# 邮游API-1

1 服务器地址：139.196.30.53 端口 32107，域名：[www.521uu.cc](http://www.521uu.cc)

2 请求格式分为两部分

A 请求头部分4个字节（也即第一个int数据）是这个请求总长度。

B 然后buffer.getUTF()一个字符串是逻辑名。然后就是各个逻辑的参数个数详细参考4 具体逻辑部分

3 返回数据格式也分为两份

A 返回头部分4个字节（也即第一个int数据）是这个请求总长度。

B 然后buffer.getUTF()一个字符串是逻辑名。和请求名称相同也即请求传的逻辑名称原封不动返回。

C 然后4个字节（int数据）表示服务器处理成功与否 public static final int RESP\_RESULT\_SUCC = 0;//成功

public static final int RESP\_RESULT\_FAIL = 1;//失败

D 最后一个字符串是josn的数据，详细参考4具体逻辑部分

4 具体逻辑部分（参数是按顺序的）

1. AppToken注册的验证码

请求参数：

参数1： String key = buffer.getUTF();//手机号码

返回josn数据：

成功 ：“{\”result\”:\”验证码已发送\”}”

失败 ：“失败的提示信息”

2、AppRegist 用户注册

请求参数：

参数1：String account = buffer.getUTF();//注册的手机号

参数2：String code = buffer.getUTF();//验证码

参数3：String pwd = buffer.getUTF();//密码

参数4：String nickname = buffer.getUTF();//昵称唯一

参数5：String name = buffer.getUTF();//用户姓名

参数6：String address = buffer.getUTF();//收货地址

返回josn数据：

成功 ：“{\”result\”:\”注册成功\”}”

失败 ：“失败的提示信息”

3、AppLogin用户登录

请求参数：

参数1：String account = buffer.getUTF();//用户注册手号

参数2: String pwd = buffer.getUTF();//登录密码

返回josn数据：

成功 ：“{\”user\”:附件1 用户数据,\”deals\”,[附件2 最近交易数据1，附件2最近交易数据2...,附件2最近交易数据n]}”

失败 ：“失败的提示信息”

4、AppFindPassword 用户找回密码

请求参数：

参数1：String account = buffer.getUTF();//用户注册手号

返回数据：

成功 ：“{\”result\”:\”密码已通过短信发送到你手机了\”}”

失败 ：“失败的提示信息”

5、AppAccountOprate 用户添加银行卡

请求参数：

参数1：byte type = buffer.get();//操作类型（0添加；1删除)

参数2：long uid = buffer.getLong();//用户编号

参数3：String name = buffer.getUTF();

参数4： String value = buffer.getUTF();(type=1没有)

参数5： String openName = buffer.getUTF();(type=1没有)

参数6： String peopleName = buffer.getUTF();(type=1没有)

返回数据：

成功：“{\”account\”:[Account1,Account2,...Accountn]}”(注意是附件1用户数据里面的Account类型)

失败：“失败的提示信息”

6、AppAddressChange 用户收货地址修改

请求参数：

参数1：byte type = buffer.get();//操作类型（0添加；1删除)

参数2：long uid = buffer.getLong();//用户编号

参数3：String address = buffer.getUTF();//地址

返回数据:

成功：“{\”addresses\”:[“地址1”,“地址2”,...“地址n”]}”

失败：“失败的提示信息”

7、AppDeployDeal 交易发帖

请求参数：

参数1：long uid = buffer.getLong();//发布者编号

参数2：byte sellFlag = buffer.get();//发布卖贴还是买帖

参数3：byte deployType = buffer.get();//发布方式 0普通发送,1 推送发送

参数4：String typeSTr = buffer.getUTF();

参数5：byte type = (byte)(typeSTr.equals("入库") ? 0 : 1);//交割类型

参数6：String bourse = buffer.getUTF();//文交所

参数7：String title = buffer.getUTF();//名称

参数8：float price = Float.parseFloat(buffer.getUTF());

参数9：int num = buffer.getInt();

参数10：String monad = buffer.getUTF();//单位

参数11：String validTime = buffer.getUTF();//有效时间

参数12：String createTime = TimeUtils.nowChStr();

参数13：String other = buffer.getUTF();//有效时间

参数14：byte helpFlag = buffer.get();//帮组标志

返回数据：

成功：“{\”deal\”:附件4 交易详情}”

失败：“失败的提示信息”

8、 AppDealPage交易区分页浏览

请求参数：

参数1：int page = buffer.getInt();//当前页码

参数2：int num = buffer.getInt();//每页数据条数

参数3：String type = buffer.getUTF();//类型:null/入库/现货

参数4：String bourse = buffer.getUTF();//文交所/交易地:其他文交所/xx文交所;其他城市/xx城市，

参数5：String title = buffer.getUTF();//null/藏品名称

参数6：String seller = buffer.getUTF();//null/卖家昵称

参数7：String buyer = buffer.getUTF();//null/买家昵称

参数8：String valid = buffer.getUTF();//null/到目前无效/到目前有效

后面的帅选条件如果是null是这个条件不限制。

返回数据：

成功：

”{

\”list\”:[附件3 交易列表1,附件3 交易列表2,...,附件3交易列表n],

\”page\”:page(请求传的值)，

\”left\”:num(还有多少也可以浏览)

}”

失败：“失败提示信息”

附件1 用户数据

public class UserData{

long userId;//用户编号

String tel;//电话

String face;//头像名称

String nikeName;//昵称

String title;//称号

String registTime;//注册时间

List<String> addresses;//地址

String name;//姓名

String identity;//身份验证

byte push = 1;//推送设置

String signature;//签名

List<Account> accounts;//绑定银行账户

AppraiseRecord ar = new AppraiseRecord();//评价次数记录

byte breach;//违约次数

String other = "";//备注信息

MyselfNum mn = new MyselfNum();//和有有关的数据

}

public class Account{

String account;//银行账号

String name;//银行名称

String openAddress;//开户行地址

String openName;//开户人姓名

String addTime;//添加时间

}

public class AppraiseRecord{

int good;//累计好评次数

int normal;//累计中评次数

int bad;//累计差评次数

}

public class AppraiseRecord{

int emailNum;//新邮件数量

int friendNum;//好友数量

}

附件2 最近交易数据

public class RecentDeal {

String bource;//文交所或者是成交城市

String type;//交易类型

String dealName;//交易藏品名称

float price;//价钱

long dealId;//交易编号

long orderId;//订单编号

String time;//成交时间

int num;//数量

String monad;//单位

}

附件3 交易列表

public class TransformDealListInfo{

long id;//编号

long uid;//用户编号

byte type;//类型0入库，1现货

String issueName;//发布人昵称

byte sellFlag;//出售帖还是求购帖

byte helpFlag;//可以使用中介服务;0未开启，1开启。

boolean revoke;//true撤销，false未撤销

String bourse;//文交所名称

String name;//藏品名称

float price;//藏品单价

String monad;//单位

int num;//藏品数量

int orderNums;//已抢单的数量

String validTime = "永久";//有效时间

String createTime;//创建时间

}

附件4 交易详情

public class TransformDealDetail {

long id;//编号

long uid;//用户编号

String issueName;//发布人昵称

byte sellFlag;//出售帖还是求购帖

byte type;//类型0入库，1现货

byte helpFlag;//可以使用中介服务;0未开启，1开启。

boolean revoke;//true撤销，false未撤销

String bourse;//文交所名称

String name;//藏品名称

float price;//藏品单价

String monad;//单位

int num;//藏品数量

String validTime = "永久";//有效时间

String createTime;//创建时间

String other;//其他描述

List<TransformOrder> orders ;//成交数据

List<TransformRevert> reverts ;//回复数据

}

public class TransformOrder {

long orderId;//订单编号

long grabId;//抢单人编号

String grabName;//抢单人姓名

int num;//抢单数量

float price;//抢单价钱

byte helpFlag;//0普通模式,1中介模式

byte state;//订单状态

String time;//订单状态的时间

}

public class TransformRevert {

long dealId;//交易编号

long sayerId;//回复人编号

String sayerName;//回复人名称

String time;//回复时间

long sayToId;//表示回复目标的比编号

String sayToName;//表示回复目标的名称

String context;//内容

}

附件5 关于字节流读取

DataBuffer类

package com.keyking.coin.service.net.buffer;

import java.io.UnsupportedEncodingException;

import java.nio.ByteBuffer;

import java.nio.ByteOrder;

import org.apache.mina.core.buffer.IoBuffer;

public class DataBuffer {

public static final byte STRING\_TYPE\_BYTE = 1;

public static final byte STRING\_TYPE\_SHORT = 2;

public static Charset DEFAULT\_CHARSET = Charset.UTF\_8;

public static byte DEFAULT\_STRING\_TYPE = STRING\_TYPE\_SHORT;

private IoBuffer buffer;

public static DataBuffer allocate(int len) {

DataBuffer joyBuffer = new DataBuffer();

joyBuffer.buffer = IoBuffer.allocate(len);

joyBuffer.buffer.setAutoExpand(true);

return joyBuffer;

}

public static DataBuffer wrap(ByteBuffer buffer) {

return wrap(buffer,true);

}

public static DataBuffer wrapBuffer(IoBuffer buffer,boolean flag) {

DataBuffer joyBuffer = new DataBuffer();

joyBuffer.buffer = buffer;

joyBuffer.buffer.setAutoExpand(flag);

return joyBuffer;

}

public static DataBuffer wrap(ByteBuffer buffer,boolean flag) {

DataBuffer joyBuffer = new DataBuffer();

joyBuffer.buffer = IoBuffer.wrap(buffer);

joyBuffer.buffer.setAutoExpand(flag);

return joyBuffer;

}

public static DataBuffer wrap(byte[] datas) {

return wrap(datas,true);

}

public static DataBuffer wrap(byte[] datas, boolean flag) {

DataBuffer joyBuffer = new DataBuffer();

joyBuffer.buffer = IoBuffer.wrap(ByteBuffer.wrap(datas));

joyBuffer.buffer.setAutoExpand(flag);

return joyBuffer;

}

public static DataBuffer wrap(byte[] datas, int a,int b) {

return wrap(datas,a, b,true);

}

public static DataBuffer wrap(byte[] datas, int a,int b, boolean flag) {

DataBuffer joyBuffer = new DataBuffer();

joyBuffer.buffer = IoBuffer.wrap(ByteBuffer.wrap(datas,a, b));

joyBuffer.buffer.setAutoExpand(flag);

return joyBuffer;

}

public IoBuffer buf() {

return buffer;

}

public int capacity() {

return buffer.capacity();

}

public int position() {

return buffer.position();

}

public DataBuffer position(int index) {

buffer.position(index);

return this;

}

public int limit() {

return buffer.limit();

}

public DataBuffer limit(int limite) {

buffer.limit(limite);

return this;

}

public DataBuffer mark() {

buffer.mark();

return this;

}

public int markValue() {

return buffer.markValue();

}

public DataBuffer reset() {

buffer.reset();

return this;

}

public DataBuffer clear() {

buffer.clear();

return this;

}

public DataBuffer flip() {

buffer.flip();

return this;

}

public DataBuffer rewind() {

buffer.rewind();

return this;

}

public int remaining() {

return buffer.remaining();

}

public boolean hasRemaining() {

return buffer.hasRemaining();

}

public DataBuffer duplicate() {

DataBuffer joyBuffer = new DataBuffer();

joyBuffer.buffer = buffer.duplicate();

return joyBuffer;

}

public DataBuffer slice() {

DataBuffer joyBuffer = new DataBuffer();

joyBuffer.buffer = buffer.slice();

return joyBuffer;

}

public DataBuffer slice(int paramInt) {

DataBuffer joyBuffer = new DataBuffer();

joyBuffer.buffer = buffer.slice();

joyBuffer.limit(paramInt);

return joyBuffer;

}

public DataBuffer sliceNew() {

return sliceNew(limit() - position());

}

public DataBuffer sliceNew(int len) {

return wrap(array(), position(), len);

}

public byte[] array() {

return buffer.array();

}

public byte[] arrayToPosition() {

int i = position();

rewind();

return getByteArray(i);

}

public int arrayOffset() {

return buffer.arrayOffset();

}

public byte get() {

return buffer.get();

}

public short getUnsigned() {

return buffer.getUnsigned();

}

public DataBuffer put(byte data) {

buffer.put(data);

return this;

}

public byte get(int data) {

return buffer.get(data);

}

public short getUnsigned(int paramInt) {

return buffer.getUnsigned(paramInt);

}

public DataBuffer put(int paramInt, byte paramByte) {

buffer.put(paramInt, paramByte);

return this;

}

public DataBuffer get(byte[] data, int index1, int index2) {

buffer.get(data, index1, index2);

return this;

}

public DataBuffer get(byte[] data) {

buffer.get(data);

return this;

}

public byte[] getByteArray(int index) {

byte[] arrayOfByte = new byte[index];

get(arrayOfByte);

return arrayOfByte;

}

public DataBuffer put(ByteBuffer buffer) {

buffer.put(buffer);

return this;

}

public DataBuffer put(byte[] data, int index1, int index2) {

buffer.put(data, index1, index2);

return this;

}

public DataBuffer put(byte[] data) {

buffer.put(data);

return this;

}

public DataBuffer compact() {

buffer.compact();

return this;

}

public ByteOrder order() {

return buffer.order();

}

public DataBuffer order(ByteOrder order) {

buffer.order(order);

return this;

}

public char getChar() {

return buffer.getChar();

}

public DataBuffer putChar(char ch) {

buffer.putChar(ch);

return this;

}

public char getChar(int index) {

return buffer.getChar(index);

}

public DataBuffer putChar(int index, char ch) {

buffer.putChar(index, ch);

return this;

}

public short getShort() {

return buffer.getShort();

}

public int getUnsignedShort() {

return buffer.getUnsignedShort();

}

public DataBuffer putShort(short num) {

buffer.putShort(num);

return this;

}

public DataBuffer putUnsigned(short num) {

buffer.put(\_toUnsigned(num));

return this;

}

public short getShort(int num) {

return buffer.getShort(num);

}

public int getUnsignedShort(int num) {

return buffer.getUnsignedShort(num);

}

public DataBuffer putShort(int index, short num) {

buffer.putShort(index, num);

return this;

}

public DataBuffer putUnsigned(int index, short num) {

buffer.put(index, \_toUnsigned(num));

return this;

}

public int getInt() {

return buffer.getInt();

}

public long getUnsignedInt() {

return buffer.getUnsignedInt();

}

public int getMediumInt() {

return buffer.getMediumInt();

}

public int getUnsignedMediumInt() {

return buffer.getUnsignedMediumInt();

}

public int getMediumInt(int num) {

return buffer.getMediumInt(num);

}

public int getUnsignedMediumInt(int num) {

return buffer.getUnsignedMediumInt(num);

}

public DataBuffer putMediumInt(int num) {

buffer.putMediumInt(num);

return this;

}

public DataBuffer putMediumInt(int index1, int index2) {

buffer.putMediumInt(index1, index2);

return this;

}

public DataBuffer putInt(int num) {

buffer.putInt(num);

return this;

}

public DataBuffer putUnsignedShort(int num) {

buffer.put(\_toUnsignedShort(num));

return this;

}

public int getInt(int num) {

return buffer.getInt(num);

}

public long getUnsignedInt(int num) {

return buffer.getUnsignedInt(num);

}

public DataBuffer putUnsignedShort(int index, int num) {

byte[] data = \_toUnsignedShort(num);

for (int i = 0; i < data.length; ++i){

buffer.put(index + i, data[i]);

}

return this;

}

public DataBuffer putInt(int index, int num) {

buffer.putInt(index, num);

return this;

}

public long getLong() {

return buffer.getLong();

}

public DataBuffer putLong(long num) {

buffer.putLong(num);

return this;

}

public DataBuffer putUnsignedInt(long num) {

buffer.put(\_toUnsignedInt(num));

return this;

}

public long getLong(int num) {

return buffer.getLong(num);

}

public DataBuffer putLong(int index, long num) {

buffer.putLong(index, num);

return this;

}

public DataBuffer putUnsignedInt(int index, long num) {

byte[] data = \_toUnsignedInt(num);

for (int i = 0; i < data.length; ++i){

buffer.put(index + i, data[i]);

}

return this;

}

public float getFloat() {

return buffer.getFloat();

}

public DataBuffer putFloat(float num) {

buffer.putFloat(num);

return this;

}

public float getFloat(int num) {

return buffer.getFloat(num);

}

public DataBuffer putFloat(int index, float num) {

buffer.putFloat(index, num);

return this;

}

public double getDouble() {

return buffer.getDouble();

}

public DataBuffer putDouble(double num) {

buffer.putDouble(num);

return this;

}

public double getDouble(int num) {

return buffer.getDouble(num);

}

public DataBuffer putDouble(int index, double num) {

buffer.putDouble(index, num);

return this;

}

public String getUTF() {

int len = getInt();

return \_byteArrayToString(getByteArray(len),DEFAULT\_CHARSET);

}

public DataBuffer putUTF(String str) {

byte[] datas = \_stringToByteArray(str,DEFAULT\_CHARSET);

putInt(datas.length);

return put(datas);

}

public String getHexDump() {

return buffer.getHexDump();

}

public String getHexDump(int num) {

return buffer.getHexDump(num);

}

public DataBuffer skip(int len) {

buffer.skip(len);

return this;

}

private String \_byteArrayToString(byte[] data,Charset set) {

try {

return new String(data, set.charsetName());

} catch (UnsupportedEncodingException exp) {

exp.printStackTrace();

}

return "";

}

private byte[] \_stringToByteArray(String str,Charset set) {

try {

return str.getBytes(set.charsetName());

} catch (UnsupportedEncodingException exp) {

exp.printStackTrace();

}

return new byte[0];

}

private byte \_toUnsigned(short num) {

return toHH(num, 1)[0];

}

private byte[] \_toUnsignedShort(int num) {

return toHH(num, 2);

}

private byte[] \_toUnsignedInt(long num) {

return toHH(num, 4);

}

public byte[] toHH(long data, int num) {

byte[] datas = new byte[num];

for (int i = num - 1; i >= 0; --i){

datas[i] = (byte) (int) (data >> (num - i - 1) \* 8 & 0xFF);

}

return datas;

}

public enum Charset{

UTF\_8("UTF-8"),

UTF\_16("UTF-16"),

UTF\_16BE("UTF-16BE"),

UTF\_16LE("UTF-16LE"),

UTF\_32("UTF-32"),

UTF\_32BE("UTF-32BE"),

UTF\_32LE("UTF-32LE"),

ISO\_8859\_1("ISO-8859-1"),

US\_ASCII("US-ASCII"),

GBK("GBK"),

GB2312("GB2312");

private String charsetName;

private Charset(String paramString){

charsetName = paramString;

}

public String charsetName(){

return charsetName;

}

}

}