淘米数据平台接入指南-4.1.1

淘米游戏-数据平台部

目录

[淘米数据平台接入指南-4.1.1 i](#_Toc510529445)

[淘米游戏-数据平台部 i](#_Toc510529446)

[一、综述 1](#_Toc510529447)

[1、 适用范围 1](#_Toc510529448)

[2、 统计标准 1](#_Toc510529449)

[3、 问题联系人 1](#_Toc510529450)

[二、接入流程 1](#_Toc510529451)

[1、 为游戏申请GameID 1](#_Toc510529452)

[2、 向工程中添加logger库文件 2](#_Toc510529453)

[3、 添加调用方法 2](#_Toc510529454)

[4、 进行数据测试 2](#_Toc510529455)

[三、添加调用方法(C++) 3](#_Toc510529456)

[调用前必读 3](#_Toc510529457)

[接口用途及调用方归属 3](#_Toc510529458)

[基础接口生成的日志，其中的通用字段说明 4](#_Toc510529459)

[0、 构造函数（StatLogger） 5](#_Toc510529460)

[1、 登录游戏验证密码（verify\_passwd） 5](#_Toc510529461)

[2、 创建游戏中角色（reg\_role） 6](#_Toc510529462)

[3、 登录游戏online服务器(login\_online) 8](#_Toc510529463)

[4、 退出游戏online服务器(logout) 10](#_Toc510529464)

[5、 统计当前在线人数(online\_count) 11](#_Toc510529465)

[6、 用户升级(level\_up) 11](#_Toc510529466)

[7、 付费（pay） 12](#_Toc510529467)

[8、 免费获得游戏币(obtain\_golds) 15](#_Toc510529468)

[9、 使用游戏币购买道具(buy\_item) 16](#_Toc510529469)

[10、 消耗游戏币（use\_golds） 17](#_Toc510529470)

[11、 接收任务（accept\_task） 18](#_Toc510529471)

[12、 完成任务（finish\_task） 19](#_Toc510529472)

[13、 放弃任务（abort\_task） 20](#_Toc510529473)

[14、 获得精灵（obtain\_spirit） 21](#_Toc510529474)

[15、 失去精灵（lose\_spirit） 21](#_Toc510529475)

[16、 退订VIP服务（unsubscribe） 22](#_Toc510529476)

[17、 销户VIP（cancel\_acct) 23](#_Toc510529477)

[18、 新用户注册转化（new\_trans） 24](#_Toc510529478)

[19、 自定义统计（log） 25](#_Toc510529479)

[20、 根据统计项ID落自定义统计数据（custom\_log） 32](#_Toc510529480)

[四、添加调用方法(PHP) 33](#_Toc510529481)

[0、 初始化 33](#_Toc510529482)

[1、 注册角色 33](#_Toc510529483)

[2、 用户登录游戏(做活跃统计) 34](#_Toc510529484)

[3、 付费(开通VIP、米币购买道具、人民币购买游戏内一级货币 34](#_Toc510529485)

[4、 自定义统计项接口(只支持统计人数人次) 34](#_Toc510529486)

[五、web端统计或AS统计 35](#_Toc510529487)

[六、Java统计 37](#_Toc510529488)

[0、 初始化sdk接口 37](#_Toc510529489)

[1、 登录游戏验证密码(verify\_passwd) 37](#_Toc510529490)

[2、 注册角色(reg\_role) 38](#_Toc510529491)

[3、 登录(login\_online) 39](#_Toc510529492)

[4、 登出(logout) 40](#_Toc510529493)

[5、 在线人数统计(online\_count) 41](#_Toc510529494)

[6、 用户升级(level\_up) 42](#_Toc510529495)

[7、 付费（pay） 42](#_Toc510529496)

[8、 免费获得游戏币(obtain\_golds) 44](#_Toc510529497)

[9、 使用游戏币购买道具(buy\_item) 45](#_Toc510529498)

[10、 消耗游戏币（use\_golds） 46](#_Toc510529499)

[11、 接收任务（accept\_task） 47](#_Toc510529500)

[12、 完成任务（finish\_task） 47](#_Toc510529501)

[13、 放弃任务（abort\_task） 48](#_Toc510529502)

[14、 获得精灵（obtain\_spirit） 49](#_Toc510529503)

[15、 失去精灵（lose\_spirit） 50](#_Toc510529504)

[16、 退订VIP服务（unsubscribe） 51](#_Toc510529505)

[17、 销户VIP（cancel\_acct) 52](#_Toc510529506)

[18、 新用户注册转化（new\_trans） 52](#_Toc510529507)

[19、 自定义统计（log） 53](#_Toc510529508)

**修订历史记录**

|  |  |  |
| --- | --- | --- |
| **日期** | **作者** | **修订内容** |
| 2013.11 | Henry | 创建文档 |
| 2014.4.10 | Lynn | “五、web端统计或AS统计“增加三级统计项调用方法 |
| 2014.4.15 | Ping | “三、添加调用方法”，增加9、获得精灵；10、失去精灵 |
| 2014.04.23 | lynn | “三、添加调用方法”，修改1~4接口中的os字段说明，页游、手游传不同值 |
| 2014.05.07 | Lynn | 1、在“三、添加调用方法”中增加接口调用责任方说明；  2、在“二、接入流程”中完善数据测试方法“； |
| 2014.05.08 | Lynn | 增加接口调用的**阅读指引**，增加每个接口的**用途说明** |
| 2014.06.05 | Berry | 制定手游页游两套接入标准，去除不用的接口，更改传输数量标准 |
| 2014.07.02 | Ping | 17、新用户注册转化 |
| 2015.03.16 | Kendy | 调整文档，添加用户自定义统计新接口 |
| 2015.06.01 | Lynn | 游戏币相关接口兼容性更改说明，涉及obtain\_golds、buy\_item、use\_golds |
| 2016.06.01 | Mukade | 六、Java统计 |
| 2017.07.13 | Mukade | 创建4.0版本文档，引入新系统tms日志格式（c++） |
| 2018.03.28 | Mukade | 更新至4.1版本，修改构造器为通过参数决定是否落新/老统计，取消serverID，修改c++ new\_trans接口 |

# 

# 一、综述

## 适用范围

淘米数据平台帮助各游戏部门解决玩家数据收集到数据标准化分析的全部繁琐过程，以行业标准形式展现于报表中。

平台提供2种形式的调用： 1） SDK库的调用，落到本地服务器，目前已提供语言SDK包括C++、PHP、C程序、Java、AS等。2）发送http请求，直接发送相关统计。

## 统计标准

用户数：以账号（米米号）计数。

付费: 指购买游戏币或开通VIP包月服务。

## 问题联系人

数据分析平台的任何问题，请联系数据平台部

# 二、接入流程

温馨提示：您也可以阅读《【淘米游戏】数据分析平台接入流程—必读！.xlsx》来获知更详尽的接入流程。

## 为游戏申请GameID

申请地址：请预先申请一个整型的gameid,用于唯一标识您的这款游戏

## 向工程中添加logger库文件

获取统计SDK的logger压缩包并解压至本地，将libstatlogger64bit.a(64位)/libstatlogger64bit.so（64位）和statlogger.h文件加入您的工程代码中。

## 添加调用方法

* 后台C++：请参考【三、添加调用方法（C++）】
* PHP：请参考【四、添加调用方法（PHP）】
* WEB或AS：请参考【五、web端统计或AS统计】
* Java：请参考【六、Java统计】

## 进行数据测试

调用方法添加完毕后，应当进行数据测试，以确保方法调用正确。

**1、内网测试方法（推荐）：**

内网测试机上，建立如下目录：

/opt/taomee/stat/data/inbox 及 /opt/taomee/stat/data/log

对新系统tms，需要建立：/opt/taomee/stat/tmsdata/inbox

落的数据将会存放在inbox目录，是文本可直接打开查看。

**2、外网测试方法**

向数据平台部申请一个测试的gameid（一般为本项目gameid+100000），生成测试数据并登录系统页面查看。

# 三、添加调用方法(C++)

先初始化一个StatLogger类的实例,只需要每次后台服务启动时初始化一次即可，建议做成全局的。

统计数据一律强制写入/opt/taomee/stat/data/inbox目录下，新系统数据一律写入/opt/taomee/stat/tmsdata/inbox目录下。**需要提前建立该目录，并将data/tmsdata目录设为777的权限，否则程序无法落下数据**。

## 调用前必读

接口分为**基础接口**和**自定义接口**。**基础接口**用来落基础统计项，字段定义相对比较固定。**自定义接口**用来落自定义事件等，相对更为灵活和自定义化。

基础接口中，stid、sstid、logID都是由SDK自动生成。

**用途中的表格**含有多条说明的，是指**如果参数全部填写完整**，将会在一次接口调用中，生成多条日志。例如：第3个接口login\_online，若传的player\_id, race**不为空**，sdk也会生成第2条，第3条日志。

## 接口用途及调用方归属

统计数据由多方调用，形成完整的统计项体系。包括游戏方（项目前端和后台开发）、账户支付平台（运营开发部）等，请各方明确各自需调用的接口。

1. **游戏方**

游戏方是用户的所有行为数据的来源，涉及**活跃、等级、任务、付费**等，对于VIP系统在平台维护的游戏，除了第1、16、17个接口外，其余均需要调用。各接口的用途参见目录。

1. **账户支付平台**

**VIP系统在平台维护的游戏，调用接口7、16、17。**

7、付费(pay)接口落vip相关的付费信息

16、退订VIP服务（unsubscribe）落取消自动续费和短信退订的的VIP信息

17、销户VIP（cancel\_acct）落因过期而被系统自动清理的VIP信息

## 基础接口生成的日志，其中的通用字段说明

|  |  |  |  |
| --- | --- | --- | --- |
| **字段** | **说明** | **必填?** | **默认值** |
| hip | 源数据发送IP | No | 接口自动落 |
| gid | 游戏id | Yes |  |
| zid | 区id，例如：1区为1，全区为-1 | No，由构造函数生成 | -1 |
| sid | 服id，例如：1服为1，全服为-1 | No，由构造函数生成 | -1 |
| pid | 平台id，例如：某游戏在多个平台运营时，需要区分平台，平台id由接入方制定 | No，由构造函数生成 | -1 |
| ts | 落统计项时的时间戳 | No | 系统自动落 |
| acid | 帐号id，例如：米米号 | Yes | -1 |
| plid | 角色id，可填id或字符串 | No | -1 |

## 构造函数（StatLogger）

**接口定义**

StatLogger(

bool writeToTongji, //是否写入统计平台

bool writeToTms, //是否写入新统计平台

int game\_id, //游戏ID,由数据分析平台统一分配

int16\_t zone\_id = -1, //区ID，缺省值-1表示全区或不分区

int16\_t svr\_id = -1, //服ID，缺省值-1标识全服或不分服

int16\_t site\_id = -1, //平台ID，缺省值为-1时代表全平台，如果这款游戏将来或者现在可能放到其他外部平台运营，那么这里填上对应平台的ID，淘米平台用1表示

int isgame = 1 //标识日志是来自游戏后台(1)还是公共平台组(0);游戏后台无需传该参数，采用默认值-1即可

);

**调用示例**

StatLogger logger(1,1,25);

## 登录游戏验证密码（verify\_passwd）

**用途**

验证密码的人数人次以及这些用户的地区分布；各个广告位来源的用户验证密码的人数人次；使用不同的浏览器进行验证密码的人数人次；使用不同设备、操作系统、分辨率、网络以及网络服务提供商进行验证用户名和密码的人数人次

|  |  |  |
| --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | logID |
| \_veripass\_ | \_veripass\_ | 1 |

**接口定义**

void verify\_passwd (

std::string acct\_id, //用户账户(米米号)

uint32\_t cli\_ip, //用户的IP地址, 无法获取时取值0

std::string ads\_id, //用户是从哪个广告渠道跳转过来的

std::string browser = "",//用户使用的浏览器

std::string device = "",//用户使用的设备

std::string os = "", //用户浏览器的flash版本。

std::string resolution = "", //用户屏幕的分辨率

std::string network = "", //用户使用的网络

std::string isp = "" //用户网络的服务提供商

);

**示例**

logger.verify\_passwd(“47159775”,“106.235.12.11”,“innermedia.taomee.seer.topbar”);//不填浏览器、设备等参数，则无法按此维度统计分析

## 创建游戏中角色（reg\_role）

**用途**

|  |  |  |  |
| --- | --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | 用途 | logID |
| \_newac\_ | \_newac\_ | 统计每天新注册的用户账户数以及这些账户的地区分布、使用的浏览器、设备、操作系统、分辨率、网络、网络服务提供商的人数人次 | 2 |
| \_newpl\_ | \_newpl\_ | 当player\_id参数非空且符合接口定义时落  统计每天新注册的角色数，即根据acct\_id和player\_id来唯一标识一个用户 | 3 |
| \_newrace\_ | 具体的职业名称 | 当race参数非空且符合接口定义时落  统计每天各个职业的新增用户数,根据acct\_id来标识 | 40 |

**接口定义**

void reg\_role(

std::string acct\_id, //用户账户(米米号)

std::string player\_id, //用户角色标识(单角色游戏可赋空值，多角色游戏可填角色id)，非空值时字符串长度要介于1~128，且不能包含|=\t中的任何一种字符

std::string race, //用户选择的职业(对无职业游戏，该字段可赋空值)，非空时长度介于1~128之间，不能以空格/下划线开头或结尾（会自动略去），且不能包含|=\t中的任何一种字符

uint32\_t cli\_ip,//用户的IP地址, 无法获取时传0

std::string ads\_id, //用户是从哪个广告渠道跳转过来的

std::string browser = "",//用户使用的浏览器

std::string device = "",//页游，填写空值

std::string os = "",//用户浏览器的flash版本

std::string resolution = "",//用户屏幕的分辨率

std::string network = "", //用户使用的网络

std::string isp = ""//用户网络的服务提供商

注意：ads\_id/browser/device/os/resolution/network/isp不能包含非法字符|=\t中的任何一种，否则对应字段数据无法统计

);

**示例**

logger.reg\_role(“47159775”, “1383019208”, “魔法师”,” 61.155.182.56”, “innermedia.taomee.mole.banner”);//这里不按照浏览器、设备等指标统计；如果项目需要,可带上相应参数

## 登录游戏online服务器(login\_online)

**用途**

|  |  |  |  |
| --- | --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | 用途 | logID |
| \_lgac\_ | \_lgac\_ | 1. 统计帐号登录人数人次，也就是活跃用户以及活跃用户的地区分布 2. VIP和非VIP用户的登录人数人次 3. 每个等级的用户登录人数人次   4、使用不同的浏览器登录人数人次，不同设备登录人数人次 不同操作系统、分辨率、网络以及网路服务提供商登录人数人次 | 5 |
| \_lgpl\_ | \_lgpl\_ | 当player\_id参数不为空且符合接口定义时落  根据角色统计登录人数人次，以acct\_id和player\_id唯一标识用户 | 6 |
| \_lgrace\_ | 各职业名称 | 当race参数不为空且符合接口定义时落  统计各个职业每天的登录人数人次 | 41 |

**接口定义**

void login\_online(

std::string acct\_id, //用户账户(米米号)

std::string player\_id,//用户角色标识(单角色游戏可赋空值，多角色游戏可填写角色id)，非空时长度介于1~128之间且不包含=|\t特殊字符中的任何一种

std::string race, //用户选择的职业(对无职业游戏，该字段可赋空值)，非空时长度介于1~128之间，不能以空格/下划线开头或结尾（会自动略去），且不能包含|=\t中的任何一种字符

bool isvip, //是否VIP用户（true:是 false：否）

int lv,//用户当前等级，取值范围必须介于0~5000之间

uint32\_t cli\_ip,//用户的IP地址, 无法获取时取值0

std::string ads\_id, //用户是从哪个广告渠道跳转过来的

std::string zone = "",//用户登陆区服（不区分区服的，填空值），不能以空格/下划线开头或结尾（会被自动略去）

std::string browser = "", //页游填用户使用的浏览器类型（如firefox, IE）

std::string device = "",//页游填写空值

std::string os = "", //用户浏览器的flash版本。

std::string resolution = "", //用户屏幕的分辨率

std::string network = "", //用户使用的网络

std::string isp = ""//用户网络的服务提供商

);

**示例**

logger.login\_online(“47159775”, “1383019208”, “”, true, 15, “115.12.116.57”, “网通一区”, “ie8”, “iphone4”, “debian”,”1280\*768”, “wifi”, “网通”);

## 退出游戏online服务器(logout)

**用途**

|  |  |  |  |
| --- | --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | 用途 | logID |
| \_logout\_ | \_logout\_ | 可统计每天登出游戏的人数人次、总的在线时长(单位秒)以及时长区间分布 | 8 |

**接口定义**

void logout(

std::string acct\_id, //用户账户(米米号)

bool isvip, //是否VIP用户(true:是 false:否)

int lv, //用户退出时的等级，必须介于0~5000

int oltime //本次用户总共的在线时长，单位为秒，取值介于0~864000之间

);

**示例**

logger.logout(“47159775”, true, 13,3204);

## 统计当前在线人数(online\_count)

**用途**

|  |  |  |  |
| --- | --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | 用途 | logID |
| \_olcnt\_ | \_olcnt\_ | 统计游戏当前在线人数，每分钟至少调用一次 | 9 |

**接口定义**

void online\_count(

int cnt, //当前在线人数，不能填负值否则将不会生成统计日志

std::string zone="" //1、不能以下划线/空格作为开头或结尾（会被自动略去），为空时填入默认值”\_all\_”，用于统计总在线人数；2举例：填写“电信”或“网通”，则分别统计电信或网通的在线人数。

);

**示例**

logger.online\_count(103476);//每分钟至少调用一次

## 用户升级(level\_up)

**用途**

|  |  |  |  |
| --- | --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | 用途 | logID |
| \_aclvup\_ | \_aclvup\_ | 统计每天的用户等级分布 | 10 |
| \_racelvup\_ | 各职业名称 | 当race参数不为空字符串时落  统计各职业用户的等级分布 | 11 |

**接口定义**

void level\_up(

std::string acct\_id, //用户帐号(米米号)

std::string race, //用户职业(如果不需要区分职业，赋值为空) ，非空时长度介于1~128之间，不能以空格/下划线开头或结尾（会自动略去），且不能包含|=\t中的任何一种字符

int lv //升之后的等级，取值介于0~5000之间

);

**示例**

每次用户升级时

logger.level\_up(“47159775”, “”, 20);

## 付费（pay）

**此统计关系到付费数据的准确性，请重点关注。**

该接口统计的是指米币级别的收入。是指用人民币或米币兑换游戏币或购买游戏内VIP服务。该接口涉及到多种类型的付费统计，需要**游戏后台、BOSS平台、PAY网站后台**同时落取。

**一、兑换游戏币**

1、没有从游戏跳转到pay页面的兑换（直接在游戏内部完成的兑换），由**游戏后台**接入统计。

2、从游戏跳转到pay页面的兑换，由**PAY网站后台**接入统计。

**二、购买VIP服务**

1、VIP系统由平台维护的，由BOSS平台接入统计

2、VIP系统不在平台维护的，如果是用人民币或米币购买VIP，由游戏后台接入统计。（如果是用游戏币购买VIP，则不需要接入）

**以下是产生的统计项日志：**

|  |  |  |  |
| --- | --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | 用途 | logID |
| \_acpay\_ | \_acpay\_ | 统计米币付费总额 | 12 |
| \_acpay\_ | \_buyitem\_ | 当pay\_reason参数填StatLogger::pay\_buy时落统计按条总额 | 14 |
| \_buyitem\_ | \_mibiitem\_ | 当pay\_reason参数填StatLogger::pay\_buy时落统计通过米币购买的道具人数人次、销售数量、销售金额 | 16 |
| \_acpay\_ | \_vipmonth\_ | 当pay\_reason参数填StatLogger::pay\_vip时落  包月付费总额 | 13 |
| \_buyvip\_ | \_buyvip\_ | 当pay\_reason参数填StatLogger::pay\_vip时落统计各个包月时长的人数人次以及付费总额 | 15 |

**接口定义**

void pay(

std::string acct\_id,//用户账户(米米号)

bool isvip, //是否VIP用户(true:是，false:否)

float pay\_amount, //付费额度，见详细说明

CurrencyType currency, //货币类型，见详细说明

PayReason pay\_reason,//支付类型，见详细说明

std::string outcome,//支付产生的物品，见详细说明

int outcnt,//获得数量，见详细说明

std::string pay\_channel = "\_mibiaccount\_"//支付渠道号，见详细说明

);

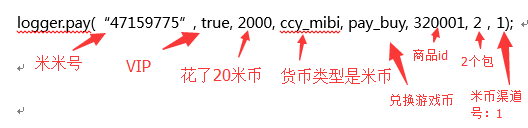
**参数详细说明**

|  |  |
| --- | --- |
| pay\_amount | 付费额度，**不能取负值，单位统一为”分”，例如花10米币，数量落1000** |
| currency | 货币类型，枚举类型  页游落**StatLogger::ccy\_mibi（米币）**  手游落**StatLogger::ccy\_cny（人民币）** |
| pay\_reason | 支付类型，枚举类型  兑换游戏币 落**StatLogger::pay\_buy**  VIP包月 落**StatLogger::pay\_vip** |
| outcome | 支付产生的物品，**当pay\_reason取pay\_buy时不能为空**，长度小于128，不能包含非法字符|=\t中的任何一种  兑换游戏币 落**商品id，由账户支付平台boss分配**  VIP包月 落**“1个月VIP”、“3个月VIP”等** |
| outcnt | 获得数量，**不能传负值，**  兑换游戏币 若**兑换率是1:10的填10\*米币数**；**兑换率是其它的，填钻石包的数量**，如兑换2个100钻石礼包，填2。  VIP包月落 **1,3,6,12**等 |
| pay\_channel | 支付渠道号，如支付宝、财付通等，由账户支付平台定的渠道号。不传时落默认值，传值时长度必须介于1~128之间且不包含非法字符|=\t中的任何一种  **页游的游戏后台请注意：游戏内兑换，渠道号填1，指米币渠道。** |

**示例：**

**1、页游：**

1） 游戏内兑换游戏币，由**游戏后台调用**（花了20米币购买2个320001对应的游戏币包）



2）VIP包月，由账户支付平台调用（花10米币购买了1个月的VIP时长）

logger.pay(“47159775”, true, 1000,ccy\_mibi,pay\_vip,”1个月VIP”, 1, “短信”);

**2、手游：**

logger.pay(“47159775”, true, 300, ccy\_cny, pay\_charge, ”游戏金币”,30);

## 免费获得游戏币(obtain\_golds)

指通过游戏内赠送或完成任务奖励获得，而非通过玩家购买获得

**用途**

|  |  |  |  |
| --- | --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | 用途 | logID |
| \_getgolds\_ | \_systemsend\_ | 统计通过各种系统赠送途径获得游戏币的人数人次以及总数量 | 17 |

**接口定义**

void obtain\_golds(

std::string acct\_id,//用户账户(米米号)

int amt//获得金币的数量，取值需要介于0~1000000000之间

);

**示例**

在IMOLE游戏里面通过完成某个任务奖励了**10**个贝壳

logger.obtain\_golds(“47159876”, **10**);

## 使用游戏币购买道具(buy\_item)

**用途**

|  |  |  |  |
| --- | --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | 用途 | logID |
| \_buyitem\_ | \_coinsbuyitem\_ | 可统计通过游戏币购买道具的人数人次、销售数量、销售金额 | 18 |
| \_usegold\_ | \_usegold\_ | 统计消耗的游戏币数量 | 36 |
| \_usegold\_ | \_buyitem\_ | 统计使用游戏币购买道具 | 37 |

**接口定义**

void buy\_item(

std::string acct\_id,//用户账户(米米号)

bool isvip,//是否VIP用户(true:是 false：否)

int lv,//用户购买道具时的等级，取值介于0~5000之间

float pay\_amount,//支付的金币数量，取值必须大于0

std::string outcome,//购买的道具，空值时落-1，非空时长度需介于1~128之间，不包含特殊字符|=\t中的任何一种，utf-8编码

int outcnt//购买的道具数量，取值需大于0

);

**示例**

用户花了**20**游戏币购买10个元旦礼包

Logger.buy\_item(“34159876”, true, 13, **20**, “元旦礼包”, 10);

## 消耗游戏币（use\_golds）

是指除了购买游戏币道具之外的消耗方式

**用途**

|  |  |  |  |
| --- | --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | 用途 | logID |
| \_usegold\_ | \_usegold\_ | 统计消耗的游戏币数量 | 36 |
| \_usegold\_ | reason | 可统计消耗游戏币的数量、VIP和非VIP用户消耗游戏币数量以及各等级用户消耗的数量 | 75 |

**接口定义**

void use\_golds(

std::string acct\_id, //用户帐号(米米号)

bool isvip, //是否VIP用户(true:是 false:否)

std::string reason, //原因（开启新功能、跳过关卡等），开头结尾不能有空格/下划线，否则会被自动略去，长度介于1~128之间，不能包含特殊字符|=\t中的任何一种，且符合utf-8编码

int amt, //支付的金币数量，取值必须介于0~100000之间

int lv//用户等级，取值介于0~5000之间

);

**示例**

用户花**18**游戏币开启新功能

logger.user\_golds(“47169879”, true, “\_开启新功能\_”, **18**, 17);

## 接收任务（accept\_task）

**用途**

|  |  |  |  |
| --- | --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | 用途 | logID |
| \_getnbtsk\_ （新手任务） | 任务名称 | 统计相应类型任务的接收人数人次 | 42 |
| \_getmaintsk\_（主线任务） | 43 |
| \_getauxtsk\_（支线任务） | 44 |
| \_getetctsk\_（其它任务） | 45 |

**接口定义**

void accept\_task(

TaskType type, //任务类型，枚举类型，如新手任务：StatLogger::task\_newbie

std::string acct\_id, //用户(米米号)

std::string task\_name, //任务名称，不能以空格/下划线开头或结尾，长度需介于1~128之间，不包含特殊字符|=\t中的任何一种，utf-8编码

int lv//接收任务时的等级

);

**示例**

Logger.accept\_task(task\_newbie, “3781654”, “打开背包”,20);

## 完成任务（finish\_task）

**用途**

|  |  |  |  |
| --- | --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | 用途 | logID |
| \_donenbtsk\_（新手任务） | 任务名称 | 统计各个任务的完成人数人次以及完成任务时的用户等级分布 | 46 |
| \_donemaintsk\_（主线任务） | 47 |
| \_doneauxtsk\_（支线任务） | 48 |
| \_doneetctsk\_（其它任务） | 49 |

**接口定义**

void finish\_task(

TaskType type, //任务类型，枚举类型，如新手任务：StatLogger::task\_newbie

std::string acct\_id, //用户(米米号)

std::string task\_name, //任务名称，不能以空格/下划线开头或结尾，长度需介于1~128之间，不包含特殊字符|=\t中的任何一种，utf-8编码

int lv//完成任务时的等级

);

**示例**

logger.finish\_task(task\_newbie, “3781654”, “打开背包”, 18);

## 放弃任务（abort\_task）

**用途**

|  |  |  |  |
| --- | --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | 用途 | logID |
| \_abrtnbtsk\_（新手任务） | 任务名称 | 统计各个任务的放弃人数人次,以及放弃任务时的用户等级分布 | 50 |
| \_abrtmaintsk\_（主线任务） | 51 |
| \_abrtauxtsk\_（支线任务） | 52 |
| \_abrtetctsk\_（其它任务） | 53 |

**接口定义**

void abort\_task(

TaskType type, //任务类型，枚举类型，如新手任务：StatLogger::task\_newbie

std::string acct\_id, //用户(米米号)

std::string task\_name, //任务名称，不能以空格/下划线开头或结尾，长度需介于1~128之间，不包含特殊字符|=\t中的任何一种，utf-8编码

int lv//放弃任务时的等级

);

**示例**

logger.abort\_task(task\_newbie, “3781654”, “打开背包”, 19);

## 获得精灵（obtain\_spirit）

**用途**

|  |  |  |  |
| --- | --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | 用途 | logID |
| \_obtainspirit\_ | \_obtainspirit\_ | 统计**获得**精灵的数量、VIP或非VIP获得精灵的数量、各等级获得精灵的数量 | 20 |

**接口定义**

void obtain\_spirit(

std::string acct\_id, //用户帐号(米米号)

bool isvip, //是否VIP用户(true:是 false:否)

int lv, //获得精灵时的用户等级，取值介于0~5000之间

std::string spirit//获得的精灵名字，不能以空格/下划线开头或结尾

);

**示例**

每次用户获得精灵时

logger. logger.obtain\_spirit(stat\_itstr("47159775"),false, 20, "小火猴");

## 失去精灵（lose\_spirit）

**用途**

|  |  |  |  |
| --- | --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | 用途 | logID |
| \_losespirit\_ | \_losespirit\_ | 统计失去精灵的数量、VIP或非VIP获得精灵的数量、各等级获得精灵的数量 | 21 |

**接口定义**

void lose\_spirit(

std::string acct\_id, //用户帐号(米米号)

bool isvip, //是否VIP用户(true:是 false:否)

int lv, //失去精灵时的用户等级，取值介于0~5000之间

std::string spirit//失去的精灵名字，不能以空格/下划线开头或结尾

);

**示例**

每次用户失去精灵时

logger. logger.lose\_spirit(stat\_itstr("47159775"),false, 25, "小火猴");

## 退订VIP服务（unsubscribe）

取消米币自动续费和退订短信VIP服务

**用途**

|  |  |  |  |
| --- | --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | 用途 | logID |
| \_unsub\_ | \_unsub\_ | 统计每天指定渠道退订VIP服务的人数人次  对uc（即渠道）做item运算 | 22 |

对米币渠道而言，是取消自动续费，对短信渠道而言，是退订VIP服务，下月不再自动续费。

**接口定义**

void unsubscribe(

std::string acct\_id, //用户(米米号)，不能为空

UnsubscribeChannel uc//渠道，目前只有短信和米币两个渠道(StatLogger::uc\_duanxin和StatLogger::uc\_mibi)

);

**示例**

Logger.unsubscribe(“342352345”, StatLogger::uc\_duanxin);

Logger.unsubscribe(“342352345”, StatLogger::uc\_mibi);

## 销户VIP（cancel\_acct)

是指VIP时间到期后，系统定时销户VIP，淘米内部由平台BOSS系统调用

**用途**

|  |  |  |  |
| --- | --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | 用途 | logID |
| \_ccacct\_ | \_ccacct\_ | 统计各渠道销户VIP的人数人次  对channel字段做item运算 | 23 |

**接口定义**

void cancel\_acct(

std::string acct\_id, //用户(米米号)，不能为空

string channel//销户渠道，如支付宝、财付通等。Channel是字符串，由调用方和平台共同商定

);

**示例**

Logger.cancel\_acct(“342352345”, “zhifubao”);

## 新用户注册转化（new\_trans）

**用途**

统计游戏用户从注册到登录阶段，前后端各步骤用户流失的比例。

|  |  |  |  |
| --- | --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | 用途 | logID |
| \_newtrans\_ | 模型号步骤号 | 对新用户注册转化的每个步骤求人数人次 | 78 |

**接口定义**

void new\_trans(

int modelid, // 模型id，如果需要统计多个转换步骤比例的情况，可以落多个模型id，每一个模型id对应一套步骤id，**必须传正值，默认从1开始**

int stepid,// 新用户注册转化步骤步骤id，**必须传正值**

std::string acct\_id//用户(米米号)

);

**示例**

Logger. new\_trans (1,1,"342352345”);

Logger. new\_trans (1,2,"342352345”); //用户触发模型1步骤2

Logger. new\_trans (2,1,"342352345”); //用户触发模型2步骤1

Logger. new\_trans (2,2,"342352345”);

Logger. new\_trans (2,3,"342352345”);

## 自定义统计（log）

**用途**

1-18的基础统计项对游戏中的各类常见场景进行了覆盖，对于除次之外的各类游戏场景统计，引入统一的自定义统计接口，满足游戏提供更加灵活的统计需求。

**概述**

自定义统计可以分为二级统计和三级统计两种。二级统计项仅对stat\_name+sub\_stat\_name组合进行人次/人数的统计，功能相对简单。而三级统计项则是以stat\_name+sub\_stat\_name+key的组合作为统计单位，为key的对应值提供了更加丰富的操作符（op）。

**接口定义**

void log(

std::string stat\_name, //统计项名称，不能以下划线/空格开头或结尾，不能为空，长度最大为64个字节，否则会被屏蔽掉。

std::string sub\_stat\_name, //子统计项名称，不能以下划线/空格开头或结尾，不能为空，长度最大为64个字节，否则会被屏蔽掉。

std::string acct\_id, //用户账户(米米号)

std::string player\_id, //用户角色标识(如不需要分角色查看，传空字符串)

const StatInfo& info //附加信息，详见OpCode操作类型说明表

);

StatInfo对象接口声明如下：

void add\_info(std::string key, float value);

void add\_info(std::string key, std::string value);

void add\_op(OpCode op, std::string key1, std::string key2 = "");

void clear();

对于二级统计项的统计，直接调用log接口即可，可以不传StatInfo对象参数info；对于三级统计项，info参数封装了需要统计的key-value对以及操作符op，**因此需要先创建StatInfo对象，通过add\_info赋值key-value对，再通过add\_op声明key的操作符op，最后将StatInfo对象赋给log接口进行调用**。

**OpCode操作类型说明表**

|  |  |
| --- | --- |
| **操作类型** | **说明** |
| StatInfo::op\_sum(key) | 对指定key的value求和；组合键statname、 substatname 、key |
| StatInfo::op\_max(key) | 对指定key的value求最大值; 组合键statname、 substatname、 key |
| StatInfo::op\_set(key) | 对指定key的value做覆盖操作; 组合键statname 、substatname、 key |
| StatInfo::op\_ucount(key) | 对指定key做去重操作; 组合键statname 、substatname 、key |
| StatInfo::op\_item(key) | 对指定key的各value计算出对应的人数人次; 组合键statname 、substatname 、 acctid 、playerid 、key |
| StatInfo::op\_item\_sum(key1,key2) | 按照key1分类，对key2的value字段求和; 组合键statname、 substatname、 acctid、 playerid 、key1 、key2 |
| StatInfo::op\_item\_max(key1, key2) | 按照key1分类，对key2的value字段求最大值; 组合键statname、 substatname、 acctid、 playerid 、key1、 key2 |
| StatInfo::op\_item\_set(key1, key2) | 按照key1分类，对key2的value字段做覆盖操作，即取每天最后一个值; 组合键statname、 substatname、 acctid、 playerid、 key1、 key |
| StatInfo::op\_sum\_distr(key) | 需要依赖于区间分布，然后计算出对key的value字段求和后属于哪个区间的分布; 组合键statname 、substatname 、 acctid、 playerid、 key |
| StatInfo::op\_max\_distr(key) | 需要依赖于区间分布，然后计算出对key的value字段求最大值后属于哪个区间的分布; ; 组合键statname、 substatname、 acctid、 playerid 、key |
| StatInfo::op\_min\_distr(key) | 需要依赖于区间分布，然后计算出对key的value字段求最小值后属于哪个区间的分布; 组合键statname、 substatname、 acctid、 playerid、 key |
| StatInfo::op\_set\_distr(key) | 对key的 value字段做set操作后做分布; 组合键statname、 substatname、 acctid 、playerid、 key1 |
| StatInfo::op\_ip\_distr(key) | ip分布，需要根据IP地址查出对应的地区 |

**特别注意：**

1. **stat\_name和sub\_stat\_name均不能含-1%/?;|=\t这些字符，否则会被当成乱码屏蔽掉。**

|  |  |
| --- | --- |
| **字符** | **释义** |
| % | 中英文输入的百分号 |
| / | 右斜划线） |
| ? | 英文输入的问号 |
| ; | 英文输入的分号 |
| -1 | 负1 |
| | | 竖线 |
| = | 等号 |
| \t | 制表符 |

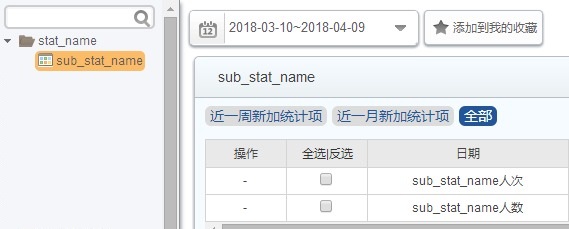
1. 接口中的key由各项目部自行定义，注意需要保证同一个项目内部不冲突，一次调用最多允许30个key-value对
2. 对key的取值不能为空，不能以 空格/下划线 开头或结束，StatInfo会自动过滤，比如\_KEY\_，会自动被修改成KEY; 不能有“= : , ; . | \t”字符中的任何一个，满足utf-8编码，长度小于128，否则会跳过该key-value对
3. 对value的取值不能为空，不能有“= | \t”字符中的任何一个，满足utf-8编码，长度小于128，否则会跳过该key-value对
4. add\_info不支持对key进行重复赋值，其值以第一次有效赋值为准（value的赋值合法，不含有非法字符）。但凡成功添加的key-value对都会记录在日志中，如需将key值纳入统计平台计算，还需要调用add\_op指定对key进行的操作符
5. add\_op声明的key1 、key2必须是已经通过add\_info方法添加好了的，对于item\_sum/item\_max/item\_set除外的op，key2不必添加。
6. add\_op支持通过多次调用对同一个key添加多个op操作符，也支持通过多次调用对多个key添加同一个op操作符。另外，对同一个key多次调用添加同一个op只会被视作一次操作。
7. 如需对添加的key-value对以及op操作符进行清空重置，调用StatInfo对象的clear()

**示例**

①**默认情况**：

logger.log(“stat\_name”, “sub\_stat\_name”, “acct\_id”, “player\_id”);

stat\_name->sub\_stat\_name（人数、人次）



② StatInfo::op\_sum(key)/StatInfo::op\_max(key)/StatInfo::op\_set(key)

**StatInfo::op\_sum(key)**：

StatInfo info;

info.add\_info("key1", 5);

info.add\_op(StatInfo::op\_sum, "key1");

logger.log("stat\_name", "sub\_stat\_name", "acct\_id", "player\_id", info);

info.clear();

info.add\_info("key1",6);

info.add\_op(StatInfo::op\_sum, "key1");

logger.log("stat\_name", "sub\_stat\_name", "acct\_id", "player\_id", info)

stat\_name->sub\_stat\_name->sub\_stat\_namekey1求和(人数、人次、key1对应value总和)



**StatInfo::op\_max(key)**：

StatInfo info;

info.add\_info("key1", 5);

info.add\_op(StatInfo::op\_max, "key1");

logger.log("stat\_name", "sub\_stat\_name", "acct\_id", "player\_id", info);

info.clear();

info.add\_info("key1",6);

info.add\_op(StatInfo::op\_max, "key1");

logger.log("stat\_name", "sub\_stat\_name", "acct\_id", "player\_id", info);

stat\_name->sub\_stat\_name->sub\_stat\_namekey1最大(人数、人次、key1对应value最大值)



**StatInfo::op\_set(key)**：

StatInfo info;

info.add\_info("key1", 5);

info.add\_op(StatInfo::op\_set, "key1");

logger.log("stat\_name", "sub\_stat\_name", "acct\_id", "player\_id", info);

info.clear();

info.add\_info("key1",6);

info.add\_op(StatInfo::op\_set, "key1");

logger.log("stat\_name", "sub\_stat\_name", "acct\_id", "player\_id", info);

stat\_name->sub\_stat\_name->sub\_stat\_namekey1(人数、人次、key1对应value最后值)



③  **StatInfo::op\_ucount(key)**：

StatInfo info;

info.add\_info("key1", 5);

info.add\_op(StatInfo::op\_ucount, "key1");

logger.log("stat\_name", "sub\_stat\_name", "acct\_id", "player\_id", info);

info.clear();

info.add\_info("key1",6);

info.add\_op(StatInfo::op\_ucount, "key1");

logger.log("stat\_name", "sub\_stat\_name", "acct\_id", "player\_id", info);

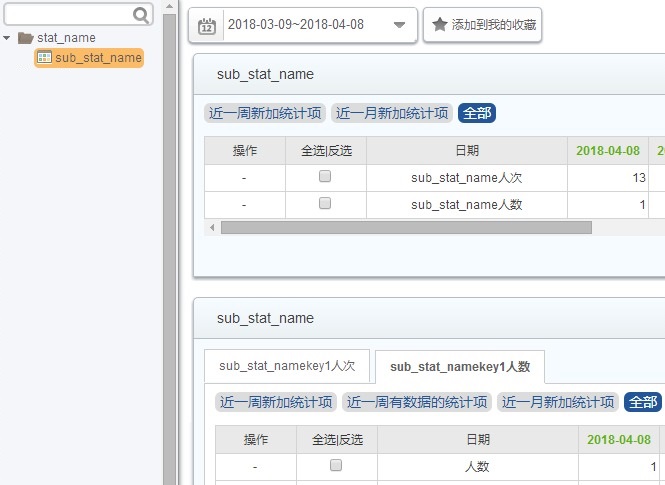
info.clear();

info.add\_info("key1",5);

info.add\_op(StatInfo::op\_ucount, "key1");

logger.log("stat\_name", "sub\_stat\_name", "acct\_id", "player\_id", info);

stat\_name->sub\_stat\_name->sub\_stat\_namekey1人数->人数(人数、人次、key1人数)



**④ StatInfo::op\_item(key)：**

StatInfo info;

info.add\_info("key1", 5);

info.add\_op(StatInfo::op\_item, "key1");

logger.log("stat\_name", "sub\_stat\_name", "acct\_id", "player\_id", info);

info.clear();

info.add\_info("key1",6);

info.add\_op(StatInfo::op\_item, "key1");

logger.log("stat\_name", "sub\_stat\_name", "acct\_id", "player\_id", info);

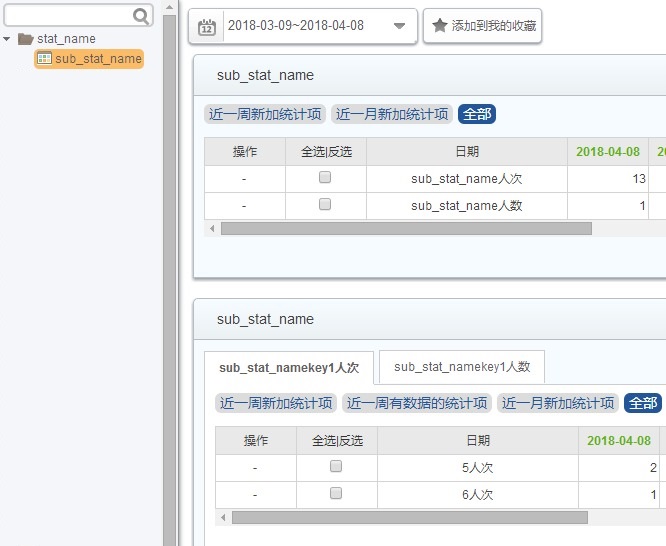
info.clear();

info.add\_info("key1",5);

info.add\_op(StatInfo::op\_item, "key1");

logger.log("stat\_name", "sub\_stat\_name", "acct\_id", "player\_id", info);

stat\_name->sub\_stat\_name->sub\_stat\_namekey1人次/sub\_stat\_namekey1人数(人数、人次、key1各value出现的人次/人数)

⑤ StatInfo::op\_item\_sum(key1,key2)/ StatInfo::op\_item\_max(key1,key2)/ StatInfo::op\_item\_set(key1,key2)

**StatInfo::op\_item\_sum(key1,key2)：**

StatInfo info;

info.add\_info("key1", 5);

info.add\_info("key2", 6);

info.add\_op(StatInfo::op\_item\_sum, "key1","key2");

logger.log("stat\_name", "sub\_stat\_name", "acct\_id", "player\_id", info);

info.clear();

info.add\_info("key1",9);

info.add\_info("key2",10);

info.add\_op(StatInfo::op\_item\_sum, "key1","key2");

logger.log("stat\_name", "sub\_stat\_name", "acct\_id", "player\_id", info);

info.clear();

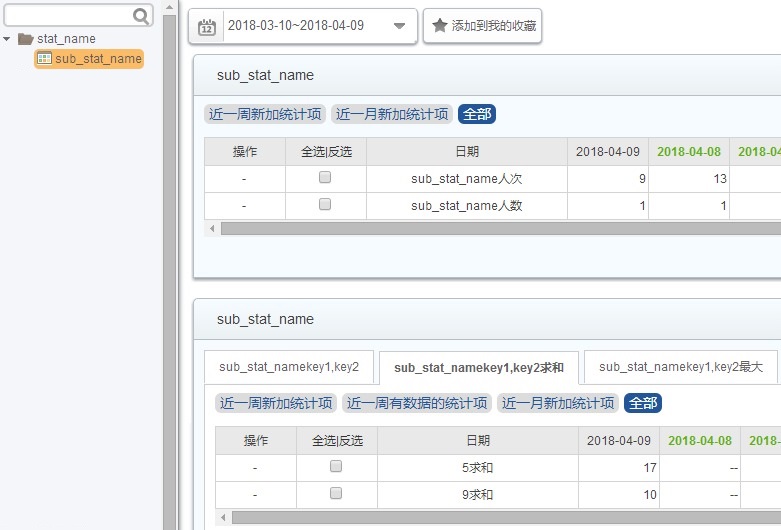
info.add\_info("key1",5);

info.add\_info("key2",11);

info.add\_op(StatInfo::op\_item\_sum, "key1","key2");

logger.log("stat\_name", "sub\_stat\_name", "acct\_id", "player\_id", info);

stat\_name->sub\_stat\_name->sub\_stat\_namekey1,key2求和（人次、人数、key1取各不同值时key2对应值的总和）



**StatInfo::op\_item\_max(key1, key2)：**

StatInfo info;

info.add\_info("key1", 5);

info.add\_info("key2", 6);

info.add\_op(StatInfo::op\_item\_max, "key1","key2");

logger.log("stat\_name", "sub\_stat\_name", "acct\_id", "player\_id", info);

info.clear();

info.add\_info("key1",9);

info.add\_info("key2",10);

info.add\_op(StatInfo::op\_item\_max, "key1","key2");

logger.log("stat\_name", "sub\_stat\_name", "acct\_id", "player\_id", info);

info.clear();

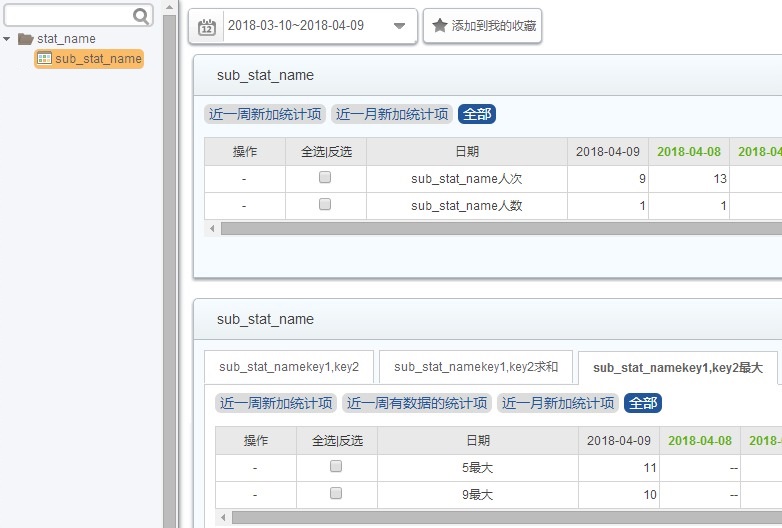
info.add\_info("key1",5);

info.add\_info("key2",11);

info.add\_op(StatInfo::op\_item\_max, "key1","key2");

logger.log("stat\_name", "sub\_stat\_name", "acct\_id", "player\_id", info);

stat\_name->sub\_stat\_name->sub\_stat\_namekey1,key2最大（人次、人数、key1取各不同值时key2对应值的最大值）



**StatInfo::op\_item\_set(key1, key2)：**

StatInfo info;

info.add\_info("key1", 5);

info.add\_info("key2", 11);

info.add\_op(StatInfo::op\_item\_set, "key1","key2");

logger.log("stat\_name", "sub\_stat\_name", "acct\_id", "player\_id", info);

info.clear();

info.add\_info("key1",9);

info.add\_info("key2",10);

info.add\_op(StatInfo::op\_item\_set, "key1","key2");

logger.log("stat\_name", "sub\_stat\_name", "acct\_id", "player\_id", info);

info.clear();

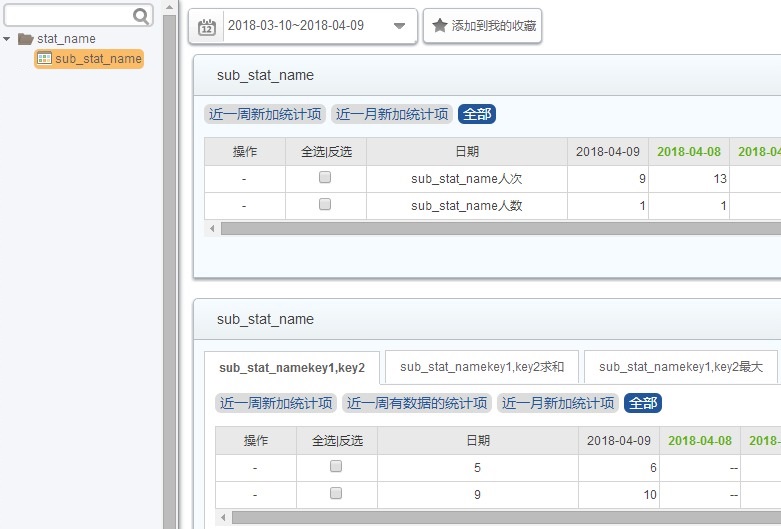
info.add\_info("key1",5);

info.add\_info("key2",6);

info.add\_op(StatInfo::op\_item\_set, "key1","key2");

logger.log("stat\_name", "sub\_stat\_name", "acct\_id", "player\_id", info);

stat\_name->sub\_stat\_name->sub\_stat\_namekey1,key2（人次、人数、key1取各不同值时key2对应值的最晚值）



⑥StatInfo::op\_sum\_distr(key)/StatInfo::op\_max\_distr(key)/Statino::op\_min\_distr(key)

这一项用于统计区间分布，只是这里的区间需要策划或相关开发人员去配置

StatInfo info;

info.add\_info(“key1”, 56);

info.add\_op(StatInfo::op\_sum\_distr, “key1”);

logger.log(“stat\_name”，“sub\_stat\_name”, “acct\_id”, “player\_id”, info);

就会将以用户为单位，计算每个人key1对应值的总和，然后看属于哪个区间，做分布

Op\_max\_distr和op\_min\_distr 类似,只是分别去每个用户的最大值或最小值而已.

⑦ StatInfo::op\_set\_distr(key)

这一项用于求最晚值分布

StatInfo info;

info.add(“key1”, 19);

info.add\_op(StatInfo::op\_set\_distr, “key1”);

logger.log(“stat\_name”, “sub\_stat\_name”, “acct\_id”, “player\_id”, info);

## 根据统计项ID落自定义统计数据（custom\_log）

**用途**

|  |  |  |
| --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | **用途** |
| \_msgid\_ | stat\_id | 根据统计项内容对应的统计项ID落数据，得到自定义统计项数据。 |

前提是统计项id对应的信息已经先行上传到平台的统计项信息库里，平台有专门提供统计项的xml表上传的页面，并有相应的校验机制，保证上传的统计项信息无误。

此法简化了落自定义统计的代码，将规则写在一条统计项信息记录里，通过统计项id与统计项信息映射的关系来实现统计项的落取。

统计项信息包含，统计项id对应的stat\_name、sub\_stat\_name、item，操作类型等。

**接口说明**

void custom\_log(std::string acct\_id, uint32\_t stat\_id, float value=0);

参数说明

|  |  |
| --- | --- |
| acct\_id | 用户(米米号) |
| stat\_id | 统计项ID |
| value | 统计项的值，例如：道具数量 |

**用法**

1. 人数人次

Logger.custom\_log (“558955495”, 156453783);

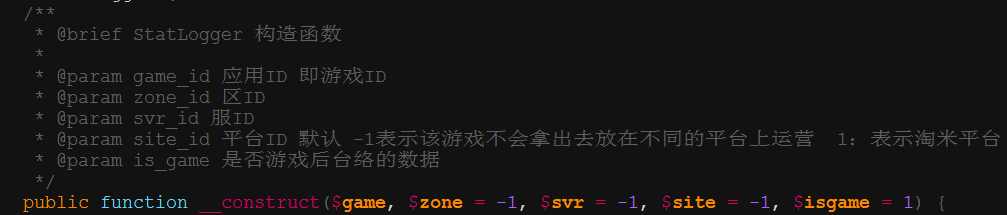
1. 数量和或最大值

Logger.custom\_log (“558955495”, 156453783，100);

# 四、添加调用方法(PHP)

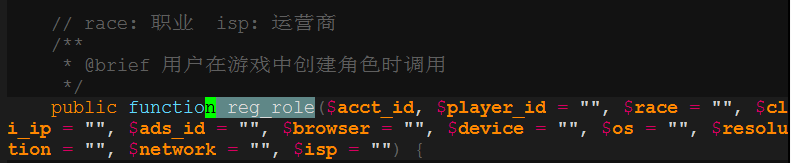
PHP提供的接口，除了第20个接口custom\_log不提供，其它接口与【三、添加调用方法(C++)】完全一致，添加方法请参照第三点。以下挑选几个接口示例如下。

## 初始化



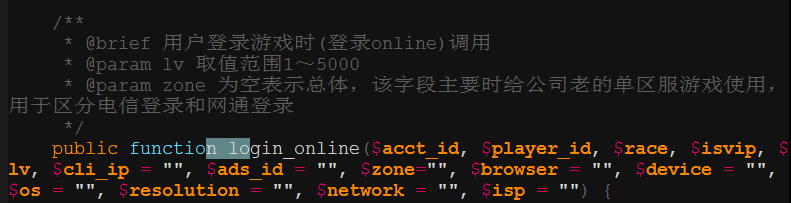
## 注册角色

接口定义:



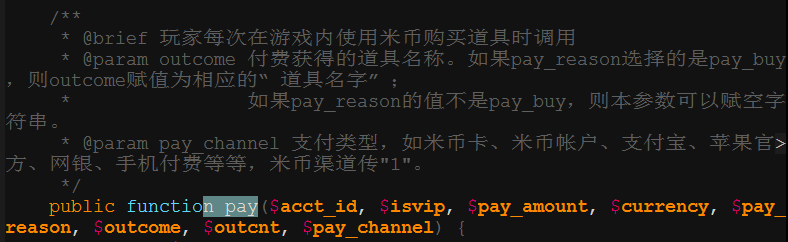
## 用户登录游戏(做活跃统计)

接口定义:

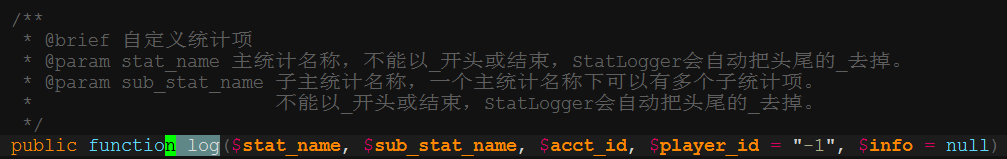


## 付费(开通VIP、米币购买道具、人民币购买游戏内一级货币

接口定义：(**注意付费金额的单位统一为 “分”)**



## 自定义统计项接口(只支持统计人数人次)

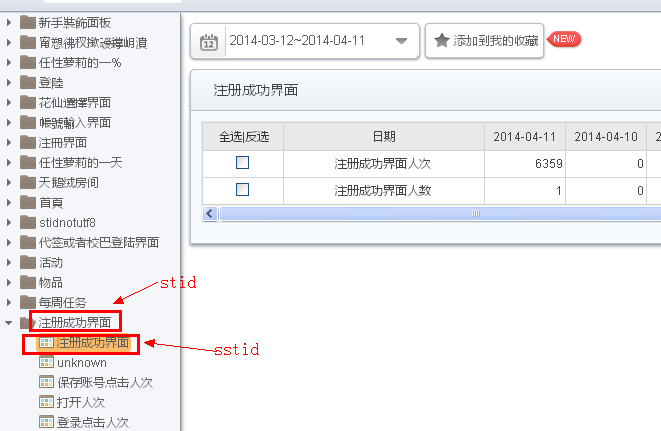


# 五、web端统计或AS统计

通过发送http请求的形式，支持人数人次的统计，提供二级和三级两种统计方式

http请求地址：

**二级统计项**：<http://newmisc.taomee.com/misc.js?gameid=gameid&stid=stid&sstid=sstid&uid=uid>



gameid:游戏ID(**必须**为数字)

stid: 统计项名称

sstid: 子统计项名称

uid: 用户米米号(**必须**为英文或数字字符串，若获取不到米米号可不填写)

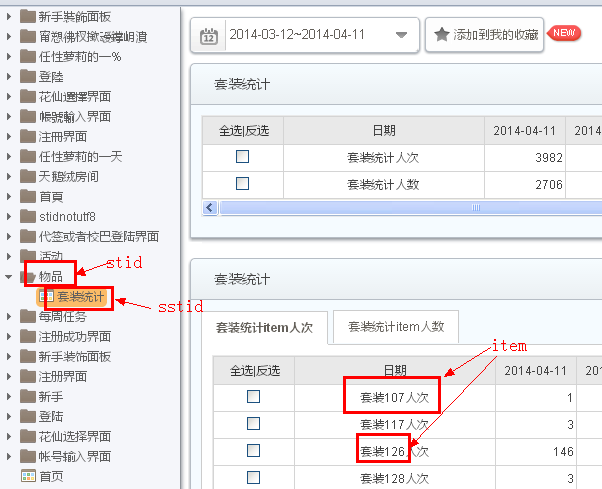
其中，stid、sstid可填写中文、英文或数字字符串，但若不能避免乱码问题，请尽量选择填写英文或数字字符串

**三级统计项**：

[http://newmisc.taomee.com/misc.js?gameid=gameid&stid=stid&sstid=sstid&uid=uid&item=item](http://misc.taomee.com/weblogger.php?gameid=gameid&stid=stid&sstid=sstid&uid=uid&item=item)

例如：

<http://newmisc.taomee.com/misc.js?gameid=2&stid>=”物品”&sstid=”套装统计”&uid=500020&item=“套装107”



gameid:游戏ID(**必须**为数字)

stid: 统计项名称

sstid: 子统计项名称

uid: 用户米米号(**必须**为英文或数字字符串，若获取不到米米号可不填写)

item：第3级统计项名称

其中，stid、sstid、item可填写中文、英文或数字字符串，但若不能避免乱码问题，请尽量选择填写英文或数字字符串

# 六、Java统计

## 初始化sdk接口



**参数说明**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 备注 |
| game | int | 游戏id，游戏标识 |
| zone | int | 游戏区id（没有就设置为-1) |
| svr | int | 游戏服id (没有就设置为-1) |
| site | int | 游戏平台ID(没有就设置为-1) |
| isgame | int | 标示是否为游戏，游戏那边直接设置为1 |

**调用示例**



## 登录游戏验证密码(verify\_passwd)

**用途**

验证密码的人数人次以及这些用户的地区分布

各个广告位来源的用户验证密码的人数人次

使用不同的浏览器进行验证密码的人数人次

|  |  |
| --- | --- |
| 统计项（stid） | 子统计项（sstid） |
| \_veripass\_ | \_veripass\_ |

**接口定义**

**参数说明**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 备注 |
| acct\_id | String | 用户账户(米米号) |
| cli\_ip | String | 用户的IP地址, 无法获取时取值0 |
| ads\_id | String | 用户是从哪个广告渠道跳转过来的 |

**调用示例**

statLogger.verify\_passwd(“47159775”,“106.235.12.11”,“innermedia.taomee.seer.topbar”);//不填浏览器、设备等参数，则无法按此维度统计分析

## 注册角色(reg\_role)

|  |  |  |
| --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | **用途** |
| \_newac\_ | \_newac\_ | 统计每天新注册的用户账户数人数人次 |
| \_newpl\_ | \_newpl\_ | 统计每天新注册的角色数，即根据acct\_id和player\_id来唯一标识一个用户 |
| \_newrace\_ | 具体的职业名称 | 统计每天各个职业的新增用户数,根据acct\_id来标识 |

**接口定义**



**参数说明**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 备注 |
| acct\_id | String | 账户ID |
| player\_id | String | 角色ID |
| race | String | 角色名称 |
| cli\_ip | String | 用户客户端的Ip |
| ads\_id | String | 标记该用户是从哪个渠道过来的 |

**调用示例**

statLogger.reg\_role("456789321", "1380562965", "兔子", "10.1.6.130", "4399.com")

## 登录(login\_online)

|  |  |  |
| --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | **用途** |
| \_lgac\_ | \_lgac\_ | 1 统计帐号登录人数人次，也就是活跃用户以及活跃用户的地区分布  2 VIP和非VIP用户的登录人数人次  3每个等级的用户登录人数人次 |
| \_lgpl\_ | \_lgpl\_ | 根据角色统计登录人数人次，以acct\_id和player\_id唯一标识用户 |
| \_lgrace\_ | 各职业名称 | 统计各个职业每天的登录人数人次 |

**接口定义**



zone参数填写””;

**参数说明**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 备注 |
| acct\_id | String | 账户ID |
| player\_id | String | 角色ID |
| race | String | 角色名称 |
| isvip | boolean | 是否是vip |
| lv | Integer | 用户游戏等级 |
| cli\_ip | String | 用户客户端的Ip |
| ads\_id | String | 标记该用户是从哪个渠道过来的 |
| zone | String | 1、默认值为“”是统计总在线人数；  2填写“telecom”或“netcom”，则分别统计电信或网通的在线人数。 |

**调用示例**



## 登出(logout)

|  |  |  |
| --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | **用途** |
| \_logout\_ | \_logout\_ | 可统计每天登出游戏的人数人次、总的在线时长(单位秒)以及时长区间分布 |

**接口定义**



**参数说明**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 备注 |
| acct\_id | String | 用户账户(米米号) |
| isvip | boolean | 是否VIP用户(true:是 false:否) |
| lv | String | 用户退出时的等级 |
| oltime | Integer | 本次用户总共的在线时长 |

**调用示例**



## 在线人数统计(online\_count)

|  |  |  |
| --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | **用途** |
| \_olcnt\_ | \_olcnt\_ | 统计游戏当前在线人数，每分钟至少调用一次 |

**接口定义**



该接口每分钟调用一次，zone可以填写””:

**参数说明**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 备注 |
| cnt | int | 当前在线人数 |
| zone | String | 1、默认值为“”是统计总在线人数；  2填写“telecom”或“netcom”，则分别统计电信或网通的在线人数。 |

**调用示例**

statLogger.online\_count(12345,"");

## 用户升级(level\_up)

|  |  |  |
| --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | **用途** |
| \_aclvup\_ | \_aclvup\_ | 统计每天的用户等级分布 |
| \_racelvup\_ | \_racelvup\_ | 统计各职业用户的等级分布 |

**接口定义**

**参数说明**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 备注 |
| acct\_id | String | 用户帐号(米米号) |
| race | String | 用户职业(如果不需要区分职业，赋值为空) |
| lv | int | 升之后的等级 |

**调用示例**

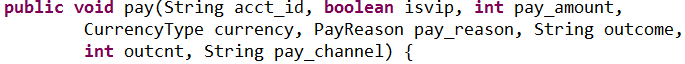
每次用户升级时

logger.level\_up(“47159775”, “”, 20);

## 付费（pay）

|  |  |  |
| --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | **用途** |
| \_acpay\_ | \_acpay\_ | 统计米币付费总额 |
| \_acpay\_ | \_buyitem\_，\_vipmonth\_ | 统计按条或包月付费总额 |
| \_buyvip\_ | \_buyvip\_ | 统计各个包月时长的人数人次以及付费总额 |
| \_buyitem\_ | \_mibiitem\_ | 统计通过米币购买的道具人数人次、销售数量、销售金额 |

**接口定义**

**参数说明**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 备注 |
| acct\_id | String | 用户账户(米米号) |
| isvip | boolean | 是否VIP用户(true:是，false:否) |
| pay\_amount | int | 付费额度,**单位统一为”分”，例如花10米币，数量落1000** |
| currency | CurrencyType | 货币类型，枚举类型  页游落CurrencyType.ccy\_mibi（米币）  手游落CurrencyType.ccy\_cny（人民币） |
| pay\_reason | PayReason | **支付类型**   |  |  | | --- | --- | | **类型** | **落的内容** | | 兑换游戏币 | PayReason.pay\_buy | | VIP包月 | PayReason.pay\_vip | |
| outcome | String | **支付产生的物品（不能为空！）**   |  |  | | --- | --- | | **类型** | **落的内容** | | 兑换游戏币 | 商品id，由账户支付平台boss分配 | | VIP包月 | “1个月VIP”、“3个月VIP”等 | |
| outcnt | int | 获得数量   |  |  | | --- | --- | | **类型** | **落的内容** | | 兑换游戏币 | 兑换率是1:10的填**10\*米币数**； | | VIP包月 | 1,3,6,12等 | |
| pay\_channel | String | **不能为空！**支付渠道号，如支付宝、财付通等，由账户支付平台定的渠道号。  **页游的游戏后台请注意：游戏内兑换，渠道号填1，指米币渠道。** |

**调用示例**

statLogger.pay("82333234", false, 15, CurrencyType.ccy\_mibi, PayReason.pay\_buy, "双倍经验药剂", 10, "支付宝");

## 免费获得游戏币(obtain\_golds)

|  |  |  |
| --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | **用途** |
| \_getgolds\_ | \_systemsend\_ | 统计通过各种系统赠送途径获得游戏币的人数人次以及总数量 |

**接口定义**



**参数说明**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 备注 |
| acct\_id | String | 用户账户(米米号) |
| amt | int | 获得金币的数量 |

**调用示例**

在IMOLE游戏里面通过完成某个任务奖励了**10**个贝壳

statLogger.obtain\_golds(“47159876”, 10);

## 使用游戏币购买道具(buy\_item)

|  |  |  |
| --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | **用途** |
| \_buyitem\_ | \_coinsbuyitem\_ | 可统计通过游戏币购买道具的人数人次、销售数量、销售金额 |

**接口定义**



**参数说明**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 备注 |
| acct\_id | String | 用户账户(米米号) |
| is\_vip | boolean | 是否VIP用户(true:是 false：否) |
| lv | int | 用户购买道具时的等级 |
| pay\_amount | int | 支付的金币数量 |
| outcome | String | 购买的道具 |
| outcnt | int | 购买的道具数量 |

**调用示例**

用户花了**20**游戏币购买10个元旦礼包

statLogger.buy\_item(“34159876”, true, 13, 20, “元旦礼包”, 10);

## 消耗游戏币（use\_golds）

|  |  |  |
| --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | **用途** |
| \_userglods\_ | reason | 可统计各原因消耗游戏币的数量、VIP和非VIP用户消耗的游戏币数量以及各等级用户消耗的数量 |

**接口定义**



**参数说明**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 备注 |
| acct\_id | String | 用户帐号(米米号) |
| isvip | boolean | 是否VIP用户(true:是 false:否) |
| reason | String | 原因（开启新功能、跳过关卡等） |
| amt | float | 支付的金币数量 |
| lv | int | 用户等级 |

**调用示例**

用户花**18**游戏币开启新功能

logger.user\_golds(“47169879”, true, “\_开启新功能\_”, **18**, 17);

## 接收任务（accept\_task）

|  |  |  |
| --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | **用途** |
| \_getnbtsk\_ （新手任务）  \_getmaintsk\_（主线任务）  \_getauxtsk\_（支线任务）  \_getetctsk\_（其它任务） | 任务名称 | 统计相应类型任务的接收人数人次 |

**接口定义**



**参数说明**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 备注 |
| type | TaskType | 任务类型(新手任务落TaskType.task\_newbie 主线任务落TaskType.task\_story支线任务落TaskType.task\_supplement ) |
| acct\_id | String | 用户(米米号) |
| task\_name | String | 任务名称 |
| Lv | int | 接收任务时的等级 |

**调用示例**

statLogger.accept\_task(TaskType.task\_newbie,“3781654”,“打开背包”,20);

## 完成任务（finish\_task）

|  |  |  |
| --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | **用途** |
| \_donenbtsk\_（新手任务）  \_donemaintsk\_（主线任务）  \_doneauxtsk\_（支线任务）  \_doneetctsk\_（其它任务） | 任务名称 | 统计各个任务的完成人数人次以及完成任务时的用户等级分布 |

**接口定义**



**参数说明**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 备注 |
| type | TaskType | 任务类型(新手任务落TaskType.task\_newbie 主线任务落TaskType.task\_story 支线任务落TaskType.task\_supplement ) |
| acct\_id | String | 用户(米米号) |
| task\_name | String | 任务名称 |
| lv | int | 完成任务时的用户等级 |

**调用示例**

statLogger.finish\_task(TaskType.task\_newbie, “3781654”, “打开背包”, 18);

## 放弃任务（abort\_task）

|  |  |  |
| --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | **用途** |
| \_abrtnbtsk\_（新手任务）  \_abrtmaintsk\_（主线任务）  \_abrtauxtsk\_（支线任务）  \_abrtetctsk\_（其它任务） | 任务名称 | 统计各个任务的放弃人数人次,以及放弃任务时的用户等级分布 |

**接口定义**



**参数说明**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 备注 |
| type | TaskType | 任务类型(新手任务落TaskType.task\_newbie 主线任务落TaskType.task\_story 支线任务落TaskType.task\_supplement ) |
| acct\_id | String | 用户(米米号) |
| task\_name | String | 任务名称 |
| lv | int | 放弃任务时的等级 |

**调用示例**

statLogger.abort\_task(task\_newbie, “3781654”, “打开背包”, 19);

## 获得精灵（obtain\_spirit）

|  |  |  |
| --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | **用途** |
| \_obtainspirit\_ | \_obtainspirit\_ | 统计**获得**精灵的数量、VIP或非VIP获得精灵的数量、各等级获得精灵的数量 |

**接口定义**



**参数说明**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 备注 |
| acct\_id | String | 用户帐号(米米号) |
| isvip | boolean | 是否VIP用户(true:是 false:否) |
| lv | int | 获得精灵时的用户等级 |
| spirit | String | 获得的精灵名字 |

**调用示例**

每次用户获得精灵时

statLogger.obtain\_spirit("47159775",false, 20, "小火猴");

## 失去精灵（lose\_spirit）

|  |  |  |
| --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | **用途** |
| \_losespirit\_ | \_losespirit\_ | 统计**失去**精灵的数量、VIP或非VIP获得精灵的数量、各等级获得精灵的数量 |

**接口定义**



**参数说明**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 备注 |
| acct\_id | String | 用户帐号(米米号) |
| isvip | boolean | 是否VIP用户(true:是 false:否) |
| lv | int | 失去精灵时的用户等级 |
| spirit | String | 失去的精灵名字 |

**调用示例**

每次用户失去精灵时

logger. lose\_spirit("47159775",false, 25, "小火猴");

## 退订VIP服务（unsubscribe）

取消米币自动续费和退订短信VIP服务

|  |  |  |
| --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | **用途** |
| \_unsub\_ | \_unsub\_ | 统计每天指定渠道退订VIP服务的人数人次  对uc（即渠道）做item运算 |

对米币渠道而言，是取消自动续费，对短信渠道而言，是退订VIP服务，下月不再自动续费。

**接口定义**



**参数说明**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 备注 |
| acct\_id | String | 用户(米米号) |
| uc | UnsubscribeChannel | 渠道，目前只有短信和米币两个渠道(UnsubscribeChannel.uc\_duanxin和UnsubscribeChannel.uc\_mibi) |

**调用示例**

statLogger.unsubscribe("342352345", UnsubscribeChannel.uc\_duanxin);

statLogger.unsubscribe("342352345", UnsubscribeChannel.uc\_mibi);

## 销户VIP（cancel\_acct)

是指VIP时间到期后，系统定时销户VIP，淘米内部由平台BOSS系统调用

|  |  |  |
| --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | **用途** |
| \_ccacct\_ | \_ccacct\_ | 统计各渠道销户VIP的人数人次  对channel字段做item运算 |

**接口定义**



**参数说明**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 备注 |
| acct\_id | String | 用户(米米号) |
| channel | String | 销户渠道，如支付宝、财付通等。Channel是字符串，由调用方和平台共同商定 |

**调用示例**

statLogger.cancel\_acct(“342352345”, “zhifubao”);

## 新用户注册转化（new\_trans）

某游戏从注册米米号到进入游戏的所有步骤落统计项，以第一步为基数，层层做过滤计算

|  |  |  |
| --- | --- | --- |
| 统计项（stid） | 子统计项（sstid） | **用途** |
| \_newtrans\_ | 新用户注册转化步骤  （步骤名称见该专题文档） | 对新用户注册转化的每个步骤求人数人次 |

**接口定义**



**参数说明**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 备注 |
| step | NewTransStep | 新用户注册转化步骤 |
| acct\_id | String | 用户(米米号) |

**调用示例**

statLogger. new\_trans (NewTransStep.bGetLoginReq, “342352345”);

## 自定义统计（log）

**概述**

**接口定义**

void log(String stat\_name, String sub\_stat\_name, String acct\_id,

String player\_id, StatInfo info);

**参数说明**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 备注 |
| stat\_name | String | 统计项名称 |
| sub\_stat\_name | String | 子统计项名称 |
| acct\_id | String | 用户账户(米米号) |
| palyer\_id | String | 用户角色标识(如不需要分角色查看，传空字符串) |
| info | StatInfo | 附加信息 |

**用途**

默认用于统计子统计项的人数人次

**用法**

需统计：参与**“保护导航仪”**这个小游戏的人数人次以及输出的**赛尔豆**总量，代码如下：

StatLogger logger = new StatLogger(1,-1,-1,-1,-1);

StatInfo info = new StatInfo();

info.add\_info("赛尔豆", 843);

info.add\_op(OpCode.op\_sum ,"赛尔豆", "");

logger.log("游戏输出", "保护导航仪", "47189678", "", info);

其中，add\_info和add\_op接口声明如下

void add\_info(String key, double value);

void add\_info(String key, int value);

void add\_info(String key, String value);

void add\_op(OpCode op, String key1, String key2);

**OpCode操作类型说明表**

|  |  |
| --- | --- |
| **操作类型** | **说明** |
| op\_sum(key) | 对指定key的value求和；组合键statname、 substatname 、key |
| op\_max(key) | 对指定key的value求最大值; 组合键statname、 substatname、 key |
| op\_set(key) | 对指定key的value做set做操作; 组合键statname 、substatname、 key |
| op\_ucount(key) | 对指定key的value做去重操作; 组合键statname 、substatname 、key |
| op\_item(key) | 对指定key的value计算出对应的人数人次; 组合键statname 、substatname 、 acctid 、playerid 、key |
| op\_item\_sum(key1,key2) | 按照key1分类，对key2的value字段求和; 组合键statname、 substatname、 acctid、 playerid 、key1 、key2 |
| op\_item\_max(key1, key2) | 按照key1分类，对key2的value字段求最大值; 组合键statname、 substatname、 acctid、 playerid 、key1、 key2 |
| op\_item\_set(key1, key2) | 按照key1分类，对key2的value字段做set操作，即取每天最后一个值; 组合键statname、 substatname、 acctid、 playerid、 key1、 key |
| op\_sum\_distr(key) | 需要依赖于区间分布，然后计算出对key的value字段求和后属于哪个区间的分布; 组合键statname 、substatname 、 acctid、 playerid、 key |
| op\_max\_distr(key) | 需要依赖于区间分布，然后计算出对key的value字段求最大值后属于哪个区间的分布; ; 组合键statname、 substatname、 acctid、 playerid 、key |
| op\_min\_distr(key) | 需要依赖于区间分布，然后计算出对key的value字段求最小值后属于哪个区间的分布; 组合键statname、 substatname、 acctid、 playerid、 key |
| op\_set\_distr(key) | 对key的 value字段做set操作后做分布; 组合键statname、 substatname、 acctid 、playerid、 key1 |
| op\_ip\_distr(key) | ip分布，需要根据IP地址查出对应的地区 |

**特别注意：**

1. stat\_name和sub\_stat\_name均不能含以下字符，否则会被当成乱码屏蔽掉。

|  |  |
| --- | --- |
| **字符** | **释义** |
| % | 中英文输入的百分号 |
| / | 右斜划线） |
| ? | 英文输入的问号 |
| ; | 英文输入的分号 |
| -1 | 负1 |
| | | 竖线 |

1. stat\_name和sub\_stat\_name统计项名称不宜过长，最大为64个字节，否则会被屏蔽掉。
2. 接口中的key由各项目部自行定义，注意需要保证同一个项目内部不冲突，一次调用最多允许30个key-value对
3. 对key的取值不能以 “\_” 开头或结束，不能有“= : , ; . | \t?!”以及空格字符中的任何一个，否则程序运行会跳过该统计项
4. 对value的取值不能有“=|”、空格、制表符中的任何一个
5. 调用add\_op方法时，key1 、key2必须是已经通过add\_info方法添加好了的

**示例**

1. 默认类型(人数人次)

比如赛尔号->用户->玩家掉线统计 人数人次



赛尔号游戏ID为2

StatLogger logger = new StatLogger(2,-1,-1,-1,-1);

logger.log("用户", "玩家掉线统计", "48197896","-1" ,null);

1. OpCode.op\_sum(key) OpCode.op\_max(key) OpCode.op\_set(key)

比如赛尔号-> 游戏输出->保护导航仪(人数、人次、产生的赛尔豆)

StatLogger logger = new StatLogger(2,-1,-1,-1,-1);

StatInfo info = new StatInfo();

info.add\_info("赛尔豆", 100);

info.add\_op(OpCode.op\_sum, "赛尔豆","");

logger.log("游戏输出", "保护导航仪", "37896574", "", info);

op\_max:是对发过来的同一天或同一分钟的数据求最大值

op\_set:是对发过来的同一天或同一分钟的数据做set操作,即覆盖操作

1. OpCode.op\_item(key)

OpCode.op\_item\_sum(key1,key2)

OpCode.op\_item\_max(key1, key2)

OpCode.op\_item\_set(key1, key2)

赛尔号->触达率->太空站

 也可以如下方式：

StatLogger logger = new StatLogger(2,-1,-1,-1,-1);

StatInfo info = new StatInfo();

info.add\_info("item", "进入英佩恩堡垒");

info.add\_info("item", "进入英佩恩堡垒|场景");

info.add\_info("item", "进入英佩恩堡垒二层|场景");

info.add\_op(OpCode.op\_item, "item","");

logger.log("触达率", "太空站", "324234534", "", info);

1. OpCode.op\_sum\_distr(key) OpCode.op\_max\_distr(key) Statino::op\_min\_distr(key)

这一项也是用于统计区间分布的，只是这里的区间需要策划或相关开发人员去配置，比如统计今日产出的金豆的区间分布，例如

产出金豆数为1-100的人数

产出金豆数为101-500的人数

产出金豆数为501-1000的人数

产出金豆数大于1000的人数

那么就需要在用户产出金豆的时候，按照如下格式发送日志：

StatLogger logger = new StatLogger(1,-1,-1,-1,-1);

StatInfo info = new StatInfo();

info.add\_info("产出金豆", 56);

info.add\_op(OpCode.op\_sum\_distr, "产出金豆","");

logger.log("小游戏产出", "产出金豆", "34478323", "", info);就会将以用户为单位，计算每个人今日产出的金豆总量，然后看属于哪个区间，做分布

Op\_max\_distr和op\_min\_distr 类似,只是分别去每个用户的最大值或最小值而已.

1. OpCode.op\_set\_distr(key)

这一项用于做等级分布

比如iseer用户最高精灵等级分布

StatLogger logger = new StatLogger(78,-1,-1,-1,-1);

StatInfo info = new StatInfo();

info.add\_info("max\_level", 19);

info.add\_op(OpCode.op\_set\_distr, "max\_level","");

logger.log("等级分布", "用户最高精灵等级分布", "2341343", "", info);

1. OpCode.op\_ip\_distr(key)

地区分布统计各个地区的人数人次，精确到省份