**客户端-服务端协议**

王彬彬

2015/10/11

|  |  |  |
| --- | --- | --- |
| **时间** | **修改内容** | **修改人** |
| 2015/10/11 | 初稿，定义协议格式，包头格式，交互协议内容 | 王彬彬 |
| 2015/10/17 | 修改协议格式 | 王彬彬 |
| 2015/10/22 | 修改密码规定，为十六进制字符串，服务端下发的也是以十六进制字符串下发 | 王彬彬 |
| 2015/10/24 | 统一规整json的格式 | 王彬彬 |
| 2015/12/26 | 添加修改协议，修改任务字段 | 王彬彬 |
| 2016/2/2 | 任务结构添加包名字段 | 王彬彬 |
| 2016/2/22 | 完善协议，部分协议修改 | 王彬彬 |

目录

[协议说明 3](#_Toc444027792)

[包头 3](#_Toc444027793)

[包体 3](#_Toc444027794)

[错误码定义 3](#_Toc444027795)

[交互协议 4](#_Toc444027796)

[心跳 CMD=100 4](#_Toc444027797)

[登录 CMD=1000 5](#_Toc444027798)

[同步数据 CMD=2000 6](#_Toc444027799)

[同步新任务数据的ack CMD=2001 7](#_Toc444027800)

[服务端的通知CMD=2002 8](#_Toc444027801)

[启动任务 CMD=4000 8](#_Toc444027802)

[上报数据 CMD=3001 9](#_Toc444027803)

[通用数据结构定义 11](#_Toc444027804)

[任务信息定义 11](#_Toc444027805)

[收入信息定义 12](#_Toc444027806)

# 协议说明

客户端与服务端交互的数据格式采用加密的json。整个数据包包括包头和包体。登录协议的数据包有所不同。

## 包头

包头定义c描述如下：

typedef struct head

{

uint32\_t len; // 标示整个数据包长度

uint32\_t cmd; // 协议号

uint32\_t seq; // 请求的序列号

uint32\_t head\_len; // 包头的长度 = sizeof(HEAD)

uint32\_t uid; // 用户id

}HEAD;

## 包体

包体完全采用json格式。json使用utf-8编码。

包体的加密方式采用XTEA加密算法加密。

## 错误码定义

const static uint32\_t ERR\_SUCCESS = 0;

const static uint32\_t ERR\_SERVER\_TIMEOUT = 6000;

const static uint32\_t ERR\_PROCOL\_NOMATCH = 6001;

const static uint32\_t ERR\_SERVER\_ENCODE\_ERROR = 6002;

# 交互协议

服务器与客户端之间协议罗列如下：

1. 心跳；
2. 登录，此登录仅仅是与服务端交换秘钥，上传简要信息；
3. 客户端去服务端同步数据；该协议通用于获取任务，拉取财富值
4. 服务端给客户端主动下发通知，通知客户端去服务端同步数据；
5. 客户端主动上报数据给服务端；
6. 绑定用户信息；

## 心跳 CMD=100

C->S:

json描述如下：

{

"request": {

"time": 1445054155,

“transid”:123

}

}

S->C:

json描述如下：

{

"res": {

"ret": 0,

"err\_msg": "ok"

},

"time": 1445454155,

“transid”:123

}

Time为32位的整型时间戳

Transid为32位整型

## 登录 CMD=1000

登录的请求包组成为**包头+客户端密钥+客户端密钥经过XTEA加密过后的json**，描述如下：

HEAD + key + key\_encrypt(message),message如下C->S的描述。其中客户端的**密钥是随机的**，但是是**固定16字节**的**二进制字符串**。

登录的结果包的包体是用**用户密码**经过XTEA加密过后的json，描述如下：

HEAD + key\_encrypt(message),message如下S->C的描述。

**登录之后**的所有交互协议数据都由**服务端下发的密钥**采用XXTEA对其加密。

C->S:

json描述如下：

{

"request": {

"devid": "sdfguikltyui",

"dev\_type": 1,

"version": 10,

"time": 1444998292,

"passwd": "4c46ec3e3aea917736999754f55c04f8",

“transid”:123

}

}

S->C:

json描述如下：

{

"res": {

"ret": 0,

"err\_msg": "login ok"

},

“login”:{

"uid": 545878989,

"key": "22203a20226c6F67206f6b21222c0a20",

"time": 1444998492,

“transid”:123

}

}

注：

devid代表自己设备的id号，字符串；

dev\_type代表设备类型，32位整型，1为android，2为ios；

version代表用户所使用客户端的版本号，32位整型；

time代表登录时间，客户端的时间戳，32位整型。

passwd是用户devid进行md5之后的**十六进制字符串。****xxtea加解密时要转成二进制字符串，作为参数传入**

uid是服务端生成的id，32位整型，区分用户；

key服务端生成的密钥，之后的交互使用此密钥加密，**十六进制字符串，xxtea加解密时要转成二进制字符串，作为参数传入**；

time服务端当前的时间戳，32位整型。

## 同步数据 CMD=2000

客户端向服务端发起数据同步请求，登录之后的第一次同步为同步所有项目，当客户端收到具体项目的通知时才去同步具体项目的数据。同步到的数据有必要做去重处理。服务端对于每次同步数据的回包有个数据量限制，比如一个包里每次最多包含20条数据，同时告诉客户端服务器上面是不是还有数据。剩余的数据依赖于客户端再次发起同步请求。本条协议应该做个简化，每次只同步一个类型的数据，便于客户端维护。

C->S:

json描述如下

{

"request": {

"count": 3,

"items": [

{

"k": 1,

"v": 641845799

},

{

"k": 2,

"v": 441165795

},

{

"k": 3,

"v": 141165795

}

],

“transid”:123,

"time": 1444993118

}

}

count代表有几项，items包含具体需要同步的项目,k的取值如下：

1,代表新任务，

2,代表历史任务列表，

3,代表收入。

10，代表系统消息。

v为当前的同步点，64位无符号整型，初始值为0。

服务端对同步请求的回应

S->C:

json描述如下：

{

"res": {

"ret": 0,

"err\_msg": ""

},

"type": 1,

“index”:0,

"count": 2,

"items": [

{},

{}

],

"continue\_flag": 0,

"sync\_value": 441165796,

“transid”:123,

"time": 1444998492

}

注：

res代表基本返回信息，包含返回码和错误信息；

index代表本次收到的包是第几个，收到最后一个包后，要验证之前的index必须连续，否则中间的包有丢失，需要重新同步。防止数据丢失。

type代表业务类型，1,代表新任务，2,代表历史任务列表，3,代表收入；

count代表下面的items有多少条；

continue\_flag代表此次同步过程是否已经结束，0表示结束，1要继续同步；

sync\_value代表服务器当前的最新值，下次同步items中的v就是这个；

time代表服务器的时间。

每个回应结果仅仅包含一种type。items里面的都属于这种type。

## 同步新任务数据的ack CMD=2001

客户端向服务端回复收到的新任务数据应答，服务端收到应答才删掉数据。这样的机制能防止客户端数据丢失。

C->S:

json描述如下

{

"request": {

"count": 3,

"items": [

{

"k": 1,

"v": 641845799

},

{

"k": 1,

"v": 641845800

},

{

"k": 1,

"v": 641845801

}

] ,

"transid":123,

"time": 1444993118

}

}

其中的v的值代表收到的任务最大同步点，客户端每次上报都应该上报最大值。客户端也需要做逻辑保证，即收全本次同步的数据才回复ack给服务器

## 服务端的通知CMD=2002

服务端主动下发通知，通知客户端有任务.随之客户端向服务端请求同步任务

C->S:

json描述如下

{

"res": {

"ret": 0,

"err\_msg": ""

},

"type": 1,

“transid”:123,

"time": 144465121

}

注：所有的通知都是这种格式，仅仅是类型type取值不一样。客户端收到通知后就拿本地缓存的同步点去服务端同步信息。

## 上报任务数据 CMD=3001

上报任务完成情况，本协议是在任务启动并完成之后发送

C->S:

json描述如下：

{

"request": {

"id": 5634378,

"type": 0,

"done\_step": 1,

"desc": "task decsribe",

“transid”:123,

"time": 1444998292

}

}

注：

id为任务的id，由服务端告知的；

type为任务类型；

done\_step当前完成到第几步；

desc任务完成情况描述；

time客户端当前时间

S->C:

服务端确认收到数据后给客户端回ack

json描述如下：

{

"res": {

"ret": 0,

"err\_msg": ""

},

"type": 0,

"id": 5634378,

"done\_step": 1,

“transid”:123,

“reward”:1,

"time": 1444998299

}

如果任务合法，且用户的操作也合法，返回值为0，否则就是非法或者异常。返回值定义如下：

enum{

TASK\_REPORT\_OK = 0, // 任务上报成功，已给用户奖励

ERR\_TASK\_EXPIRE = 400, // 任务过期

ERR\_TASK\_INVALID = 403, // 任务不存在，可能被回收，也可能是异常数据

ERR\_TASK\_DONE = 405, // 任务已完成

UNKNOWN\_ERR = 200, // 未知异常，可能是服务器出故障

};

## 上报手机信息 CMD=3002

本协议用于上报手机的token，机型，位置等参数信息，用于后台push推送及定位

C->S:

json描述如下

{

"request": {

"token": “EA23SDC43429678KJX798”,

"phone\_type": “HUAWEI C199”,

“processor\_num”:4,

“model\_name”:” Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz”,

“imei”:”865344034402210”,

“mem\_total”, 4044548,

“mem\_free”, 27996,

“batt\_free”,”0.9”,

“location”:” {\"latitude\":40.071562,\"longitude\":116.321943}”,

“transid”:12,

"time": 1444998292

}

}

注：

Token即为手机的token

phone\_type手机机型完整描述

processor\_num手机cpu个数

model\_name手机cpu名称及频率信息

imei手机的imei

mem\_total手机总内存，int整型，单位是KB

mem\_free手机剩余内存，int整型，单位是KB

batt\_free手机电量剩余百分比，后台可以根据电量信息决策给用户推送多少信息

location当前的位置信息，json格式的字符串,里面仅包含经纬度，后续的扩展待定。可选项，因为不一定能获取成功。没获取成功就没有这个字段。目前应该不需要地图sdk

服务端对本协议不做回复

## 用户设置的信息 CMD=3003

在本系统中，为了提高账号安全系数，需要用户为本应用设置密码，昵称，且绑定一个邮箱或手机号。没有设置密码的账号，是不能被提现的。密码在传输时做了一次MD5。本协议走http通道，仅支持post请求。

C->S:

http请求的参数描述如下，最后要使用url encode

{

“cmd”:3003,

“nick\_name”:”hahawang”,

"passwd":” “ASDWQ231216876Js”,

“bind\_mail”:”[126783432@qq.com](mailto:126783432@qq.com),

“bind\_phone”:13267854562,

”sign”:”yes I am”,

“verify\_code”:”167523”,

“transid”:12,

"time": 1444998292

}

Urlencode request举例:

[www.dsp.com/setting](http://www.dsp.com/setting)

?cmd=3003

&nick\_name=hahawang

&passwd=ASDWQ231216876Js

&bind\_mail=[126783432@qq.com](mailto:126783432@qq.com)

&sign=yes%20I%20am

&verify\_code=167523

&transid=12

&time=1444998292

注：

http请求地址暂时没有，使用临时地址<http://139.196.42.222:3000/setting>

cmd字段固定为3003

nick\_name为用户设置的昵称

passwd为用户设置的密码，与邮箱或者手机号绑定

bind\_mail用户绑定的邮箱，可选项，因为可以选手机号绑定

bind\_phone用户绑定的手机号，可选项，因为可以选邮箱绑定，但二者必选其一

sign用户设置的个性签名描述。可选项

verify\_code绑定邮箱或者手机号时，手机或者邮箱收到的验证码，有效期为10分钟。验证码的获取在设置页面先点击获取完成，urlencode之后请求格式：[www.dsp.com/verify?type=phone&value=13267854562](http://www.dsp.com/verify?type=phone&value=13267854562)

或者：

www.dsp.com/verify?type=mail&value=[126783432%CEqq%2Acom](mailto:126783432@qq.com)

地址暂时未提供，使用临时地址http://139.196.42.222:3000/verify

S->C:

返回结果json描述如下：

{

"res": {

"ret": 0,

"err\_msg": ""

},

“nick\_name”:”hahawang”,

“transid”:123,

"time": 1444998299

}

注：

登录返回值取值如下：

Enum{

OK = 0, // 设置成功

ERR\_PARAM\_ERROR = 6210, // 参数错误

ERR\_CODE\_INVAILD = 406, // 验证码失效

UNKNOWN\_ERROR = 200, // 未知错误，可能是服务器错误

};

## 启动任务 CMD=4000

用户点击任务的时候发送。客户端带上taskid和url启动任务，可能是下载任务，也可能是其他的任务，后续的涉及到web服务，所以需要带上url。启动任务这条协议的目的是检测任务状态以及修改任务可用数，如果是首次启动任务，就校验任务状态，并且在任务状态ok的情况下将任务池中的本任务的可用任务数减一，反之之前已经启动过，则仅仅是检验任务状态。

C->S:

json描述如下

{

"request": {

"id": 5634378,

“transid”:12,

"type": 1,

"url": “www.dsp.task\_resource/121213”,

"time": 1444998292

}

}

注：

id为任务的id，由服务端告知的；64位整型

type为任务类型；

time客户端当前时间

url为任务的地址，对应web服务

S->C:

{

"res": {

"ret": 0,

"err\_msg": ""

},

"type": 1,

“id”: 5634378,

“transid”:123,

"time": 1444998299

}

返回值取值如下：

Enum{

ERR\_TASK\_FIRST\_START = 0, // 任务首次启动，正常情况

ERR\_TASK\_ALREADY\_START = 401, // 任务已经启动，但未完成，正常情况

ERR\_TASK\_EXPIRE = 402, // 任务已经过期

ERR\_TASK\_INVALID = 403, // 任务已失效，可能是任务已经关闭了

ERR\_TASK\_OVERFLOW = 404, // 任务已经溢出，被抢光了

ERR\_TASK\_DONE = 405, // 任务已经完成过

UNKNOWN\_ERR = 200, // 未知异常，可能是服务器出故障

};

## 第三方登录 CMD=6000

用户点击第三方登录的时候，调用第三方验证完成后，触发该协议，只有用第三方登录的用户才有资格提现到第三方账户中。

C->S:

json描述如下

{

"request": {

"type": 1,

“login\_flag”:1,

“openid”:”oxTWJHVksdgau76JKHVjvfcadkhvvWkzxhgadcadsvcajksdh”,

“token”:”A9DB6534247C4894”,

"nickname":”wangerxiao”,

“passwd”:”95AE8A9CDB8764573424E7C46D89345C”,

“sex”:1,

"province": “shanghai”,

“city”:”shanghai”,

“country”:”chain”,

“headimgurl”:”www.qq.weixin/headpic/hhiad.jpg”,

“desc”:”fetch cash.”,

“email”:”1sd7923@163.com”,

“phone”:”13671869346”,

“transid”:123,

"time": 1444998292

}

}

注：

Type第三方类型，取值如下：

Enum {

WEIXIN = 1,

QQ = 2,

ALIPAY = 3,

BAIDU = 4,

};

login\_flag标识是否强制登录，1位强制覆盖，如果之前已经有同类型的第三方账户信息则覆盖。默认填0，不覆盖，等服务器返回做提示后再决定是否做强制覆盖登录。

openid为第三方登录成功返回的id

token是手机的token，与1000协议的token一致

nickname，sex，province，city，country，headimgurl同样从第三方获取

passwd是自己设置的，这里是MD5之后的hex字符串

desc登录的描述，目前没有用，后续可能用到

email 邮箱，用户自己填写，非必须，但是有此信息可能会收到较多推送任务

phone手机号，用户自己填写，非必须，但是有此信息可能会收到较多推送任务

Time当前客户端时间戳

S->C:

{

"res": {

"ret": 0,

"err\_msg": ""

},

"type": 1,

“transid”:123,

"time": 1444998299

}

登录返回值取值如下：

Enum{

OK = 0, // 登录成功

ERR\_ALREADY\_BIND = 601, // 已经登录过，绑定过第三方账户

UNKNOWN\_ERROR = 200, // 未知错误，可能是服务器错误

};

## 提现请求 CMD=6666

多么喜庆的协议号！用户点击提现发送此协议。服务端返回可能是提现成功，也可能是提示用户进行第三方登录，跳转到第三方登录界面，登录之后才能进行提现操作。

C->S:

json描述如下

{

"request": {

"id":”145692367abc”,

“transid”:12,

"type": 1,

“withdraw\_cash”:1000,

“openid”:”oxTWJHVksdgau76JKHVjvfcadkhvvWkzxhgadcadsvcajksdh”,

“passwd”:” BE768A9CDB8AC4573424E7C46D893498”

“user\_name”:”wang五”,

“desc”:”fetch cash.”

"time": 1444998292

}

}

注：

Id是客户端生成的提现操作的id，标识一次提现操作

Transid是客户端每次请求的标识

Type第三方类型，标识提现到哪个账户，取值如下：

Enum {

WEIXIN = 1,

QQ = 2,

ALIPAY = 3,

};

withdraw\_cash，就代表要提现的金额咯，int类型，单位是分(1元=10角=100分)

openid，第三方账户的id，转账的对象

passwd，用户在本系统中设置的提现密码

user\_name，用户真实姓名

desc，描述信息,目前就填上述默认值吧

time客户端的时间戳

S->C:

json描述如下

{

"res": {

"ret": 0,

"err\_msg": ""

},

"type": 1,

“trade\_no”: “100045677897645678912”,

“withdraw\_cash”:1000,

“transid”:123,

“pre\_cash”:”30.00”,

“useable\_cash”:”10.00”,

“fetched\_cash”:”20.00”,

“desc”:”fetch cash.”,

"time": 1444998299

}

Ret取值如下：

Enum{

OK = 0, // 提现成功

ERR\_NOT\_BIND = 609, // 未绑定第三方账户，需要重新登录绑定一下

ERR\_AMOUNT\_LIMIT = 610, // 提现金额太小

ERR\_FETCH\_FAILED = 611, // 提现失败，可能是商家余额不足，可能是没有权限，可能是参数错误，可能是商家openid错误，可能是签名出错，可能证书出错

ERR\_SYSTEM\_BUSY = 612, // 系统繁忙，稍后再试

ERR\_NAME\_MISMATCH = 613, // 真实姓名校验失败，第三方账户需要绑定身份证的

ERR\_FATAL\_ERROR = 614, // 两次相同的请求，但是参数不一样

UNKNOWN\_ERROR = 200，// 服务端故障

};

Type，指明是哪种第三方账户

trade\_no,提现的流水单号，唯一标识一次提现

withdraw\_cash代表经过服务端校验完成后提现的金额, int类型，单位是分(1元=10角=100分)

pre\_cash,useable\_cash,fetched\_cash分别代表提现之后，当前账户中的最新金额数据

# 通用数据结构定义

## 任务信息定义

json描述如下：

{

"taskinfo": {

"id": 144789767878879,

"type": 0,

"name": "use app1",

"link": "www.baidu.com/xxx",

"desc": "task describe",

"pkgname": "com.dsp",

"size": 1024,

"pay": "2.5",

"tstep": 3,

"cstep": 0,

"status": 0,

"publisher": "bbwang",

"icon": "www.dsp.icon/download",

"task\_num":1500,

"used\_num":100,

"stime": 1444998492,

"etime": 1445003127

}

}

id，任务的id，由服务端生成指定；

type，任务类型，当前未细分，默认值填0吧；

name，任务名称；

link，任务所在地，链接地址；

desc，任务的简单描述；

pkgname，包名，适用于下载app安装的任务，用于区分客户机是否已经安装该app

size，任务大小，需要耗费的流量；

pay，发布者为完整任务支付的金额；

tstep，任务总共分为几步；

cstep，任务当前进度；

status，任务当前所处状态。0表示未开始可执行，1表示已开始未完成，2表示已完成，3表示已抢光，4表示过期

publisher，任务发布者

icon，表示图标地址

task\_num，任务数量

used\_num，任务已执行的数量

stime，任务起始时间；

etime，任务结束时间，过期任务作废的。

## 收入信息定义

json描述如下：

{

"cashinfo": {

pre\_cash:”30.00”,

useable\_cash:”10.00”,

fetched\_cash:”20.00”

}

}

pre\_cash，表示已经分派给用户的金额

useable\_cash，表示可以提取的金额

fetched\_cash，表示用户已经提取的金额

pre\_cash = useable\_cash + fetched\_cash