Tracking API说明文档

# 一 Kingdom, Phylum, Classfield, Family, Genus五级分类的说明

Kingdom, phylum, classfield, family, genus的基本解释可以理解为界、门、纲、科、属，这五级分类的界定范围大小一次Kingdom>phylum>classfield>family>genus（详见图1），对于每一个Metrics里边的类，这五个等级逐级递减的细化了用户某个行为或者某个相关操作的分类。例如，记录一个用户的tutorial\_counter的Kingdom, phylum, classfield, family, genus可以分别为”update\_building”, ”click\_ok”, “”, “”, “”, 这五级分类可以不全部充满，如果两级或者三级可以说明用户操作的说明的话，其他的次级分类可以直接赋值为空字符串。在这五级分类之外扩展了一层extra，以应对当五级分类并不能细化问题之时的扩展层分类。

图1 五级分类的边界大小关系

五级分类举例：

战国的economy的五级分类分析举例：

战国的经济系统是由coin来进行经济活动的。对战国的经济系统的五级分类就要从coin的用途入手来一步一步细分用户的经济活动。

显而易见，coin的最原始的用途只有两类：获得和花费。这样，根据五级分类的要求，kingdom的分类就被确定下来，即，kingdom = ‘earning’ 和 kingdom=’expenditure’。

以kingdom=’earning’为例继续划分其余四级分类：

在游戏中，earning的渠道会有领取俸禄、交叉推广活动赚取、pve战斗获得等，因此，kingdom的次级分类中phylum就分别会等于’retrive\_salary’, ’cross\_award’, ’pve\_award’。

‘retrive\_salary’和’pve\_award’到phylum这一层级已经不能再划分，所以对于这两种情况，一下三级的分类均要被赋值为空字符串。对于’cross\_award’，由于交叉推广还会有不同的获得方式，因此，’cross\_award’还会继续划分下一层级classfield。

交叉推广在战国初期的设计中，用户可以通过邀请好友、日常任务、提高等级获得金钱，因此，在’cross\_award’下的classfield会由’invite\_friends’, ‘daily\_mission’, ‘level\_up’构成。

‘daily\_mission’已经不能再分类到更细，所以，daily\_mission的下两个层级会为空串。’invite\_friends’在游戏中，用户邀请不同数量的好友会获得不同数量的coin，2个、10个、36个都会有奖励，所以’invite\_friends’下的family分别是’reach\_two’, ‘reach\_ten’, ‘reach\_thirtysix’。’level\_up’在游戏中，用户达到6、7、8级也会被奖励不同的coin，因此，’level\_up’下的family分别是’level\_six\_award’, ‘level\_seven\_award’, ‘level\_eight\_award’。

到此，earning的经济活动方式已经被准确细致的记录下来，可以看到，五级分类中，有些层级是没有用到的。

# 二 Metrics的类说明

##### **1 Counter**

Counter通俗来说是对用户的操作的记数操作。 Counter是针对游戏中用户实际操作的记录，可以说Counter对于游戏来说是万能的记录工具。通过针对游戏中各个用户操作的Counter打点记录，BI可以很轻易的得到用户在此打点步骤中所有的人数及用户行为，这样通过这些人数的增减以及和其他类的合并分析，BI可以很轻易的找到用户行为的方向。

属性说明：

**private** String user\_level;

用户被记录时的等级。

**private** String counter;

用户被记录counter的内容，可以是”tutorialCounter”, “startupCounter”等等，这些counter的定义是根据需求而变的。

**private** String value = "";

用户被记录时的相关参数值。

**private** String extra = "";

为了应对五级分类带来的记录的局限性（有些用户行为五级分类并不能详细说明），BI新加入了extra来扩展五级分类。

**private** String kingdom = "";

**private** String phylum = "";

**private** String classfield = "";

**private** String family = "";

**private** String genus = "";

以上五类详见Kingdom, Phylum, Classfield, Family, Genus的说明

**private** String counter\_date;

Counter被记录的日期，例如”2012-03-23”

**private** String counter\_time;

Counter被记录的时间，例如”03:04:54”

例：战国的startup\_counter举例

|  |  |  |
| --- | --- | --- |
| **Counter** | **Kingdom** | **Phylum** |
| startup | [first,all] | load\_init |
| startup | [first,all] | load\_10 |
| startup | [first,all] | load\_20 |
| startup | [first,all] | load\_30 |
| startup | [first,all] | load\_40 |
| startup | [first,all] | load\_50 |
| startup | [first,all] | load\_60 |
| startup | [first,all] | load\_70 |
| startup | [first,all] | load\_80 |
| startup | [first,all] | load\_90 |
| startup | [first,all] | load\_completed |

Counter字段被定义为startup，这样所有相关startup的计数器都被称为startup这个名称。Kingdom被定义为两类，first（新手），all（老用户），phylum在first或all的基础上可以分为load\_init, 10, 20…….直到load\_complete，这样通过对startup\_counter的各个字段的定义，就可以清晰的统计出开始游戏是新用户还是老用户，有多少用户在load的哪一个阶段流失掉了，或者可以分析出有多少用户通过了loading这个阶段。通过分析这样清晰的流失率数据，能够更好地帮助游戏改善counter所统计的相关阶段的游戏体验。

##### **2 DAU**

DAU是记录用户每天登陆情况的类，接入DAU时要注意要按天去除重复，还有就是当用户被记录Install之后也要记录DAU，因为Install的用户在当天也是DAU用户。

属性说明：

**private** String source = "";

这个字段是BI处理的字段，传空字符串即可。

**private** String affiliate = "";

这个字段是BI处理的字段，传空字符串即可。

**private** String creative = "";

creative记录了用户的来源的渠道，这些渠道是游戏平台提供的，例如用户是通过搜索进入游戏，或者是通过好友介绍进入游戏等。腾讯平台是通过via这个参数传递的。

**private** String family = "";

在DAU没有做分类的情况下可以不传。

**private** String genus = "";

在DAU没有做分类的情况下可以不传。

注：如果是有必要对用户进行简单的分类，可以按照“Kingdom, Phylum, Classfield, Family, Genus五级分类的说明”进行划分。

**private** String ip = "";

用户登陆所用的ip。

**private** String dau\_date;

**private** String dau\_time;

**private** String from\_uid = "";

用户登陆进来如果是通过好友进入到游戏的，则通过from\_uid记录这个用户好友的platformId。

**private** String extra = "";

五级分类的扩展分类

##### 3 Demographic

Demographic记录了游戏用户的真实信息，这些信息是从游戏所处的平台获取到的。当用户更改了自己的用户信息，这些修改后的信息也应当被记录到demographic之中去。当获取不到完整的信息是，Demographic之中的属性可以为空字符串

属性说明：

**private** String age = "";

用户的年龄。

**private** String first = "";

用户的名。

**private** String last = "";

用户的姓。

**private** String name = "";

用户的姓名。

**private** String gender = "";

用户的性别。

**private** String email = "";

用户的电子邮件地址。

**private** String city = "";

用户所在的城市、村或者县。

**private** String state = "";

用户所在的省或者州。

**private** String country = "";

用户所在国家。

**private** String locale = "";

用户所处地区的语言。

**private** String timezone = "";

用户所处的时区。

**private** String ip = "";

用户的ip。

**private** String picurl = "";

用户的头像。

**private** String is\_vip = "";

用户是否是vip。

**private** String vip\_type = "";

用户vip的类型。

**private** String extra = "";

用户的扩展信息，记录前边未提到的用户相关信息。

##### 4 Economy

Economy记录了游戏中用户所有涉及虚拟货币的行为。Economy可以为数据分析提供很好的数据支持，用于分析用户的经济行为和游戏的经济系统是否合理。

属性说明：

**private** String currency = "";

游戏中的币种，它可以是coin(gold)，honor，或者是qpoint等虚拟的可以获取或花费的货币或类货币。

**private** String amount = "";

amount是用户花费的总数，例如用户花费500gold购买了2个货物A，则amount应为500。

**private** String value = "";

value是用户购买物品的数量。如上例，value为2。

**private** String kingdom = "";

**private** String phylum = "";

**private** String classfield = "";

**private** String family = "";

**private** String genus = "";

五级分类。

**private** String economy\_date;

**private** String economy\_time;

**private** String extra = "";

五级分类的扩展分类。

##### 5 Install

Install是用来记录新用户的类，当系统判断当前用户为新用户的时候，应记录此用户的install， 同时记录此用户的dau，因为新用户也同样是活跃用户。

属性说明：

**private** String source = "";

这个字段是BI处理的字段，传空字符串即可。

**private** String affiliate = "";

这个字段是BI处理的字段，传空字符串即可。

**private** String creative = "";

creative记录了用户的来源的渠道，这些渠道是游戏平台提供的，例如用户是通过搜索进入游戏，或者是通过好友介绍进入游戏等。腾讯平台是通过via这个参数传递的。

**private** String family = "";

在install没有做分类的情况下可以不传。

**private** String genus = "";

在install没有做分类的情况下可以不传。

注：如果是有必要对用户进行简单的分类，可以按照“Kingdom, Phylum, Classfield, Family, Genus五级分类的说明”进行划分。

**private** String from\_uid = "";

用户登陆进来如果是通过好友进入到游戏的，则通过from\_uid记录这个用户好友的platformId。

**private** String install\_date;

**private** String install\_time;

**private** String extra = "";

五级分类的扩展分类。

##### 6 Payment

Payment不同于economy，payment是记录游戏中用户花费真实货币的购买行为的类。

属性说明：

**private** String amount;

amount是用户花费真实货币的数量。例如用户花费20RMB购买10个q币，则amount为20。

**private** String currency;

currency是记录用户花费的币种，人民币为RMB，美元为USD等。

**private** String provider = "";

provider是记录此次用户购买行为所通过的交易平台。

**private** String ip = "";

用户购买时所用的ip。

**private** String transactionid;

用户购买时所获取到的交易id号。

**private** String status;

交易的状态，成功（success）或不成功（failure）。

**private** String kingdom = "";

**private** String phylum = "";

层级分类，详见第一章。

**private** String value2 = "";

记录用户购买产品的数量。

**private** String payment\_date;

**private** String payment\_time;

**private** String extra = "";

扩展分类。

##### 7 Milestone

Milestone记录了用户完成一些重要任务或者达到某些等级的信息。Milestone可以记录用户达到新的等级时的信息。

属性说明：

**private** String milestone;

里程碑的信息。

**private** String value;

里程碑相对应的值。

**private** String milestone\_date;

**private** String milestone\_time;

**private** String extra = "";

扩展分级。

例：当用户达到新的一个等级时，milestone的信息应记录为：

milestone = “new\_level”;

value = “6”; 假定用户达到6级.

milestone\_date = “2012-03-27”;

milestone\_time = “18:47:00”;

extra = “”;

##### 8 Message

**private** String sendkey = "";

**private** String channel = "";

**private** String to\_uid = "";

**private** String status = "ok";

**private** String kingdom = "";

**private** String phylum = "";

**private** String classfield = "";

**private** String family = "";

**private** String genus = "";

**private** String message\_date;

**private** String message\_time;

**private** String extra = "";

例：

##### 9 MessageClick

**private** String sendkey = "";

**private** String channel = "";

**private** String from\_uid = "";

**private** String status = "ok";

**private** String kingdom = "";

**private** String phylum = "";

**private** String classfield = "";

**private** String family = "";

**private** String genus = "";

**private** String send\_date = "";

**private** String send\_time = "";

**private** String click\_date;

**private** String click\_time;

**private** String extra = "";

##### 10 Friends

**private** String friend\_uid = "";

**private** String kingdom = "";

**private** String phylum = "";

**private** String classfield = "";

**private** String friend\_date = "";

**private** String friend\_time = "";

**private** String extra = "";

**private** **int** deleted = 0;

# 三 接入的方法调用

##### 1 初始化BI信息

当后端启动服务时，应同时初始化BI的相关的参数信息。这些参数信息可以写到配置文件中，在applicationContext启动时自动调用相关方法初始化，发送到服务器当中去。

参数信息：

sn\_id：平台id。

client\_id：客户端id。

game\_id：游戏id。

scribed\_host：scribed\_host指向的地址，测试为192.168.40.104，正式环境为localhost。

core\_pool\_size：保留的线程池的大小，固定为2。

max\_pool\_size：最大的线程池的大小，固定为4。

queue\_capacity：队列的承载，固定为2147483647。

初始化方法：

TrackServices.init(String sn\_id, String client\_id, String game\_id, String scribed\_host, **int** core\_pool\_size, **int** max\_pool\_size, **int** queue\_capacity)

##### 2 发送数据

每次向BI传输数据时，程序后端应先初始化Metrics中相应的类的对象，然后调用TrackServices.add(TrackBase Bean)方法进行传输。

例：

（1）后端接收信息并构造counter对象

**public** Map<String, Object> trackingTutorialCounter(**int** userId,String platformId,

String kingdom, String phylum, String value,

String user\_level) {

**try** {

**long** pId = Long.*parseLong*(platformId, 16);

**这里构造counter对象**

Counter tutorialCounter = **TrackBaseFactory.*****createCounter***( 、

String.*valueOf*(pId), user\_level, "tutorial", kingdom,

phylum, "", "", "", value);

**传输数据调用的方法**

**TrackServices.*add*(tutorialCounter);**

} **catch** (Exception e) {

// ignore

}

**return** **new** HashMap<String, Object>();

}

createCounter的方法如下：

**public** **static** **final** Counter createCounter(String platformIdStr,String userLevel, String counter, String kingdom, String phylum, String classfield, String family, String genus, String value) {

Counter counterInstance = **new** Counter();

counterInstance.setUserId(platformIdStr);

counterInstance.setUser\_level(userLevel);

counterInstance.setCounter(counter);

counterInstance.setKingdom(kingdom);

counterInstance.setPhylum(phylum);

counterInstance.setClassfield(classfield);

counterInstance.setFamily(family);

counterInstance.setGenus(genus);

counterInstance.setValue(value);

Date now = **new** Date();

counterInstance.setCounter\_date(*getDateFormat*().format(now));

counterInstance.setCounter\_time(*getTimeFormat*().format(now));

**return** counterInstance;

}