**掌趣统计日志**

**服务端接入文档（C++版本）**

# 修订记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **修改说明** | **作者** |
| 2017-06-05 | 1.0 |  | 张宇 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[修订记录 1](#_Toc25865)

[1. 服务端日志接入流程 3](#_Toc21413)

[2. 客户端接口 3](#_Toc26424)

[2.1. IOS客户端接口介绍 3](#_Toc19192)

[2.2. Android客户端接口介绍 3](#_Toc20029)

[3. 服务端接入工作 3](#_Toc23318)

[3.1. 服务端接口介绍 3](#_Toc18948)

[需要注意的地方： 5](#_Toc28797)

[3.2 头文件引入说明 5](#_Toc18486)

[配置信息说明 5](#_Toc12498)

[3.日志接入详细介绍 6](#_Toc29716)

[玩家角色注册日志（10006） 6](#_Toc3516)

[玩家角色登陆日志(10007) 7](#_Toc18034)

[玩家充值日志(10008) 8](#_Toc15221)

[玩家虚拟货币变更日志(10091) 9](#_Toc11884)

[玩家虚拟物品变更日志(10009) 11](#_Toc4405)

[玩家属性变更日志(10010)单属性变动 12](#_Toc24766)

[玩家属性变更日志(10010)多属性变动 13](#_Toc18850)

[玩家物品属性变更日志(10013) 15](#_Toc2797)

[玩家卡牌属性变更日志(10014) 16](#_Toc3033)

[任务，关卡场景等事件日志说明 17](#_Toc20639)

[玩家任务日志(11101) 18](#_Toc10719)

[玩家关卡场景日志(11102) 20](#_Toc27863)

[玩家自定义行为日志(11103) 21](#_Toc10103)

[玩家交互日志(12001) 22](#_Toc24049)

[关卡场景事件日志综合使用举例 23](#_Toc29862)

[服务端异常日志(10011) 27](#_Toc31373)

[自定义日志(11000) 28](#_Toc24951)

[附1.1货币规范 29](#_Toc32462)

# 服务端日志接入流程

服务端日志接入流程大致分为四个步骤：

1. 客户端接入负责人接入掌趣SDK，使用其中的API获取ServiceCode
2. 服务端接入负责人根据日志接入文档，调用API，完成日志的接入
3. 掌趣科技数据中心对服务端接入的日志进行格式和字段内容检查
4. 服务端接入负责人与掌趣技术人员通过日志接入测试后，日志接入工作正式完成

# 客户端接口

## IOS客户端接口介绍

调用示例：

std::string gameResVersion("1.0"); //游戏资源版本号

OPGameKit::GetInstance().GetServiceCode(gameResVersion);

## Android客户端接口介绍

接口定义：

/\*\*

\* 获取ServiceCode

\*

\* @param gameResVer

\* 游戏资源版本号,无资源版本号可传""

\* @return String

\*/

public String Ourpalm\_getServiceCode(final String gameResVer);

调用示例：

// 获取掌趣serviceCode接口，游戏需要调用(SDK接口)

Ourpalm\_ServiceCode\_Entry.getInstance(this).Ourpalm\_getServiceCode("");

# 服务端接入工作

## 服务端接口介绍

服务端日志接口，用于采集诸如玩家注册、登录，玩家虚拟物品变更、属性变更，玩家游戏内的任务、关卡场景等用户行为，为产品运营，数据分析，推广营销等工作提供数据支持。

在调用日志接口时，需要为日志接口提供客户端配置信息，设备信息，用户信息，角色信息，日志内容等参数，最终组装成为一个json对象，作为http请求的content body发送到数据中心日志接口

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 字段名 | 字段类型 | 字段样例 | 字段含义 | 备注 |
| Log对象 | id | String | 10006 | 日志id |  |
| serviceCode | String |  | 封装设备信息的字符串 |  |
| uid | String | 0100010000000000000000000000000090019780 | 用户中心UID |  |
| clientIp | String | 1.202.218.162 | 游戏客户端ip地址 |  |
| server | String | 1 | 逻辑服 |  |
| roleId | String | 671092810 | 角色ID |  |
| roleName | String | 马克.吐温 | 角色名 |  |
| roleLevel | String | 10 | 角色等级 |  |
| roleVipLevel | String | 1 | 角色VIP等级 |  |
| customId | String | lz\_rolelogin | 自定义日志id（如果该日志不是自定义日志，则该字段为空） |  |
| roleSpecAttr | Array |  | 用于封装角色特殊属性的一个Map |  |
| logInfo | Array |  | 用于封装具体日志信息的一个Map |  |
| createLogTime | String |  | 记录游戏日志生成的时间戳，因为游戏采用异步进程的方式发送日志，日志发送的时间，不一定为日志生成的时间，所以要记录日志生成的时间 |  |

### 

### 需要注意的地方：

编码：当传入中文字符串时，务必确保该中文字符串使用的是utf-8编码。例如，商品名称。

配置一致性：有的产品人员和开发人员自行设定的字段信息，也必须由相应的负责人确保内容的一致性。 例如同一个逻辑服下，如果在用户登录注册时调用日志接口传递的逻辑服标识是1001，那么在调用计费中心的充值计费接口时逻辑服标识也应该是同样的1001，而不应该是其他的值。

ID一致性：同样的，如果A用户在购买一个极品武器时记录的物品id是001weapon，那么另外一个用户在拍卖同样一个极品武器时物品id也应该是001weapon。

接入日志类型的要求：目前UC平台的游戏都需要接入10006注册日志、10007登录日志和10010角色变更日志，其它的日志类型请根据游戏各自需要接入。

运行日志：如果需要打印运行日志，需要调用OurpalmLog::PalmLogSender类的setDebugMode(Boolean mode)方法，并传入参数true

## 3.2 头文件引入说明

为了便于研发人员接入服务端日志接口，我们把所有用到的类声明都放到了logsender.h这个头文件中，只需要#include 这一个文件即可，这个文件的命名空间为OurpalmLog

通常情况下，使用我们提供的OurpalmLog::{Log,LogRolePropUpdate,PalmLogSender}类就足够完成日志发送工作，如果研发人员认为有必要自行实现日志发送功能，请先与我们取得联系，避免沟通不到位引发接入过程中的各种问题。

下面给出示例：

#include "logsender.h"

using namespace std;

using OurpalmLog::Log;

using OurpalmLog::LogRolePropUpdate;

using OurpalmLog::PalmLogSender;

### 配置信息说明

在logsender.h同目录下的config.ini文件里有一些配置项需要研发人员根据实际情况手动修改，一般不许变动此文件的配置信息，我们将配置项对应的意义列举如下

|  |  |  |  |
| --- | --- | --- | --- |
| **属性名** | **属性值示例** | **说明** | **备注** |
| log.sendurl | http://bicenter.gamebean.net/bicenter/prismClient/sendLogV1.do | 日志发送地址 |  |
| log.errorlogpath | /opt/ourpalm/logs/prismClient/fail | 发送失败的日志保存目录 | 请确保这个目录有写入权限！！ |
| log.size | 300 | 单次批量发送日志的最大数量 |  |

# 3.日志接入详细介绍

目前日志接口支持以下日志的接入

|  |  |
| --- | --- |
| **日志ID** | **日志名** |
| 10006 | 服务端玩家注册日志 |
| 10007 | 服务端玩家登录日志 |
| 10008 | 服务端玩家充值日志 |
| 10091 | 服务端玩家货币变更日志 |
| 10009 | 服务端玩家物品变更日志 |
| 10010 | 服务端玩家属性变更日志 |
| 10011 | 服务端异常日志 |
| 10013 | 服务端玩家商品属性变更日志 |
| 10014 | 服务端玩家卡牌属性变更日志 |
| 11000 | 服务端自定义日志 |
| 11101 | 服务端玩家任务日志 |
| 11102 | 服务端玩家副本关卡场景日志 |
| 11103 | 服务端玩家自定义行为日志 |
| 12001 | 服务端玩家自定义交互日志 |

## 玩家角色注册日志（10006）

当玩家注册一个游戏内的新角色时，记录一条角色注册日志。

***调用示例:***

try{

string serviceCode = “从SDKAPI中获取的serviceCode”;

std::map<string, string> map;

Log log;

log.setId("10006");

log.putServer("1");

log.putRoleId("11111111");

log.putRoleName("马克.吐温");

log.putRoleLevel("1");

log.putRoleVipLevel("0");

log.setServiceCode(serviceCode);

log.putUserId("0100010000000000000000000000000090019780");

log.setClientIp("111.111.111.111");

log.setCreateLogTime(time(NULL) \* 1000);

log.setLogInfo(map);

vector<Log> logArr;

logArr.push\_back(log);

PalmLogSender::sendLogList(logArr);

} catch(exception &e) {

cout << "catch exception:" << e.what() << endl;

}

该记录表示玩家马克吐温已经在逻辑服1注册完成了

## 玩家角色登陆日志(10007)

当玩家的角色登录游戏时，记录一条角色登录日志。

***调用示例:***

try{

string serviceCode = “从SDKAPI中获取的serviceCode”;

std::map<string, string> map;

Log log;

log.setId("10007");

log.putServer("1");

log.putRoleId("11111111");

log.putRoleName("马克.吐温");

log.putRoleLevel("1");

log.putRoleVipLevel("0");

log.setServiceCode(serviceCode);

log.putUserId("0100010000000000000000000000000090019780");

log.setClientIp("111.111.111.111");

log.setCreateLogTime(time(NULL) \* 1000);

log.setLogInfo(map);

vector<Log> logArr;

logArr.push\_back(log);

PalmLogSender::sendLogList(logArr);

}catch(exception &e){

cout << "catch exception:" << e.what() << endl;

}

该记录表示玩家马克吐温已经登录逻辑服1了

## 玩家充值日志(10008)

当用户已经启动游戏客户端，且通过发货通知接口收到发货通知并给用户成功发货时，需要记录一条对应的发货成功日志来跟踪玩家充值。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **LogId** | **日志名称** | **字段名** | **字段类型** | **字段说明** | **备注** |
| 10008 | 玩家充值日志 | payChannelId | String | 支付方式 | 例如，91支付，UC支付等，非必需 |
| orderNo | String | 订单号 | 必需 |
| currencyType | String | 货币类型 | （参照附1.1货币规范）必需 |
| currencyAmount | Integer | 订单金额 | 单位:分，必需 |
| itemId | String | 商品ID | 游戏中商品或者道具ID，必需 |
| itemName | String | 商品名称 | 便于运营人员识别商品，一般传入中文，必需 |
| itemCount | Integer | 商品个数 | 必需 |

**支付方式ID**：如支付宝，神州付，财付通，微信支付等，由开发者自行指定，需要在同一个产品的所有订单中保持支付方式id的一致性

**订单号**：由开发者自行指定该笔充值的唯一标识

**订单金额**：即发货通知接口的订单金额

**货币类型**：玩家支付的是人民币还是美元或者欧元，见附1.1货币规范

**商品ID**：用于记录本次充值给用户发放了哪一个商品，商品id建议由英文字母或数字组成，以避免乱码等问题。

**商品名称**：商品名称建议为中文信息。

日志时间字段，sdk会根据调用日志接口时间自行填充。

***调用示例:***

try{

string serviceCode = “从SDKAPI中获取的serviceCode”;

std::map<string, string> map;

map.insert(pair<string, string>("payChannelId", "3100052033100012"));

map.insert(pair<string, string>("orderNo", "0012014090110143611500"));

map.insert(pair<string, string>("currencyType", "1"));

map.insert(pair<string, string>("currencyAmount", "100"));

map.insert(pair<string, string>("itemId", "diamond"));

map.insert(pair<string, string>("itemName", "钻石"));

map.insert(pair<string, string>("itemCount", "100"));

Log log;

log.setId("10008");

log.putServer("1");

log.putRoleId("11111111");

log.putRoleName("马克.吐温");

log.putRoleLevel("1");

log.putRoleVipLevel("0");

log.setServiceCode(serviceCode);

log.putUserId("0100010000000000000000000000000090019780");

log.setClientIp("111.111.111.111");

log.setCreateLogTime(time(NULL) \* 1000);

log.setLogInfo(map);

vector<Log> logArr;

logArr.push\_back(log);

PalmLogSender::sendLogList(logArr);

} catch(exception &e) {

cout << "catch exception:" << e.what() << endl;

}

该记录表示当前登录的角色马克.吐温充值100分人民币购买游戏内100钻石

## 玩家虚拟货币变更日志(10091)

当用户在游戏内，因为种种原因产生虚拟货币的增加和消耗时，需要记录一条对应的虚拟货币变更日志。如果玩家的一次用户行为导致多种虚拟货币变更，比如完成一次充值同时获得1000钻石20000铜币，则需要记录相应的两条虚拟货币变更日志。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **LogId** | **日志名称** | **字段名** | **字段类型** | **字段说明** | **备注** |
| 10091 | 玩家虚拟货币变更日志 | updateType | String | 变更类型 | 1：新增  0：消耗 |
| itemId | String | 货币ID |  |
| itemName | String | 货币名称 | 便于运营人员识别商品，一般传入中文 |
| isPrecious | String | 是否为珍贵物品 | 1：是  0：否 |
| itemCount | Integer | 货币个数 | 必须是一个大于0的整数 |
| remains | Integer | 变化后货币个数 | 必须是一个大于等于0的整数 |
| custom | String | 自定义 |  |

**变更类型**：用于标识本次变更是新增了虚拟货币还是消耗了虚拟货币，新增记录1，消耗记录0。

**货币ID**：用于记录本次引发虚拟货币变更的货币ID。

**货币名称**：用于记录本次引发虚拟货币变更的货币名称。

**是否为珍贵物品**：用于区分珍贵物品，一般情况下直接充值获得的即为珍贵物品，例如 钻石和铜币， 钻石可视为珍贵物品，铜币可视为非珍贵物品。

**货币个数**：用于记录用户本次消费行为涉及多少个货币，如果用户本次消费行为与具体货币无关，则该字段置空即可。

**变化后货币个数**：用于记录用户本次消费行为完成后总的货币剩余量。

**自定义变更描述**：用于标识用户因为什么原因变更虚拟货币。

***调用示例:***

try{

string serviceCode = “从SDKAPI中获取的serviceCode”;

std::map<string, string> map;

map.insert(pair<string, string>("updateType", "0"));

map.insert(pair<string, string>("itemId", "diamond"));

map.insert(pair<string, string>("itemName", "钻石"));

map.insert(pair<string, string>("isPrecious", "1"));

map.insert(pair<string, string>("itemCount", "100"));

map.insert(pair<string, string>("remains", "200"));

map.insert(pair<string, string>("custom", "购买道具"));

Log log;

log.setId("10091");

log.putServer("1");

log.putRoleId("11111111");

log.putRoleName("马克.吐温");

log.putRoleLevel("1");

log.putRoleVipLevel("0");

log.setServiceCode(serviceCode);

log.putUserId("0100010000000000000000000000000090019780");

log.setClientIp("111.111.111.111");

log.setCreateLogTime(time(NULL) \* 1000);

log.setLogInfo(map);

vector<Log> logArr;

logArr.push\_back(log);

PalmLogSender::sendLogList(logArr);

} catch(exception &e) {

cout << "catch exception:" << e.what() << endl;

}

这条日志表示当前登录的角色马克.吐温为了购买道具花了100钻石，剩余200钻石。

## 玩家虚拟物品变更日志(10009)

当用户在游戏内，因为种种原因产生虚拟物品的增加和消耗时，需要记录一条对应的虚拟物品变更日志。如果玩家的一次用户行为导致多种虚拟物品变更，比如完成一次史诗副本同时获得10个物品A和20个物品B，则需要记录相应的两条日志。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **LogId** | **日志名称** | **字段名** | **字段类型** | **字段说明** | **备注** |
| 10009 | 玩家虚拟物品变更日志 | updateType | String | 变更类型 | 1：新增  0：消耗 |
| itemId | String | 商品ID |  |
| itemName | String | 商品名称 | 便于运营人员识别商品，一般传入中文 |
| itemCount | Integer | 商品个数 | 必须是一个大于0的整数 |
| custom | String | 自定义 |  |

**变更类型**：用于标识本次变更是新增了虚拟物品还是消耗了虚拟物品，新增记录1，消耗记录0。

**商品ID**：用于记录用户本次是因为哪一个游戏内的商品引发的虚拟物品变更，例如购买游戏内的装备、经验、头衔、坐骑等。商品名称由游戏策划人员自行定义，比如 0001对应装备图纸，0002对应黄金翅膀，0003对应经验丹，0004对应体力等等。传入0001,0002,0003……这些ID即可

**商品名称**：用于记录用户本次是因为哪一个游戏内的商品引发的虚拟物品变更，例如购买游戏内的装备、经验、头衔、坐骑等。商品名称由游戏策划人员自行定义，比如 0001\_装备图纸，0002\_黄金翅膀，0003\_经验丹，0004\_体力等等。统计结果也会直接将日志中的商品名称展示出来，如果用户本次变更行为与具体商品道具无关，则该字段置空即可。

**商品个数**：用于记录用户本次消费行为涉及多少个商品，如果用户本次消费行为与具体商品无关，则该字段置空即可。

**自定义变更描述**：用于标识用户因为什么原因变更虚拟物品，如任务获得，怪物掉落，系统赠送，副本奖励，世界喊话，家族奉献，婚姻喜宴等等。

***调用示例:***

try{

string serviceCode = “从SDKAPI中获取的serviceCode”;

std::map<string, string> map;

map.insert(pair<string, string>("updateType", "1"));

map.insert(pair<string, string>("itemId", "hammer"));

map.insert(pair<string, string>("itemName", "铁锤"));

map.insert(pair<string, string>("itemCount", "1"));

map.insert(pair<string, string>("custom", "购买道具"));

Log log;

log.setId("10009");

log.putServer("1");

log.putRoleId("11111111");

log.putRoleName("马克.吐温");

log.putRoleLevel("1");

log.putRoleVipLevel("0");

log.setServiceCode(serviceCode);

log.putUserId("0100010000000000000000000000000090019780");

log.setClientIp("111.111.111.111");

log.setCreateLogTime(time(NULL) \* 1000);

log.setLogInfo(map);

vector<Log> logArr;

logArr.push\_back(log);

PalmLogSender::sendLogList(logArr);

} catch(exception &e) {

cout << "catch exception:" << e.what() << endl;

}

上面两条日志表示当前登录的角色马克.吐温购买了1把铁锤 。

## 玩家属性变更日志(10010)单属性变动

当用户的角色，发生一些属性上的变化时，记录相应的角色属性变更日志。例如等级提升，军衔提升，VIP等级提升，属性点重配，种族天赋变更等等。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **LogId** | **日志名称** | **字段名** | **字段类型** | **字段说明** | **备注** |
| 10010 | 玩家属性变更日志 | propKey | String | 属性标识 |  |
| propValue | String | 新属性值 |  |
| rangeability | String | 变更值 |  |

**属性标识**：角色变更了哪种属性，其中等级标识为**level**，VIP等级标识为**vipLevel**，其余属性标识由游戏产品和开发人员自行设定。请注意标识大小写。

**新属性值**：角色完成属性变更之后，该属性的值变成了什么。

**变更值**：如果该属性值是一个数字，比如等级level，那么本次用户的等级变更了多少，例如购买大礼包level连升三级，则变更值为3。

***调用示例:***

try{

string serviceCode = “从SDKAPI中获取的serviceCode”;

std::map<string, string> map;

map.insert(pair<string, string>("propKey", "level"));

map.insert(pair<string, string>("propValue", "1"));

map.insert(pair<string, string>("rangeability", "1"));

Log log;

log.setId("10010");

log.putServer("1");

log.putRoleId("11111111");

log.putRoleName("马克.吐温");

log.putRoleLevel("1");

log.putRoleVipLevel("0");

log.setServiceCode(serviceCode);

log.putUserId("0100010000000000000000000000000090019780");

log.setClientIp("111.111.111.111");

log.setCreateLogTime(time(NULL) \* 1000);

log.setLogInfo(map);

vector<Log> logArr;

logArr.push\_back(log);

PalmLogSender::sendLogList(logArr);

} catch(exception &e) {

cout << "catch exception:" << e.what() << endl;

}

这两条日志表示当前角色马克.吐温等级升了一级，目前等级为1。

## 玩家属性变更日志(10010)多属性变动

当用户的角色，发生一些属性上的变化时，记录相应的角色属性变更日志。例如等级提升，军衔提升，VIP等级提升，属性点重配，种族天赋变更等等。

为了满足同一个玩家同时有多个属性变化的游戏场景，我们提供第二种玩家属性变更日志的接入方式，同时传入玩家变更前和变更后的属性Map，由数据中心判断玩家有哪些属性发生了变化

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **LogId** | **日志名称** | **字段名** | **字段类型** | **字段说明** | **备注** |
| 10010 | 玩家属性变更日志 | propMapOld | Map | 旧属性集合 |  |
| propMapNew | Map | 新属性集合 |  |

**旧属性集合**：玩家的角色在变更之前有哪些属性值，用Map<String, String>的对象来保存，用属标识作为key，属性值作为val。

**新属性集合**：玩家的角色在变更之前有哪些属性值，用Map<String, String>的对象来保存，用属标识作为key，属性值作为val。

注意，只有数值类型的属性，才应该通过这个方法来发送属性变更日志，如果是一些文本类型的属性，比如玩家更改了角色名称，服务器id，种族天赋类型等，则不应该使用这个方法来保存

***调用示例:***

try{

string serviceCode = “从SDKAPI中获取的serviceCode”;

std::map<string, string> propMapOld;

propMapOld.insert(pair<string, string>("level", "3"));

propMapOld.insert(pair<string, string>("vipLevel", "1"));

propMapOld.insert(pair<string, string>("hp", "1000"));

propMapOld.insert(pair<string, string>("mp", "1000"));

std::map<string, string> propMapNew;

propMapNew.insert(pair<string, string>("level", "4"));

propMapNew.insert(pair<string, string>("vipLevel", "1"));

propMapNew.insert(pair<string, string>("hp", "1500"));

propMapNew.insert(pair<string, string>("mp", "1500"));

LogRolePropUpdate log;

log.setId("10010");

log.putServer("1");

log.putRoleId("11111111");

log.putRoleName("马克.吐温");

log.putRoleLevel("1");

log.putRoleVipLevel("0");

log.setServiceCode(serviceCode);

log.putUserId("0100010000000000000000000000000090019780");

log.setClientIp("112.23.22.22");

log.setCreateLogTime(time(NULL) \* 1000);

log.setPropMapOld(propMapOld);

log.setPropMapNew(propMapNew);

vector<LogRolePropUpdate> logArr;

logArr.push\_back(log);

PalmLogSender::sendLogList(logArr);

} catch(exception &e) {

cout << "catch exception:" << e.what() << endl;

}

这条日志表示当前角色马克.吐温等级升了1级，目前等级为4。他的hp从1000增加到了1500，mp也从1000增加到了1500。三个属性发生了变化，vip等级不变

## 玩家物品属性变更日志(10013)

某些游戏的游戏内商品也有各种属性，比如射击类有戏的枪械会因为改装增加各种新功能，赛车游戏的赛车可以更改引擎、喷漆、轮胎，rpg游戏的玩家会饲养和升级宠物，升级随从的等级和装备等等，因此我们提供了一个游戏内商品属性变更日志，用于记录游戏内各种物品，道具，宠物，随从，武器的等级、技能等属性变更情况。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **LogId** | **日志名称** | **字段名** | **字段类型** | **字段说明** | **备注** |
| 10013 | 玩家商品属性变更日志 | itemId | String | 商品ID | 必需 |
| itemName | String | 商品名称 | 便于运营人员识别商品，一般传入中文，必需 |
| propKey | String | 属性标识 | 必需 |
| propValue | String | 新属性值 | 必需 |
| rangeability | String | 变更值 | 必需 |

玩家商品属性变更日志的格式与玩家属性变更日志的区别在于，商品属性变更日志中多了两个用于描述商品的itemId和itemName字段。

***调用示例:***

try{

string serviceCode = “从SDKAPI中获取的serviceCode”;

std::map<string, string> map;

map.insert(pair<string, string>("itemId", "MGH\_Weapon\_001"));

map.insert(pair<string, string>("itemName", "玛格汉铁锤"));

map.insert(pair<string, string>("propKey", "level"));

map.insert(pair<string, string>("propValue", "2"));

map.insert(pair<string, string>("rangeability", "1"));

Log log;

log.setId("10013");

log.putServer("1");

log.putRoleId("11111111");

log.putRoleName("马克.吐温");

log.putRoleLevel("1");

log.putRoleVipLevel("0");

log.setServiceCode(serviceCode);

log.putUserId("0100010000000000000000000000000090019780");

log.setClientIp("111.111.111.111");

log.setCreateLogTime(time(NULL) \* 1000);

log.setLogInfo(map);

vector<Log> logArr;

logArr.push\_back(log);

PalmLogSender::sendLogList(logArr);

} catch(exception &e) {

cout << "catch exception:" << e.what() << endl;

}

该条日志记录，玩家的玛格汉铁锤等级提升1级，现在的等级为2级

## 玩家卡牌属性变更日志(10014)

对于卡牌类游戏而言，卡牌的收集和升级一直是游戏重点，比如卡牌的稀有程度，等级变化，技能等级等等，因此我们提供了一个游戏内卡牌属性变更日志，用于记录游戏内对应英雄、侠客、随扈等等卡牌的属性变更情况。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **LogId** | **日志名称** | **字段名** | **字段类型** | **字段说明** | **备注** |
| 10014 | 玩家卡牌属性变更日志 | cardId | String | 卡牌ID | 必需 |
| cardName | String | 卡牌名称 | 便于运营人员识别商品，一般传入中文，必需 |
| propKey | String | 属性标识 | 必需 |
| newValue | String | 新属性值 | 必需 |
| oldValue | String | 旧属性值 | 必需 |

玩家卡牌属性变更日志的格式与玩家商品属性变更日志的区别在于，商品属性变更日志中记录的是游戏内商品道具的属性变更，比如武器装备升级之类，而卡牌属性变更则适用于卡牌类游戏。

***调用示例:***

try{

string serviceCode = “从SDKAPI中获取的serviceCode”;

std::map<string, string> map;

map.insert(pair<string, string>("cardId", "blr\_man\_001"));

map.insert(pair<string, string>("cardName", "李星云"));

map.insert(pair<string, string>("propKey", "level"));

map.insert(pair<string, string>("newValue", "蓝+1"));

map.insert(pair<string, string>("oldValue", "绿"));

Log log;

log.setId("10014");

log.putServer("1");

log.putRoleId("11111111");

log.putRoleName("马克.吐温");

log.putRoleLevel("1");

log.putRoleVipLevel("0");

log.setServiceCode(serviceCode);

log.putUserId("0100010000000000000000000000000090019780");

log.setClientIp("111.111.111.111");

log.setCreateLogTime(time(NULL) \* 1000);

log.setLogInfo(map);

vector<Log> logArr;

logArr.push\_back(log);

PalmLogSender::sendLogList(logArr);

} catch(exception &e) {

cout << "catch exception:" << e.what() << endl;

}

该条日志记录，玩家的侠客——李星云，进阶从绿变成了蓝+1。

## 任务，关卡场景等事件日志说明

当运营人员需要对游戏内的任务完成情况，关卡完成情况进行记录时，即可记录一条游戏自定义事件。在记录日志之前，游戏运营人员有必要向掌趣科技统计后台提供一份任务，关卡，以及其他自定义事件的id和名称对照表，并在掌趣科技统计后台描述任务，关卡，道具之间的关系。

任务，副本关卡日志的格式均为stageId,stageName,detail三个字段，三个字段的值完全由游戏研发人员和运营人员自行指定，例如新手引导任务可以用 newbieguide标识stageId，也可以用task001或其他字符串标识来表示stageId，任务完成，既可以用 SUCC表示，也可以用complete表示，还可以根据任务的中断，放弃，失败等多种状况，为同一个任务记录多条日志。对于同一种事件结果，建议使用同样的detail内容来描述。尽量做到同一个detail的含义一致，同一个事件结果的detail一致。

**需要注意的地方**：

无论是任务，关卡还是其他自定义事件日志，taskId/stageId/actId字段都必须是相对于该日志的logId唯一，不能够与其他任务，关卡或道具的id字段重复。

尽管任务，副本/场景，以及游戏自定义事件三个日志的字段数量和格式相同，但是我们希望研发人员能够按照预设的不同logid和logkey分别记录这三类日志，这样一来，便于将相对重要的日志同其它日志区分出来，也便于我们对任务和场景副本日志采用预设的特定算法来进行统计运算。

我们要求用以下的默认字段来记录事件日志的detail，以便统计后台可以用默认的配置来对游戏自定义事件进行计算成功率，通过率等进一步指标。

如果列表中的detail无法满足游戏的日志记录要求，也允许各个游戏使用自己定义的detail来记录日志，实际上在下面的日志范例中也有一些detail字段灵活发挥的情况。

|  |  |  |
| --- | --- | --- |
| **事件类型** | **detail** | **含义** |
| role-task | accept | 接任务 |
| fail | 任务失败 |
| cancel | 放弃任务 |
| succ | 达成任务条件 |
| complete | 任务完成 |
|  |  |  |
| role-stage | begin | 场景开始 |
| fail | 场景失败 |
| cancel | 场景取消 |
| end | 场景结束 |
|  |  |  |
| role-act | begin | 事件开始 |
| fail | 事件失败 |
| cancel | 事件取消 |
| end | 事件结束 |

### 玩家任务日志(11101)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LogId** | **日志名称** | **字段名** | **字段类型** | **字段说明** |
| 11101 | 任务日志 | taskId | String | 任务编码，英文或数字 |
| taskName | String | 人物名称，最好传中文 |
| detail | String | 相关信息，全为英文 |

***调用示例:***

try{

string serviceCode = “从SDKAPI中获取的serviceCode”;

std::map<string, string> map1;

map1.insert(pair<string, string>("taskId", "guide1"));

map1.insert(pair<string, string>("taskName", "引导流程1"));

map1.insert(pair<string, string>("detail", "accept"));

std::map<string, string> map2;

map2.insert(pair<string, string>("taskId", "guide1"));

map2.insert(pair<string, string>("taskName", "引导流程"));

map2.insert(pair<string, string>("detail", "complete"));

Log log;

log.setId("11101");

log.putServer("1");

log.putRoleId("11111111");

log.putRoleName("马克.吐温");

log.putRoleLevel("1");

log.putRoleVipLevel("0");

log.setServiceCode(serviceCode);

log.putUserId("0100010000000000000000000000000090019780");

log.setClientIp("112.23.22.22");

log.setCreateLogTime(time(NULL) \* 1000);

log.setLogInfo(map1);

Log log2;

log2.setId("11101");

log2.putServer("1");

log2.putRoleId("11111111");

log2.putRoleName("马克.吐温");

log2.putRoleLevel("1");

log2.putRoleVipLevel("0");

log2.setServiceCode(serviceCode);

log2.putUserId("0100010000000000000000000000000090019780");

log2.setClientIp("112.23.22.22");

log2.setCreateLogTime(time(NULL) \* 1000);

log2.setLogInfo(map2);

vector<Log> logArr;

logArr.push\_back(log);

logArr.push\_back(log2);

PalmLogSender::sendLogList(logArr);

} catch(exception &e) {

cout << "catch exception:" << e.what() << endl;

}

这两条日志表示，当前角色马克.吐温接受并且完成了新手流程任务1.

### 玩家关卡场景日志(11102)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LogId** | **日志名称** | **字段名** | **字段类型** | **字段说明** |
| 11102 | 副本场景日志 | stageId | String | 副本场景标识，英文或数字 |
| stageName | String | 副本场景名称，最好传中文 |
| detail | String | 相关信息，全为英文 |

***调用示例:***

try{

string serviceCode = “从SDKAPI中获取的serviceCode”;

std::map<string, string> map1;

map1.insert(pair<string, string>("stageId", "deadMine"));

map1.insert(pair<string, string>("stageName", "副本-死亡矿井"));

map1.insert(pair<string, string>("detail", "begin"));

std::map<string, string> map2;

map2.insert(pair<string, string>("stageId", "deadMine"));

map2.insert(pair<string, string>("stageName", "副本-死亡矿井"));

map2.insert(pair<string, string>("detail", "end"));

Log log;

log.setId("11102");

log.putServer("1");

log.putRoleId("11111111");

log.putRoleName("马克.吐温");

log.putRoleLevel("1");

log.putRoleVipLevel("0");

log.setServiceCode(serviceCode);

log.putUserId("0100010000000000000000000000000090019780");

log.setClientIp("112.23.22.22");

log.setCreateLogTime(time(NULL) \* 1000);

log.setLogInfo(map1);

Log log2;

log2.setId("11102");

log2.putServer("1");

log2.putRoleId("11111111");

log2.putRoleName("马克.吐温");

log2.putRoleLevel("1");

log2.putRoleVipLevel("0");

log2.setServiceCode(serviceCode);

log2.putUserId("0100010000000000000000000000000090019780");

log2.setClientIp("112.23.22.22");

log2.setCreateLogTime(time(NULL) \* 1000);

log2.setLogInfo(map2);

vector<Log> logArr;

logArr.push\_back(log);

logArr.push\_back(log2);

PalmLogSender::sendLogList(logArr);

} catch(exception &e) {

cout << "catch exception:" << e.what() << endl;

}

这两条日志表示，当前角色开始并且完成了场景死亡矿井副本.

### 玩家自定义行为日志(11103)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LogId** | **日志名称** | **字段名** | **字段类型** | **字段说明** |
| 11103 | 游戏自定义事件 | actId | String | 事件标识，英文或数字 |
| actName | String | 事件名称，最好传中文 |
| detail | String | 相关信息，参考上面detail定义，尽量用英文 |

***调用示例:***

try{

string serviceCode = “从SDKAPI中获取的serviceCode”;

std::map<string, string> map;

map.insert(pair<string, string>("actId", "fight-0001"));

map.insert(pair<string, string>("actName", "进入战斗-克尔苏加德"));

map.insert(pair<string, string>("detail", "begin"));

Log log;

log.setId("11103");

log.putServer("1");

log.putRoleId("11111111");

log.putRoleName("马克.吐温");

log.putRoleLevel("1");

log.putRoleVipLevel("0");

log.setServiceCode(serviceCode);

log.putUserId("0100010000000000000000000000000090019780");

log.setClientIp("111.111.111.111");

log.setCreateLogTime(time(NULL) \* 1000);

log.setLogInfo(map);

vector<Log> logArr;

logArr.push\_back(log);

PalmLogSender::sendLogList(logArr);

} catch(exception &e) {

cout << "catch exception:" << e.what() << endl;

}

这条日志表示，当前角色开始了一场战斗

### 玩家交互日志(12001)

游戏中除了角色与系统间发生的事件之外，还会有玩家之间的交互事件，这类事件使用交互事件日志来记录。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LogId** | **日志名称** | **字段名** | **字段类型** | **字段说明** |
| 12001 | 交互行为日志 | interactId | String | 自定义事件编码，数字或字母 |
| interactName | String | 自定义事件名称，建议写中文 |
| targetType | String | 目标类型，0:系统，1:玩家，2:自定义类型 |
| targetCode | String | 目标，若目标类型为玩家则为对方角色ID |
| targetInfo | String | 目标详细信息，若目标类型为玩家，则为对方逻辑服ID |
| detail | String | 相关信息，全为英文 |

与游戏自定义事件相比，交互事件多了一个交互对象的概念，targetType字段用于描述本次交互的对象是哪一类对象，0、系统交互，1、玩家之间互动，2、游戏自定义交互。

系统交互事件应该尽量使用游戏自定义事件来记录，除非交互对象较为重要需要额外记录。

targetCode用于描述当前玩家与具体哪一个对象进行了交互，当targetType为2时，targetCode必须为对方角色id，targetInfo必须为对方逻辑服标识。

***调用示例:***

try{

string serviceCode = “从SDKAPI中获取的serviceCode”;

std::map<string, string> map;

map.insert(pair<string, string>("interactId", "addfriend"));

map.insert(pair<string, string>("interactName", "添加好友"));

map.insert(pair<string, string>("targetType", "1"));

map.insert(pair<string, string>("targetCode", "0010000067543"));

map.insert(pair<string, string>("targetInfo", "珍妮.爱荷华"));

map.insert(pair<string, string>("detail", "SUCC"));

Log log;

log.setId("12001");

log.putServer("1");

log.putRoleId("11111111");

log.putRoleName("马克.吐温");

log.putRoleLevel("1");

log.putRoleVipLevel("0");

log.setServiceCode(serviceCode);

log.putUserId("0100010000000000000000000000000090019780");

log.setClientIp("111.111.111.111");

log.setCreateLogTime(time(NULL) \* 1000);

log.setLogInfo(map);

vector<Log> logArr;

logArr.push\_back(log);

PalmLogSender::sendLogList(logArr);

} catch(exception &e) {

cout << "catch exception:" << e.what() << endl;

}

这条日志表示当前玩家马克.吐温添加了玩家珍妮为好友

### 关卡场景事件日志综合使用举例

下面我们用一系列日志，来记录一个复杂的副本过程，以此作为游戏内自定义事件的开发样例

|  |  |  |
| --- | --- | --- |
| **logkey** | **事件** | **日志** |
| role-task | 接受任务：迪菲亚兄弟会 | time:2014-04-18 20:00:00 taskId:taskdifeiya003  taskName:迪菲亚兄弟会 detail:accept |
| role-stage | 进入死亡矿井 | time:2014-04-18 21:00:00 stageId:enterstage  stageName:死亡矿井 detail:deadmine |
| role-act | boos战格鲁巴托克开始 | time:2014-04-18 21:05:00 actId:fightdeadmineboos1  actName:boos战格鲁巴托克 detail:begin |
| role-act | boss战格鲁巴托克完成 | time:2014-04-18 21:07:00 actId:fightdeadmineboos1  actName:boss战格鲁巴托克 detail:end |
| role-act | boos战格范克里夫开始 | time:2014-04-18 21:20:00 actId:fightdeadmineboos4  actName:boos战格范克里夫 detail:begin |
| role-act | boss战格范克里夫失败 | time:2014-04-18 21:21:00 actId:fightdeadmineboos4  actName:boos战格范克里夫 detail:fail |
| role-act | 死亡 | time:2014-04-18 21:21:00 actId:dead  actName:角色死亡 detail:end |
| role-stage | 出死亡矿井 | time:2014-04-18 21:21:20 stageId:quitstage  stageName:死亡矿井 detail:deadmine |
| role-stage | 进入死亡矿井 | time:2014-04-18 21:23:20 stageId:enterstage  stageName:死亡矿井 detail:deadmine |
| role-act | 复活 | time:2014-04-18 21:23:21 actId:bufferuse  actName:复活  detail:relive |
| role-act | boos战格范克里夫开始 | time:2014-04-18 21:25:00 actId:fightdeadmineboos4  actName:boos战格范克里夫 detail:begin |
| role-act | boos战格范克里夫完成 | time:2014-04-18 21:30:00 actId:fightdeadmineboos4  actName:boos战格范克里夫 detail:end |
| role-task | 任务迪菲亚兄弟会达到完成条件 | time:2014-04-18 21:30:00 taskId:taskdifeiya003  taskName:迪菲亚兄弟会 detail:succ |
| role-stage | 出死亡矿井 | time:2014-04-18 21:40:20 stageId:quitstage  stageName:死亡矿井 detail:deadmine |
| role-task | 交任务：迪菲亚兄弟会 | time:2014-04-18 21:45:00 taskId:taskdifeiya003  taskName:迪菲亚兄弟会 detail:complete |

从中，运营人员可以发现，当前角色马克.吐温进入死亡矿井后，顺利击败第一个boss，但是在挑战最终boss范克里夫时不幸阵亡，玩家没有气馁，复活之后重整旗鼓，终于击败范克里夫，打通了死亡矿井，完成了任务迪菲亚兄弟会。

|  |  |  |
| --- | --- | --- |
| **logKey** | **事件** | **日志** |
| role-interact | 加入公会：圣光兄弟会 | time:2014-04-18 22:00:00 interactId:jioncrop  interactName:加入公会 targetType:0 targetCode:crop100991 targetInfo:crop detail:end |
| role-act | 获得一次公会等级提升 | time:2014-04-18 22:01:00 actId:croppromote  actName:公会等级提升 detail:end |
| role-interact | 对敌对公会：战神传说发起公会战争 | time:2014-04-18 22:10:00 interactId:declarewar  interactName:参战 targetType:0 targetCode:crop1000022 targetInfo:crop detail:begin |
| role-stage | 进入战场：长城 | time:2014-04-18 22:11:00 stageId:enterbattlefield  stageName:进入战场 detail:greatwall |
| role-interact | 攻击敌对玩家 | time:2014-04-18 22:11:21 interactId:attack  interactName:PVP攻击 targetType:1 targetCode:1002000100021 targetInfo:1002 detail:-3000 |
| role-interact | 被敌对玩家杀死 | time:2014-04-18 22:11:39 interactId:killed  interactName:pvp战败死亡 targetType:1 targetCode:1002000100021 targetInfo:1002 detail:-5000 |
| role-interact | 被敌对玩家嘲弄 | time:2014-04-18 22:11:42 interactId:laughed  interactName:遭到嘲弄 targetType:1 targetCode:1002000100021 targetInfo:1002 detail:end |
| role-debit | 使用收费道具：复仇之镰 | time:2014-04-18 22:11:50 roleLevel:50 roleVipLevel:3 updateType:0 itemId:killer1001 itemName:复仇之镰 itemCount:1 customer:use |
| role-interact | 攻击敌对玩家 | time:2014-04-18 22:11:21 interactId:attack  interactName:pvp攻击 targetType:1 targetCode:1002000100021 targetInfo:1002 detail:-99999 |
| role-interact | 击杀敌对玩家 | time:2014-04-18 22:11:39 interactId:kill  interactName:pvp战斗击杀对方 targetType:1 targetCode:1002000100021 targetInfo:1002 detail:-99999 |
| role-stage | 退出战场：长城 | time:2014-04-18 22:11:00 stageId:quitbattlefield  stageName:退出战场 detail:greatwall |

从上面一系列日志，我们可以发现，玩家马克吐温在加入公会圣光兄弟会后，被卷入了一场公会战争，他在战场上被敌人杀死，随后使用收费道具复仇杀死对方。

## 服务端异常日志(10011)

因为种种原因发生异常时，开发人员可以记录一条异常日志，如果客户端日志程序还能正常运行，该条日志将被发送到统计后台供技术人员和运营人员分析崩溃原因，如果客户端日志程序的日志发送功能已经无法正常运行，该日志有可能被存入客户端本的缓存文件中，在用户下次启动游戏时发送到统计接口

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LogId** | **日志名称** | **字段名** |  | **字段说明** |
| 10011 | 异常日志 | code | String | 异常标识 |
| message | String | 异常概要信息 |
| detail | String | 异常详细信息 |

***调用示例:***

try{

string serviceCode = “从SDKAPI中获取的serviceCode”;

std::map<string, string> map;

map.insert(pair<string, string>("code", "addfriend"));

map.insert(pair<string, string>("message", "添加好友"));

map.insert(pair<string, string>("detail", "{\”serverIp\”:\”113.31.128.143\”,\”port\”:\”13556\”,\”proxy\”:\”null\”,\”clientIp\”:\”118.44.112.28\”,\”mac\”:\”ac:f7:f3:10:ee:bb\”}"));

Log log;

log.setId("10011");

log.putServer("1");

log.putRoleId("11111111");

log.putRoleName("马克.吐温");

log.putRoleLevel("1");

log.putRoleVipLevel("0");

log.setServiceCode(serviceCode);

log.putUserId("0100010000000000000000000000000090019780");

log.setClientIp("111.111.111.111");

log.setCreateLogTime(time(NULL) \* 1000);

log.setLogInfo(map);

vector<Log> logArr;

logArr.push\_back(log);

PalmLogSender::sendLogList(logArr);

} catch(exception &e) {

cout << "catch exception:" << e.what() << endl;

}

该条日志记录了一次连接超时

## 自定义日志(11000)

由于游戏行业的产品类型丰富，各种新的玩法和子系统层出不穷，用固定格式固定字段的日志无法满足不同产品的日志记录和查询需求。我们在固定格式的日志之外，提供了一个非固定格式自定义日志，这份日志可以自己定义日志的标识（customId），可以自己定义日志中的字段（val是个字段与值的map）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LogId** | **日志名称** | **字段名** |  | **字段说明** |
| 11000 | 自定义日志 | customId | String | 日志标识，仅限字母，数字，下划线 |
| val | map | 日志内容，用map方式传入 |

**日志标识**:用来唯一确定一类日志的自定义的表的标识（后续会被识别为表名的一部分），同一类日志必须传入同一标识，标识只能由英文字母，数字，下划线构成，不允许传入特殊字符

**日志内容**:key-val方式(Map<String, String>)传入，接入者必须确保所有字段都处于同一层级，同一类日志的key必须保持一致，如果有修改某一类customId对应日志的需求，理论上只能增加字段，不能删除字段。

***调用示例：***

try{

string serviceCode = “从SDKAPI中获取的serviceCode”;

std::map<string, string> map;

map.insert(pair<string, string>("Time", "2016-05-31 12:01:00"));

map.insert(pair<string, string>("roleid", "123000"));

map.insert(pair<string, string>("rolename", "艾亚诺"));

map.insert(pair<string, string>("occupation", "战士"));

map.insert(pair<string, string>("lev", "10"));

map.insert(pair<string, string>("fightvalue", "5"));

map.insert(pair<string, string>("logintime", "2016-05-31 11:01:00"));

map.insert(pair<string, string>("Vip", "1"));

map.insert(pair<string, string>("Coin", "50000"));

map.insert(pair<string, string>("Diamond", "10"));

map.insert(pair<string, string>("Gold", "200"));

map.insert(pair<string, string>("Itemsn", "79001"));

map.insert(pair<string, string>("Price", "300"));

map.insert(pair<string, string>("Tax", "45"));

Log log;

log.setCustomId("tbl\_moneyflow");

log.setId("11000");

log.setServiceCode(serviceCode);

log.setClientIp("112.32.112.22");

log.setCreateLogTime(time(NULL) \* 1000);

log.setLogInfo(map);

vector<Log> logArr;

logArr.push\_back(log);

PalmLogSender::sendLogList(logArr);

} catch(exception &e) {

cout << "catch exception:" << e.what() << endl;

}

该条日志记录了一次交易所物品上架

# 附1.1货币规范

在MIS中的货币有总共十一种，货币ID是没有限定位数的纯自增值。

|  |  |  |
| --- | --- | --- |
| **货币ID** | **货币名称** | **货币单位** |
| 1 | 人民币 | 分 |
| 2 | 美元 | 美分 |
| 3 | 日元 | 円 |
| 4 | 港币 | 分 |
| 5 | 英镑 | 便士 |
| 6 | 新加坡币 | 分 |
| 7 | 越南盾 | 盾 |
| 8 | 台币 | 元 |
| 9 | 韩币 | 分 |
| 10 | 泰铢 | 萨当 |
| 11 | 卢布 | 戈比 |