的激活数据，广点通为广告主提供了移动 APP 激活统计的方案：

对 Android 应用，激活统计方案包括：SDK 方案、API 统计方案和 API 上报方案。

对 ios 应用，激活统计方案包括：API 统计方案和 API 上报方案。

*说明：此文档为移动 APP 激活统计：**API 上报方案（即：API 方案二）**的对接说明，**适用于 Android 应用和 ios 应用**。*

1. 方案说明

API 统计方案（即：API 方案二）：广告主上报激活数据，广点通搭建服务系统关联点击数据和广告主提供的所有激活数据。

- 当广告主收到 APP 激活请求时，广告主系统记录激活数据，并将所有的激活数据提交给广点通系统；
- 广点通搭建服务系统对比激活数据和点击数据，如果有 7 天内广点通带来的激活，则记为有效激活，最后计算激活量，并在 gdt 投放端展示激活数。

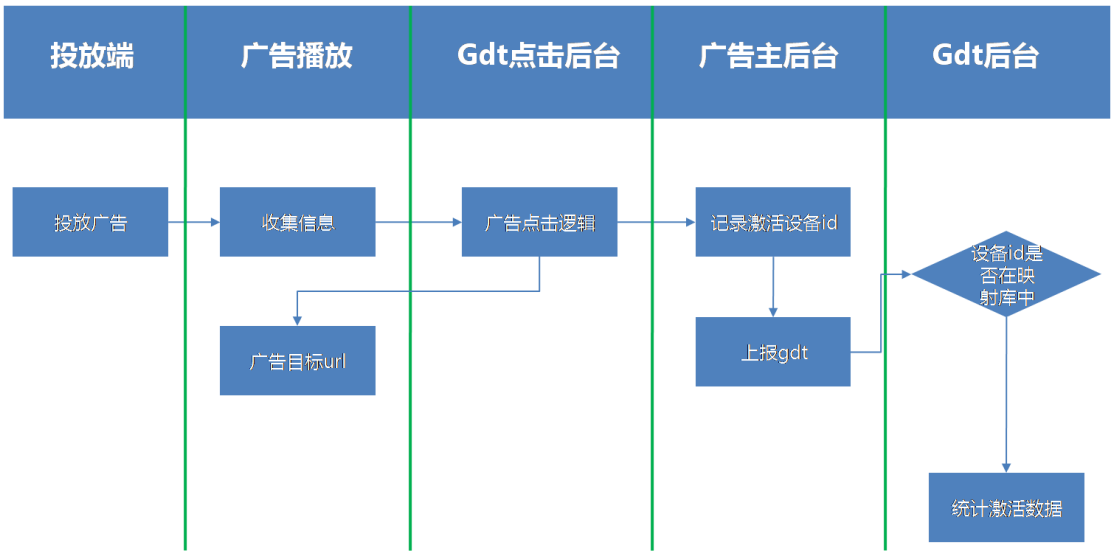

图 1：方案 2 数据流示意图

2. 接口说明

实现该接口，广告主可以将激活上报给广点通，广点通搭建对比服务系统，根据广告主上报的激活设备 id 和激活时间，对比广点通点击设备 id 和点击时间，从而统计到激活数据，并与广告关联，跟踪广告的转化效果。

上报方式：广告主必须在收到 APP 激活后**实时**将效果上报给广点通，以免出现统计无效的情况。

简介

请求格式：

http://t.gdt.qq.com/conv/app/{appid}/conv?v={data}&conv_type={conv_type}&app_type={app_type}&advertiser_id={uid}

参数说明：

appid：数值，android 应用为开放平台移动应用的 id，或者 ios 应用在 Apple App Store 的 id；广告主在广点通（e.qq.com）创建转化之后，系统会自动生成该 id；

data：为加密的数据结构，字符串，详细描述见本文第 3 部分；

conv_type：为转化类型，枚举值，现在只有移动应用激活类型**(MOBILEAPP_ACTIVITE)**；

app_type：为应用类型，枚举值，现阶段只有 ANDROID 和 IOS；**注意要大写；**

uid：数值，广告主在广点通（e.qq.com）的账户 id；广告主在广点通（e.qq.com）创建转化之后，系统会自动生成该 id；

请求方式：

HTTP GET 方法

相应格式：

Json 数据格式

相应内容：

{"ret": 返回码, "msg": "错误提示"}

返回码为 0 标识正常接收，其他返回码标识错误。

返回码	说明
0	成功

-1	请求非法参数
-2	参数解析失败
-3	参数解码失败
-12	获取密钥失败
-13	非法的应用类型
-14	非法的转化时间
-15	非法的广点通移动设备标识
-17	获取转化规则失败

3. V 参数加密方案

整个加密方案分为了四个部分：**组装参数、参数签名、参数加密、组装请求。**

而对于每一个 APPID 我们会分配一个加密密钥 encrypt_key 和一个签名密钥 sign_key：

加密密钥 encrypt_key 和签名密钥 sign_key 获取方式：广告主在广点通（e.qq.com）创建转化之后，系统会自动生成密钥。

加密密钥 encrypt_key 和签名密钥 sign_key 粒度：广告主+APPid+转化规则（默认：激活），即：对每一个广告主账户、每一个 app、每一种转化行为，都会生成一组密钥。

（1）组合参数

首先需要需要获取下列取值：

- click_id // 广点通点击跟踪 ID（必选）

广点通系统中标识用户每次点击生成的唯一标识；

- muid // 广点通设备标识 ID（必选）

*用户设备的 IMEI 或 idfa 进行 MD5SUM 以后得到的 32 位全小写 MD5 表现字符串；**加密方案详见本文第 4 部分***

- conv_time // 转化发生时间（必选）

*激活发生的时间的标准时间戳，**秒级别，不能是毫秒**；由于激活效果统计有时间期限限制，目前系统要求：激活与点击时间间隔最长为 7 天*

- client_ip // 转化发生 IP（可选） *激活发生用户的客户端 IP，选填；*

正确取值以后，我们将上述的内容按照下列方式拼接成一个合法的 query_string：

{key1}=urlencode({value1})&{key2}=urlencode({value2})

注：

1. 此处如果不填写 client_iP，可以直接在 query_string 中去除该参数；

2. 此处组合参数无顺序要求。

例：

muid: 0f074dc8e1f0547310e729032ac0730b
conv_time: 1422263664
client_ip: 10.11.12.13

变为

muid=0f074dc8e1f0547310e729032ac0730b&conv_time=1422263664&client_ip=10.11.12.13

（2）参数签名

按照上述组合参数的规则组成了一个完整的 query_string 串以后，此时我们可以得到一个基础的请求结构，我们先按照如下结构组成一个新的字符串 page：

http://t.gdt.qq.com/conv/app/{appid}/conv?{query_string}

例：

appid: 112233
query_string:
muid=0f074dc8e1f0547310e729032ac0730b&conv_time=1422263664&client_ip=10.11.12.13

变为

http://t.gdt.qq.com/conv/app/112233/conv?muid=0f074dc8e1f0547310e729032ac0730b&conv_time=1422263664&client_ip=10.11.12.13

通过得到的 page 字符串，我们进行整体 urlencode 得到 encode_page，然后按照下述描述组装成新的字符串 property。

{sign_key}&GET&{encode_page}

例：

sign_key: test_sign_key

变为：

test_sign_key&GET&http%3A%2F%2Ft.gdt.qq.com%2Fconv%2Fapp%2F112233%2Fconv%3Fmuid%3D0f074dc8e1f0547310e729032ac0730b%26conv_time%3D1422263664%26client_ip%3D10.11.12.13

我们对 property 进行 md5sum 获得 32 位的小写加密串即为 signature。例：

property:
test_sign_key&GET&http%3A%2F%2Ft.gdt.qq.com%2Fconv%2Fapp%2F112233%2Fconv%3Fmuid%3D0f074dc8e1f0547310e729032ac0730b%26conv_time%3D1422263664%26client_ip%3D10.11.12.13

变为

8a4d7f5323fd91b37430d639e6f7371b

（3）参数加密

按照上述组合参数的规则组成的 query_string 以及参数签名得到的 signature 组合，通过下述方式得到新的 base_data。

{query_string}&sign=urlencode({signature})

例：

query_string:
muid=0f074dc8e1f0547310e729032ac0730b&conv_time=1422263664&client_ip=10.11.12.13
signature: 8a4d7f5323fd91b37430d639e6f7371b

变为

muid=0f074dc8e1f0547310e729032ac0730b&conv_time=1422263664&client_ip=10.11.12.13&sign=8a4d7f5323fd91b37430d639e6f7371b

通过上一个步骤得到的 base_data，我们与 encrypt_key 进行简单异或，最终得到 base64 表达形式就是 data。此时按照接口约定中的请求格式向广点通这边发送请求。

base64(simple_xor({base_data}, {encrypt_key}))

简单异或加密代码示例（C++）：

```
bool SimpleXor(
        const std::string& info,
        const std::string& key,
        std::string* result) {
    if (result == NULL || info.empty() || key.empty()) {
```

```
            return false;
        }
        result->clear();

        uint32_t i = 0;
        uint32_t j = 0;
        for (; i < info.size(); ++i) {
            result->push_back(static_cast<unsigned char>(info[i] ^ key[j]));
            j = (++j) % (key.length());
        }

        return true;
    }
```

简单异或加密代码示例（Python）：

```
这边有一个python的示例代码可以看下
def SimpleXor(source, key):
 retval = ''
 j = 0
 for ch in source:
  retval = retval + chr(ord(ch)^ord(key[j]))
  j = j + 1
  j = j % (len(key))
 return retval
```

例：

base_data:
muid=0f074dc8e1f0547310e729032ac0730b&conv_time=1422263664&client_ip=10.11.12.13&sign=8a4d7f5323fd91b37430d639e6f7371b
encrypt_key: test_encrypt_key

变为

GRAaEGJVCFNFTRQXZw5UH0RQR0NsVF4GRUtJRGxZBBpEUkBEPUMNDBwPLwA2BgBERVFBRm1TXVVETVYXMwIAFwA6GgRiVF5NQ0heRW1FVEpSFhoTMVhWAkYdRxJqWFdKEgFKRT1WWVdBSRRCbFIATxJSQENuBw==

（4）组装请求：获取以下参数取值。

conv_type　// 转化类型（必选）

取值为 MOBILEAPP_ACTIVITE；

app_type　// 转化应用类型（必选）

激活发生的 APP 类型，**ios 应用取值为 IOS，Android 应用取值为 ANDROID**

advertiser_id　// 广告主 ID（必选）

广告主在广点通广告系统中的广告主标识 ID

正确取值以后，我们将上述的内容按照下列方式拼接成一个合法的 attachment。

{key1}=urlencode({value1})&{key2}=urlencode({value2})

例：

conv_type: MOBILEAPP_ACTIVITE
app_type: ANDROID
advertiser_id: 10000

变为

conv_type=MOBILEAPP_ACTIVITE&app_type=ANDROID&advertiser_id=10000

获取到正确的 attachment 以后，我们便可以按照下列形式，拼接出最后的请求。

http://t.gdt.qq.com/conv/app/{appid}/conv?v={data}&{attachment}

最终请求的形式为：

http://t.gdt.qq.com/conv/app/112233/conv?v=GRAaEGJVCFNFTRQXZw5UH0RQR0NsVF4GRUtJRGxZBBpEUkBEPUMNDBwPLwA2BgBERVFBRm1TXVVETVYXMwIAFwA6GgRiVF5NQ0heRW1FVEpSFhoTMVhWAkYdRxJqWFdKEgFKRT1WWVdBSRRCbFIATxJSQENuBw%3D%3D&conv_type=MOBILEAPP_ACTIVITE&app_type=ANDROID&advertiser_id=10000

4. muid 加密方案

muid：设备 id，由 IMEI（Android 应用）MD5 生成，或是由 IDFA（iOS 应用）MD5 生成；

具体加密方案如下：

Android 设备-muid 加密规则：IMEI 号(需转小写)，进行 MD5SUM 以后得到的 32 位全小写 MD5 表现字符串。

IOS 设备-muid 加密规则：IDFA 码（需转大写），进行 MD5SUM 以后得到的 32 位全小写 MD5 表现字符串。

Muid 加密代码示例（C++）：

```
bool GenerateMuid(
        const std::string& uni_id, const int32_t app_type, std::string* muid) {
    if (muid == NULL || uni_id.empty()) {
        return false;
    }
    muid->clear();
    std::string tmp;
    if (app_type == kAppTypeAndoirdImei) {
        LatinToLower(uni_id, &tmp);
    } else if (app_type == kAppTypeiOSIfa) {
        LatinToUpper(uni_id, &tmp);
    } else {
        return false;
    }
    md5sum32l(tmp, muid);    // 32bit lower
    return true;
}
```

测试用例：

Android 设备号加密测试用例：

原始 IMEI 号：354649050046412

加密之后：b496ec1169770ea274a2b4f42ca4fb71

IOS 设备号加密测试用例：

原始 IDFA 码：1E2DFA89-496A-47FD-9941-DF1FC4E6484A

加密之后：40c7084b4845eebce9d07b8a18a055fc