淘米数据平台接入指南-4.1.1

淘米游戏-数据平台部

目录

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

1、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。

2、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。

3、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。

1、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

2、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。

3、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】

4、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。

2、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

0、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);

1、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.se
er.topbar”);//不填浏览器、设备等参数，则无法按此维度统计分析
```

2、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、	2

		分辨率、网络、网络服务提供商的人数人次	
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

3、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”);

4、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);

5、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

6、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);

7、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);

8、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**);

9、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);

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);

11、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);

12、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	46
donemaintsk（主线任务）		人次以及完成任务时的用	47
doneauxtsk（支线任务）		户等级分布	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);

13、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);

14、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");
```

15、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	21

		获得精灵的数量、各等级获得精灵的数量	

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");
```

16、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);

17、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”);

18、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");

19、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

（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

（4） 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

（5） 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

（6） 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。

（7） 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。

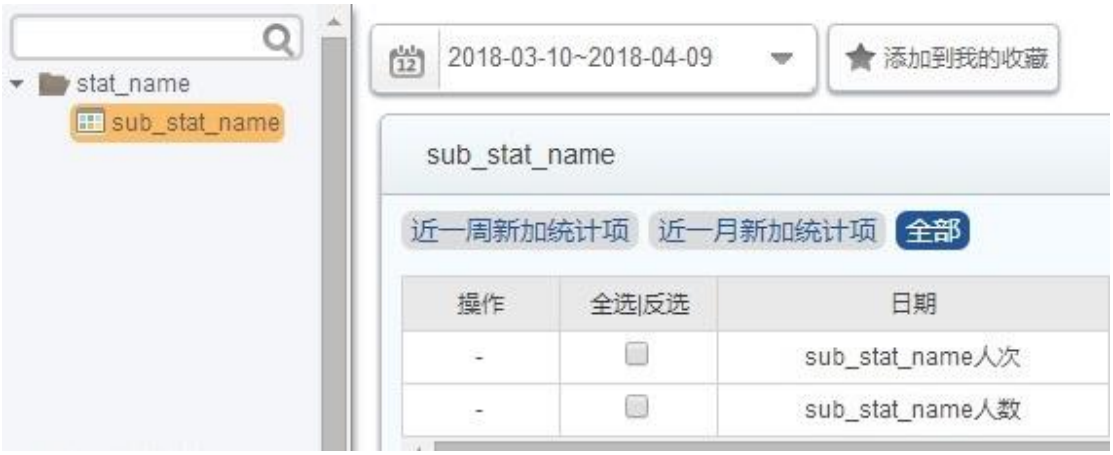
（8） 如需对添加的 key-value 对以及 op 操作符进行清空重置，调用 StatInfo 对象的 clear()

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);

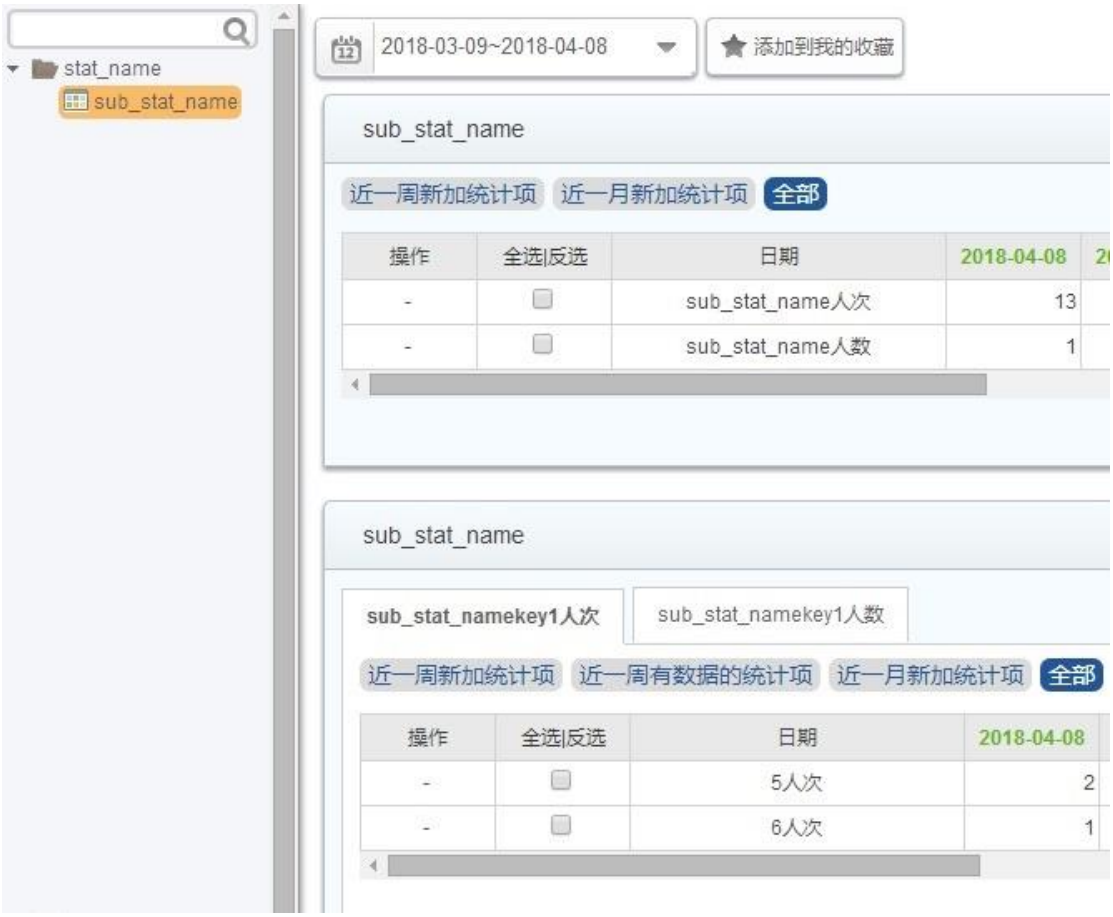
stat_name->sub_stat_name->sub_stat_namekey1 人数->人数(人数、人次、key1 人数)

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)



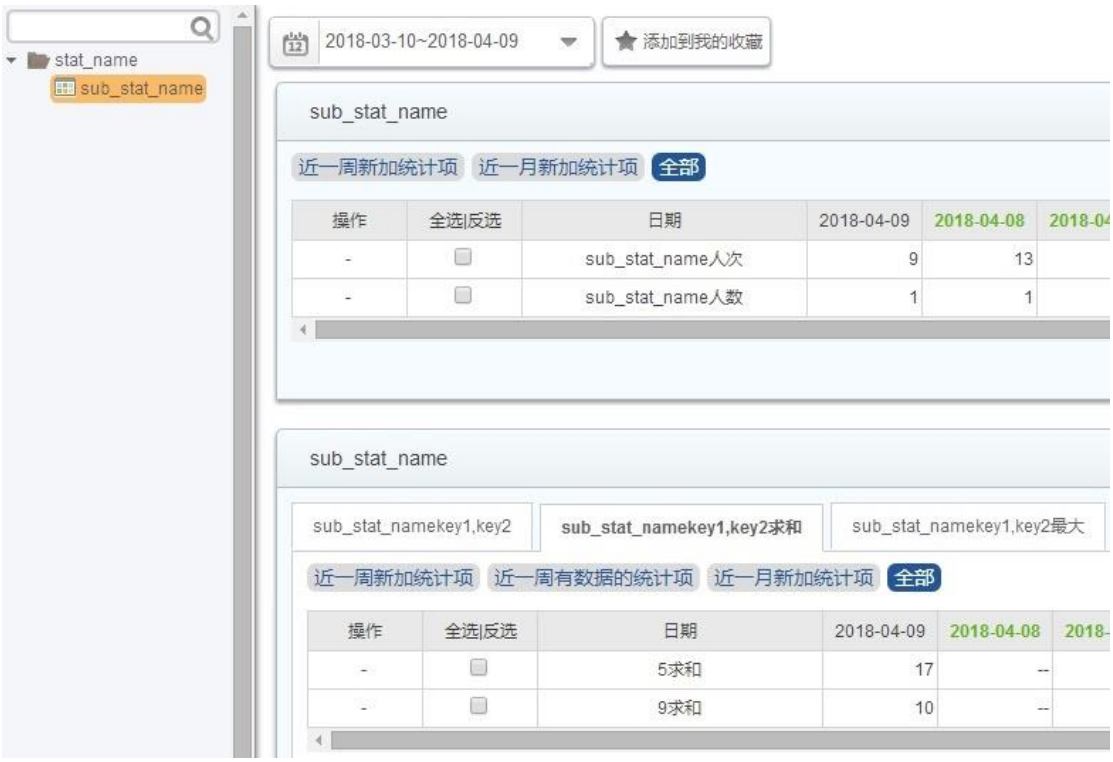
⑤ StatInfo::op_item_sum(key1,key2)/ StatInfo::op_item_max(key1,key2)/ StatInfo::op_item_set(key1,key2)

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）

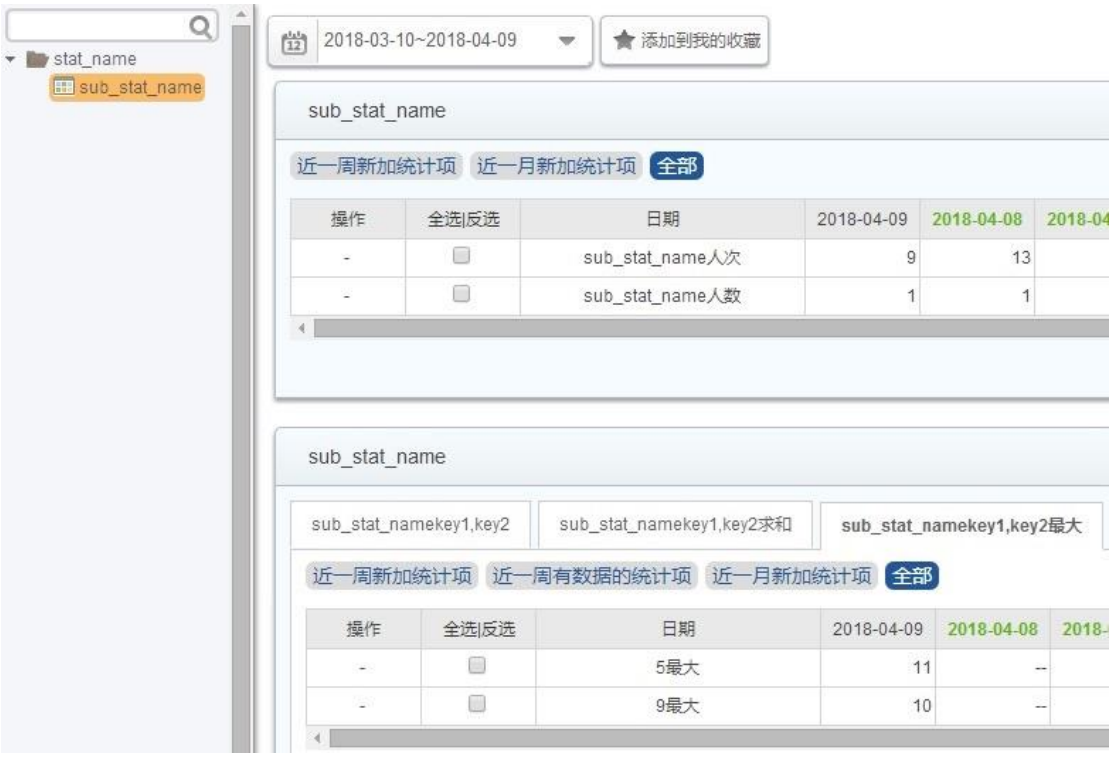


StatInfo::op_item_max(key1, key2)：

```
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）

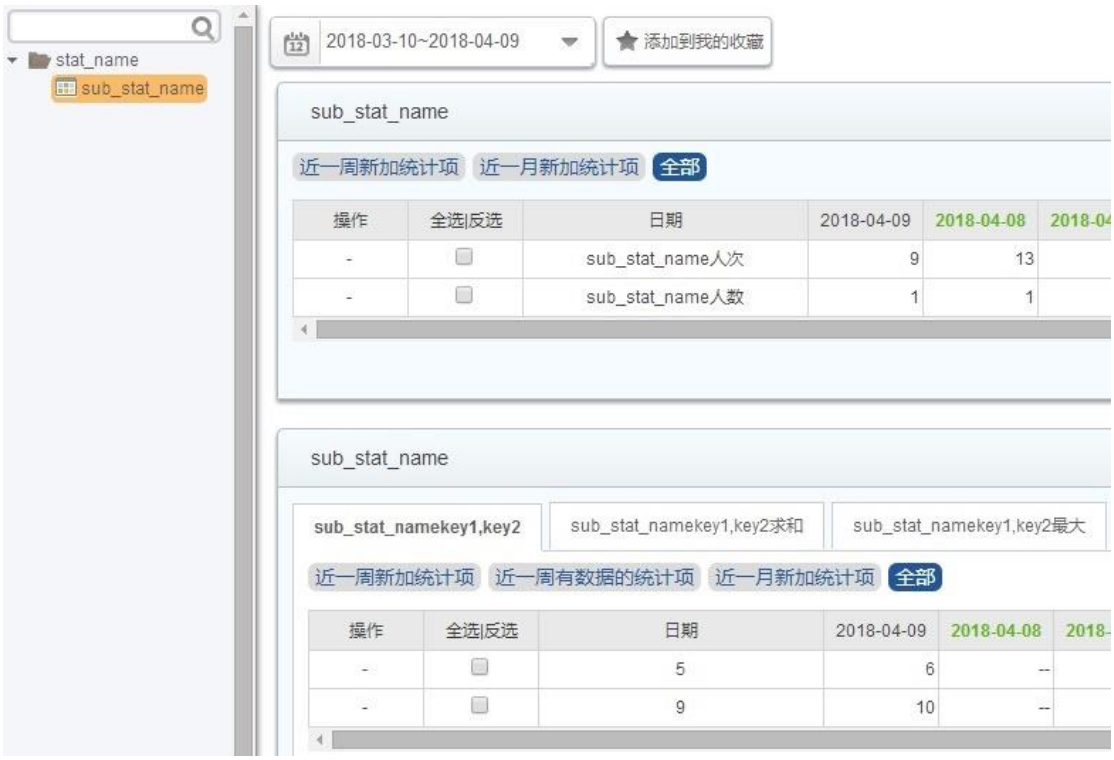


StatInfo::op_item_set(key1, key2)：

```
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）



⑥

StatInfo::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);

20、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);

2. 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。

0、初始化

```
/**
 * @brief StatLogger 构造函数
 *
 * @param game_id 应用ID 即游戏ID
 * @param zone_id 区ID
 * @param svr_id 服ID
 * @param site_id 平台ID 默认 -1表示该游戏不会拿出去放在不同的平台上运营  1：表示淘米平台
 * @param is_game 是否游戏后台络的数据
 */
public function __construct($game, $zone = -1, $svr = -1, $site = -1, $isgame = 1) {
```

1、注册角色

接口定义:

```
    // race: 职业  isp: 运营商
    /**
     * @brief 用户在游戏中创建角色时调用
     */
    public function reg_role($acct_id, $player_id = "", $race = "", $cli_ip = "", $ads_id = "", $browser = "", $device = "", $os = "", $resolution = "", $network = "", $isp = "") {
```

2、用户登录游戏(做活跃统计)

接口定义:

```
    /**
     * @brief 用户登录游戏时(登录online)调用
     * @param lv 取值范围1～5000
     * @param zone 为空表示总体，该字段主要时给公司老的单区服游戏使用，
用于区分电信登录和网通登录
     */
    public function login_online($acct_id, $player_id, $race, $isvip, $
lv, $cli_ip = "", $ads_id = "", $zone="", $browser = "", $device = "",
$os = "", $resolution = "", $network = "", $isp = "") {
```

3、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”**)

```
    /**
     * @brief 玩家每次在游戏内使用米币购买道具时调用
     * @param outcome 付费获得的道具名称。如果pay_reason选择的是pay_buy
，则outcome赋值为相应的“ 道具名字” ；
     *                如果pay_reason的值不是pay_buy，则本参数可以赋空字
符串。
     * @param pay_channel 支付类型，如米币卡、米币帐户、支付宝、苹果官>
方、网银、手机付费等等，米币渠道传"1"。
     */
    public function pay($acct_id, $isvip, $pay_amount, $currency, $pay_
reason, $outcome, $outcnt, $pay_channel) {
```

4、自定义统计项接口(只支持统计人数人次)

```
/**
 * @brief 自定义统计项
 * @param stat_name 主统计名称，不能以_开头或结束，StatLogger会自动把头尾的_去掉。
 * @param sub_stat_name 子主统计名称，一个主统计名称下可以有多个子统计项。
 *                      不能以_开头或结束，StatLogger会自动把头尾的_去掉。
 */
public function log($stat_name, $sub_stat_name, $acct_id, $player_id = "-1", $info = null)
```

五、web 端统计或 AS 统计

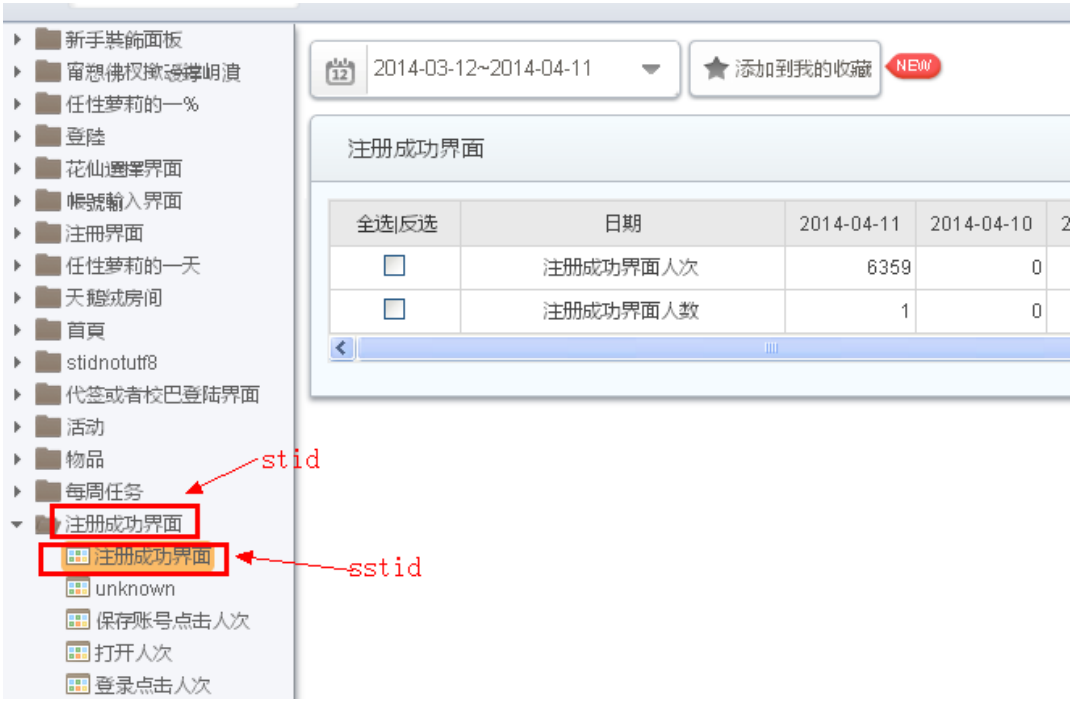
通过发送 http 请求的形式，支持人数人次的统计，提供二级和三级两种统计方式

http 请求地址：

二级统计项：

http://newmisc.taomee.com/misc.js?gameid=gameid&stid=stid&sstid=sstid&uid=u

id



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)

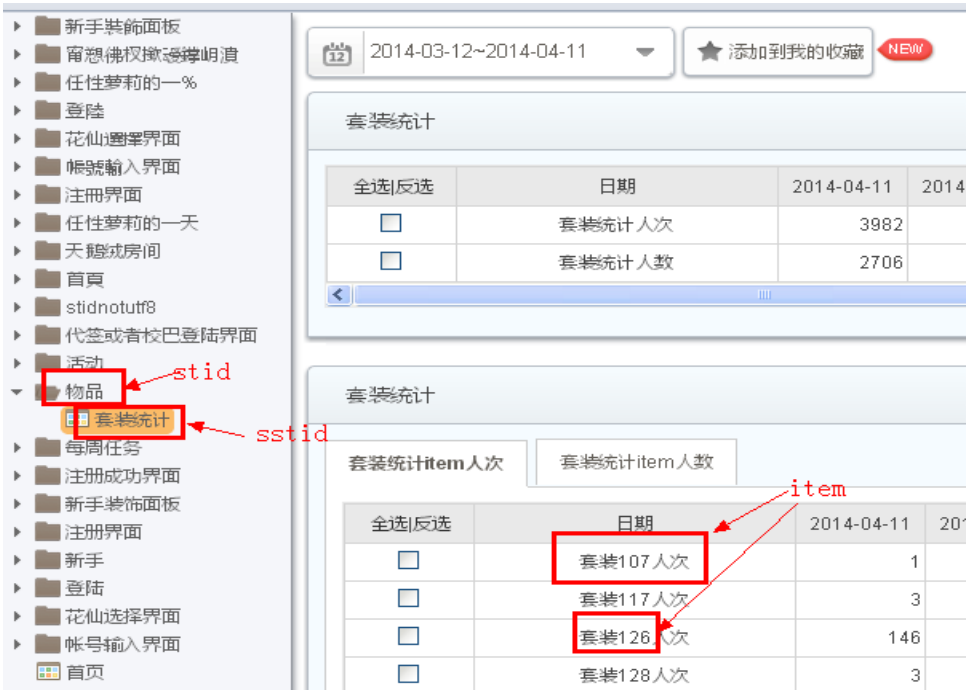
其中，stid、sstid 可填写中文、英文或数字字符串，但若不能避免乱码问题，请尽量选择填写英文或数字字符串

三级统计项：

http://newmisc.taomee.com/misc.js?gameid=gameid&stid=stid&sstid=sstid&uid=uid&item=item

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

0、初始化 sdk 接口

```
public StatLogger(int game, int zone, int svr, int site, int isgame)
```

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

```
static final StatLogger statLogger=new StatLogger(652, -1, -1, -1, 1);
```

1、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

```
public void verify_passwd(String acct_id,String cli_ip,String ads_id)
```

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.taome e.seer.topbar”);//不填浏览器、设备等参数，则无法按此维度统计分析

2、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

```
public void reg_role(String acct_id, String player_id, String race,
        String cli_ip, String ads_id) {
```

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")

3、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

```
public void login_online(String acct_id, String player_id, String race,
        boolean isvip, Integer lv, String cli_ip, String ads_id, String zone)
```

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

```
statLogger.login_online("331025680", "-1", "-1", false, 1, "192.168.11.120", "-1", "");
```

4、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

```
public void logout(String acct_id, boolean isvip, String lv, Integer oltime)
```

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

```
statLogger.logout("331025680",false, "5", 300);
```

5、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

```
public void online_count(int cnt, String zone)
```

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,"");

6、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

```
public void level_up(String acct_id,String race,int lv)
```

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);

7、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

```
public void pay(String acct_id, boolean isvip, int pay_amount,
        CurrencyType currency, PayReason pay_reason, String outcome,
        int outcnt, String pay_channel) {
```

参数说明

<table>
<tr><th>参数名</th><th>类型</th><th>备注</th></tr>
<tr><td>acct_id</td><td>String</td><td>用户账户(米米号)</td></tr>
<tr><td>isvip</td><td>boolean</td><td>是否 VIP 用户(true:是，false:否)</td></tr>
<tr><td>pay_amount</td><td>int</td><td>付费额度,单位统一为”分”，例如花 10 米币，数量落 1000</td></tr>
<tr><td>currency</td><td>CurrencyType</td><td>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）</td></tr>
<tr><td>pay_reason</td><td>PayReason</td><td>支付类型<table><tr><th>类型</th><th>落的内容</th></tr><tr><td>兑换游戏币</td><td>PayReason.pay_buy</td></tr><tr><td>VIP 包月</td><td>PayReason.pay_vip</td></tr></table></td></tr>
<tr><td>outcome</td><td>String</td><td>支付产生的物品（不能为空！）<table><tr><th>类型</th><th>落的内容</th></tr><tr><td>兑换游戏币</td><td>商品 id，由账户支付平台 boss 分配</td></tr><tr><td>VIP 包月</td><td>“1 个月 VIP”、“3 个月 VIP”等</td></tr></table></td></tr>
<tr><td>outcnt</td><td>int</td><td>获得数量</td></tr>
</table>

		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");

8、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

```
public void obtain_golds(String acct_id,int amt)
```

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);

9、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

```
public void buy_item(String acct_id, boolean isvip, int lv, int pay_amount,
        String outcome, int outcnt) {
```

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);

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

```
public void use_golds(String acct_id, boolean is_vip, String reason,
        float amt, int lv) {
```

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);

11、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

```
public void accept_task(TaskType type, String acct_id, String task_name, int lv)
```

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);

12、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）	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

doneetctsk（其它任务）		

接口定义

```
public void finish_task(TaskType type, String acct_id, String task_name, int lv)
```

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);

13、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

```
public void abort_task(TaskType type, String acct_id, String task_name, int lv)
```

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);

14、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

```
public void obtain_spirit(String acct_id,boolean isvip,int lv ,String spirit)
```

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");

15、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

```
public void lose_spirit(String acct_id,boolean isvip,int lv,String spirit)
```

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");

16、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

```
public void unsubscribe(String acct_id,UnsubscribeChannel uc)
```

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);
```

17、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

```
public void cancel_acct(String acct_id,String channel)
```

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”);

18、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

```
public void new_trans(NewTransStep step,String acct_id)
```

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”);

19、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

（2） 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。

（3） 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

（4） 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

（5） 对 value 的取值不能有“=|”、空格、制表符中的任何一个

（6） 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

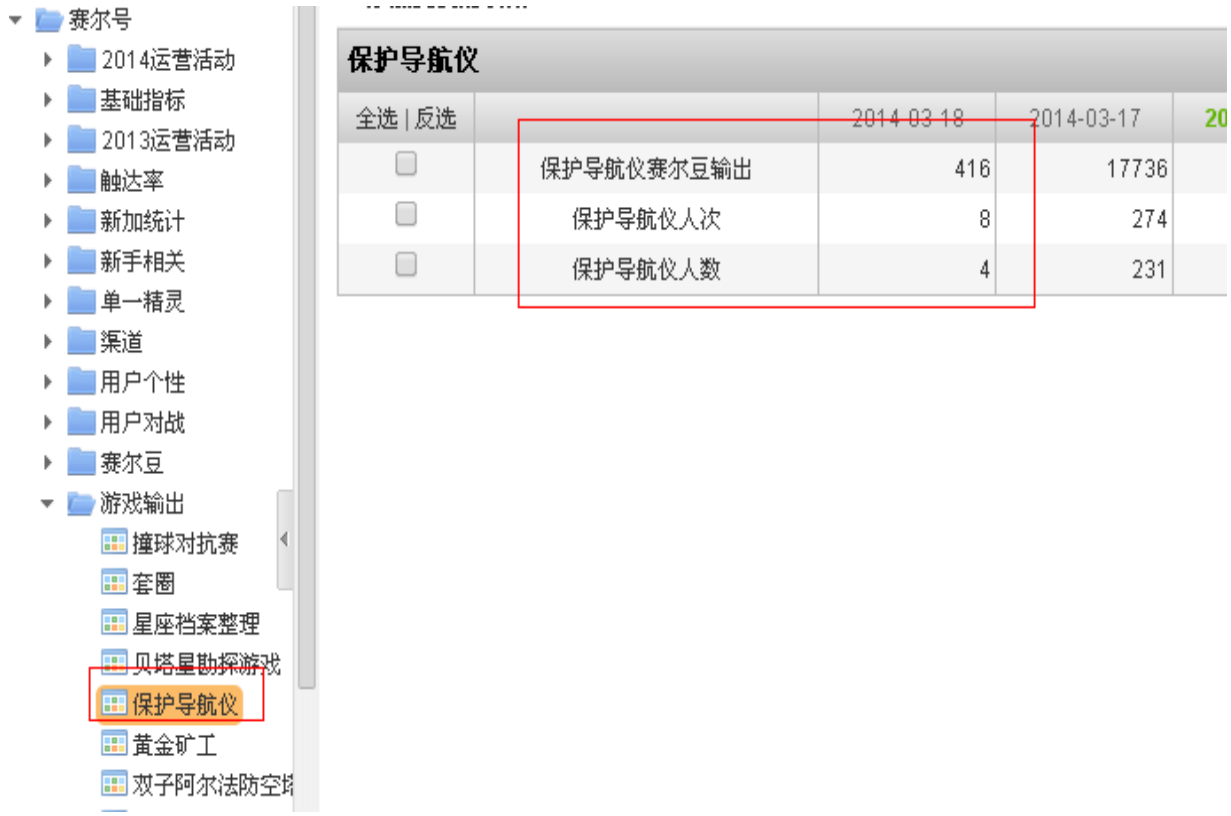
赛尔号游戏 ID 为 2

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

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

2、 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

3、 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);
```

4、 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.

5、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);

6、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