# 友盟+ Finplus 产品接入手册v6.6

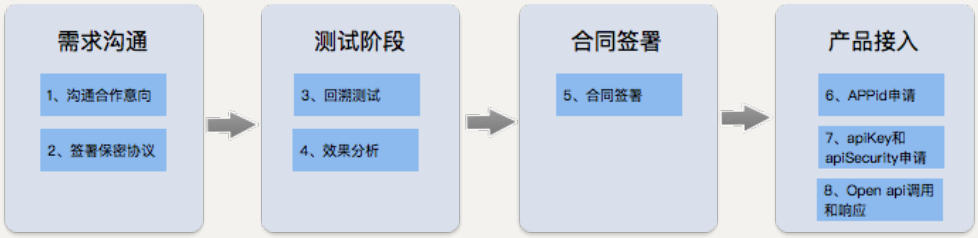
金融产品线

2019/06/24

# 一、产品介绍

Finplus产品是友盟+通过大数据、机器学习等技术提供的大数据风控和营销解决方案，主要通过风险指数预测的方式，以OpenAPI调用的形式对外提供服务。

本文档主要介绍OpenAPI调用的接入方法及接入阶段的流程说明，包括接入准备、接口调用说明、接口请求示例、接口响应示例、常见问题和处理办法。如果您对Finplus测试、合同阶段有任何疑问，可发送邮件至：finplus-apply@service.umeng.com。




# 二、接入准备

## 友盟+账号注册

未开通账号客户需前往友盟+官网申请申请注册，已开通可沿用现有的账号。

## apiKey和apiSecurity申请

apiKey和apiSecurity需授权开通，首次开通需登录友盟+账号，前往https://developer.umeng.com/open-api/state点击“获取身份认证信息”，不需提交其他信息，友盟+开放平台将给您分配apiKey和apiSecurity；已开通账户也可通过该网址查看apiKey和apiSecurity。


- 每个友盟+账号关联唯一apiKey和apiSecurity信息，作为您访问

OpenAPI时签名计算使用，apiKey为7位数字，apiSecurity为10到12位的字符串。

- apiKey和apiSecurity是访问OpenAPI的唯一身份认证信息，请妥善保存，不要泄露。
- 您开通的的所有app_id都可通过友盟+账号下的apiKey和apiSecurity信息来访问Finplus服务。
- OpenAPI支持IP白名单访问，产品接入时，您需提供友盟+账号及其要添加的IP白名单列表并发送邮件至：finplus-apply@service.umeng.com。注意，OpenAPI只支持IP白名单访问，后续IP白名单变更时，你可以同样方式申请变更。IP白名单应是公网IP地址，而非局域网IP地址。

## 公钥秘钥获取

生成方式一（推荐）：使用以下一键生成工具**（内附使用说明）

- Windows：下载
- MACOSX：下载

  解压打开文件夹，直接运行“密钥生成器 SHAwithRSA1024*V1.0.bat*”*（WINDOWS）
  或“SHAwithRSA1024*V1.0.command”（MACOSX），点击“生成RSA密钥”，会自动生成公私钥，然后点击“打开文件位置”，即可找到工具自动生成的密钥。

生成方式二：也可以使用OpenSSL工具命令生成**

首先进入OpenSSL工具，再输入以下命令。

```
OpenSSL> genrsa -out rsa_private_key.pem   1024  #生成私钥
```

```
OpenSSL> pkcs8 -topk8 -inform PEM -in
rsa_private_key.pem -outform PEM -nocrypt -out
rsa_private_key_pkcs8.pem #Java开发者需要将私钥转换成PKCS8格式
```

```
OpenSSL> rsa -in rsa_private_key.pem -pubout -out
rsa_public_key.pem #生成公钥
```

```
OpenSSL> exit #退出OpenSSL程序
```

经过以上步骤，您可以在当前文件夹中（OpenSSL运行文件夹），看到rsa_private_key.pem（RSA私钥：适用于.NET和PHP的开发者来说，无需进行pkcs8命令行操作）、**rsa_private_key_pkcs8.pem**（pkcs8格式RSA私钥：适用于Java的开发者，将pkcs8在console中输出的私钥去除头尾、换行和空格，作为

开发者私钥）和**rsa_public_key.pem**（对应RSA公钥）3个文件。请您注意保管好私钥，同时将公钥提交给Finplus，用于验证签名使用。

# app_id申请

## 什么是app_id

- app_id可以看做是您在使用Finplus服务时的账号。
- 每个app_id的调用结果，返回且只返回一个score，以及合同中约定的其他字段。因此，您申请的每一个app_id对应一个模型。
- 为适应您为不同使用场景建立不同的模型，您可以基于签署合同的公司名称申请多个app_id，计费系统会综合您申请的所app_id下的调用次数进行计算。

## 申请资料

申请app_id，**您需发送邮件至：**finplus-apply@service.umeng.com**，**并同时提供如下信息：

- 公司名称：签署合同的公司名。
- 公司英文或拼音缩写：管理后台导出账单数据时，文件名中会包含该名称。
- 友盟+账号：如果有使用友盟+其他服务， 最好使用已有账号，有利于查询覆盖率的提升；如果没有账号，请前往https://passport.umeng.com/signup?lang=zh_CN注册。友盟+账号最多支持5人在线，请自行维护好登陆权限和人员变动带来的权限交割问题。
- 联系人：用于接收开通、数据回溯、商务沟通等重要通知，后续可修改。
- 联系手机号：用于接收开通、数据回溯、商务沟通等重要通知，后续可修改。
- 联系邮箱：用于接收开通、数据回溯等重要邮件，后续可修改。为保证正常接收系统重要邮件通知，建议提供群邮箱（邮件组），同时配置**邮件地址白名单(cplus_sample@service.umeng.com)和邮箱域名白名单(service.umeng.com)**。
- 公钥：为保证您的数据、调取安全，计费次数准确。您需基于我们提供的工具或方法生成公钥私钥，并将公钥（**rsa_public_key.pem**）以邮件附件方式提供，不要直接提供内容，以避免产生格式问题。公钥和您的友盟+账号绑定，您在该账号下开通的所有app_id共用同一套公钥。后续您接口请求使用的公钥，需和提交我们的公钥保持一致。另外，为避免公私钥更换影响线上业务，建议您妥善保管公私钥文件。
- IP白名单：产品接入时，您需提供IP白名单列表并发送邮件至：finplus-apply@service.umeng.com。后续IP白名单变更时，您可以同样方式申请IP白名单变更。注意，IP白名单应是公网IP地址，而非局域网IP地址。
- 账户每日限额：默认200万，当日账号下所有app_id共享该额度，超过200万次调用，会返回错误状态码，后续可修改。本设置主要防止系统错误带来的调用次数浪费。

# 三、接口调用说明

## 概览

Finplus只支持https协议访问，按照客户提供的应用ID，返回对应应用所设置的标签与模型分。当调用成功时，响应的顶层对象tags字段会包含请求的业务数据：格式为tag_name: model_value，其中tag_name是应用选中标签的英文名，model_value是应用所选中标签下所选中模型的分值，以下是示例，后续接口调用解释均围绕该例展开

```
https请求方式:GET
https://gateway.open.umeng.com/param2/1/com.umeng.finplus/umeng.cplus.creditTrace/ur_apiKey?app_id=ur_app_id&req_id=idfa&id_value=EC3B219D-881C-4E6C-BF58-1A51D3B37D36&req_id=20190304211111&sign=LwbybbFpO3jNFd%2FSoxzW%2BDthDXkqs%2B6OPSVCaKxFvmKsBb05NUXkheMq3tOCyRn9GEvaTjDCnr23%0D%0ADEalDnmnRtWdq7xmUaJM7%2BfbubvJLMMLJweBryL6RwQZv83im6%2Bg3jQ%2BdCMvU9uZK9ghuxd%2FgNtx%0D%0A0xd9sQk33QH8W6Kngxg%3D%0D%0A&_aop_signature=58F12888265C5779B0A1CDCF240C1270C35A9E48
```

## 请求地址

您只需将请求地址部分的ur_apiKey值替换，apiKey值申请和查看详见apiKey和apiSecurity申请章节。

```
https://gateway.open.umeng.com/param2/1/com.umeng.finplus/umeng.cplus.creditTrace/ur_apiKey
```

## 请求参数

app_id、id_type、id_value、req_id、sign、_aop_signature都是接口必选参数，所有参数都应当作为url的query组件提供，并进行urlencode。

| 参数名 | 类型 | 是否必须 | 示例 | 备注 |
| --- | --- | --- | --- | --- |
| app_id | Long | 是 | 290504603875856 | 邮件申请后获取 |
| id_type | String | 是 | imei/idfa | 有效的imei, idfa |
| id_value | String | 是 | EC3B219D-881C-4E6C-BF58-1A51D3B37D36 | imei/idfa<br>示例依次为: |

| | | | | |
|---|---|---|---|---|
| | | | | 860529038487085<br>EC3B219D-881C-4E6C-BF58-1A51D3B37D36 |
| req_id | String | 是 | 14913216151305305011 | 客户指定的，全局唯一的用于对账审计的流水号，推荐生成方式为uuid；将在结果中原样返回。 |
| sign | String | 是 | | 使用1024bit 的RSA私钥对请求中所有其他参数按字典序构造的querystring进行签名所得到的Base64结果，参数发生变化时均需重新生成签名。 |
| _aop_signature | String | 是 | | 请求签名，其值是根据OpenAPI权限开通时得到的apiSecurity和本次请求的URL共同生成，请参考API签名规则。 |

## app_id

邮件申请后获取 ，联调成功后，不需更换app_id。

## id_type和id_value

id_type为参数类型，值为：imei/idfa；同时为便于联调，我们提供测试id(imei：60529038487085、IDFA:EC3B219D-881C-4E6C-BF58-1A51D3B37D36)供您联调使用。

### imei/meid

imei和meid两者都是安卓手机的唯一设备码，可通过系统设置-关于手机中查找imei和meid信息，区别点在于：

- imei适用于GSM、WCDMA制式手机（移动/联通手机），序列号共有15-17位数字：第一部分类型分配码(TAC,Type Allocation Code)由8位数字组成(早期是6位)，是区分手机品牌和型号的代码；第二部分最终装配地代码(FAC,Final Assembly Code)由2位数字构成，FAC码用于生产商内部区分生产地代码；第三部分序列号(SNR,Serial Number)由第9位开始的6位数字组成，区分每部手机的生产序列号；第四部分最后1位备用检验码不参与信息传输。
- meid适用于CDMA制式手机（电信手机），序列号共有15位数字：第一部分区域分配标识(RC,Regional Code)由2位数字组成，是授权imei

码分配机构的代码；第二部分生产机构代码（TAC,Type Allocation Code）由第3位开始的6位数字组成，区分手机的生产商；第三部分生产序列号（SNR,Serial Number）由第9位开始的6位数字组成，区分每部手机的生产序列号；第四部分最后1位备用校验码不参与信息传输。
- 部分运营商合约机以a或A开头，后面为数字，也是合规的格式

### idfa

- idfa是用来标志苹果手机的唯一标志码，共有36位字符串，包括32位16进制字符（ 每一位为ABCDEF0123456789）和4位连字符"-"，格式为8-4-4-4-12。
- 示例：EC3B219D-881C-4E6C-BF58-1A51D3B37D36 。

## req_id

req_id是由用户生成的全局唯一的用于对账审计的流水号，推荐生成方式uuid v1。

## sign

key的顺序必须按app_id、id_type、id_value、req_id这个字典序拼接生产sign值，任意参数变化，sign值都需重新计算生成，具体详见接口请求示例。

## _aop_signature

_aop_signature是根据OpenAPI权限开通时得到的apiSecurity和本次请求的URL共同生成，签名规则为：

1. 构造**urlPath**：从param2开始截取，到“?”为止，urlPath=param2/1/com.umeng.finplus/umeng.cplus.creditTrace/ur_apiKey
2. 构造**拼装参数**：将参数(app_id、id_type、id_value、req_id、sign)的key和value(去掉“=”号)拼在一起并按字母从小到大排序，最后按顺序拼一起。
3. 构造签名因子：将**urlPath**及其拼装参数合并，示例请求的签名因子为 param2/1/com.umeng.finplus/umeng.cplus.creditTrace/ur_apiKeyapp_idur_app_idreq_ididfaid_valueEC3B219D-881C-4E6C-BF58-1A51D3B37D36req_id20190304211111signLwbybbFpO3jNFd%2FSoxzW%2BDthDXkqs%2B6OPSVCaKxFvmKsBb05NUXkheMq3tOCyRn9GEvaTjDCnr23%0D%0ADEalDnmnRtWdq7xmUaJM7%2BfbubvJLMMLJweBryL6RwQZv83im6%2Bg3jQ%2BdCMvU9uZK9ghuxd%2FgNtx%0D%0A0xd9sQk33QH8W6Kngxg%3D%0D%0A
4. 计算_**aop_signature**：对签名因子执行hmac_sha1算法uppercase (hex (hmac_sha1 (s, apiSecurity))得到该请求的签名值，示例的签名

值为58F12888265C5779B0A1CDCF240C1270C35A9E48

其中：

- apiSecurity：OpenAPI的签名秘钥，详见apiKey和apiSecurity申请。
- sign：是使用1024bit 的RSA私钥对请求中所有其他参数key按字典序构造的querystring进行签名所得到的Base64结果。（生成sign的时候key的顺序必须按app_id、id_type、id_value、req_id这个字典序拼接。）
- hmac_sha1：通用的hmac_sha1算法，各编程语言一般都有对应类库
- hex：转为十六进制
- uppercase：转为大写字符

# 四、接口请求示例

为让您更好接入Finplus服务，我们提供完整的java代码请求示例，调试时可直接复制HTML版本的java代码，配置您自己的参数后即可接入服务。

## java

```
//package com.wayne;

import org.apache.commons.codec.binary.Base64;

import javax.crypto.Cipher;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.InputStreamReader;
import java.net.URLEncoder;
import java.nio.charset.Charset;
import java.security.*;
import java.security.interfaces.RSAPrivateKey;
import java.security.interfaces.RSAPublicKey;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import java.util.*;




public class UmApiUtil {
```

```
    // 字节流转字段串，此函数直接使用即可
    public static final String
openApiHead="https://gateway.open.umeng.com/openapi/";//
这个是finplus固定的url头，不需要修改

    public static final char[] digital =
"0123456789ABCDEF".toCharArray();

    //读取公私钥文件，此函数直接使用即可
    public static String readKeyFile(String keyFile)
            throws Exception {
        BufferedReader br = new BufferedReader(new
                InputStreamReader(new
FileInputStream(keyFile)));
        String readLine = null;
        StringBuilder sb = new StringBuilder();
        while ((readLine = br.readLine()) != null) {
            if (readLine.charAt(0) != '-') {
                sb.append(readLine);

 sb.append(System.getProperty("line.separator"));
            }
        }
        return sb.toString();
    }
    //，此函数直接使用即可
    public static String readKeyStr(String keyStr)
            throws Exception {
        String[] lines = keyStr.trim().split("\n");
        StringBuilder sb = new StringBuilder();
        for (String line : lines) {
            String readLine = line.trim();
            if (readLine.charAt(0) != '-') {
                sb.append(readLine);

 sb.append(System.getProperty("line.separator"));
            }
        }
        return sb.toString();
    }

    //构造私钥对象，此函数直接使用即可
    public static PrivateKey loadPrivateKey(String

 privateKey) throws Exception {
        // 解密由base64编码的私钥
        byte[] keyBytes =
```

```
 Base64.decodeBase64(privateKey.getBytes());
        // 构造PKCS8EncodedKeySpec对象
        PKCS8EncodedKeySpec pkcs8KeySpec = new
                PKCS8EncodedKeySpec(keyBytes);
        // KEY_ALGORITHM 指定的加密算法
        KeyFactory keyFactory =
                KeyFactory.getInstance("RSA");
        // 取私钥匙对象
        return keyFactory.generatePrivate(pkcs8KeySpec);
    }

    //构造公钥对象，此函数直接使用即可
    public static PublicKey loadPublicKey(String

 publicKey) throws Exception {
        byte[] buffer =
Base64.decodeBase64(publicKey.getBytes());
        KeyFactory keyFactory =
                KeyFactory.getInstance("RSA");
        X509EncodedKeySpec keySpec = new
                X509EncodedKeySpec(buffer);
        return keyFactory.generatePublic(keySpec);
    }

    //rsa加密，此函数直接使用即可
    public static byte[] rsaEncrypt(RSAPublicKey
                                            publicKey,
byte[] plainTextData)
            throws Exception {
        Cipher cipher = Cipher.getInstance("RSA");
        cipher.init(Cipher.ENCRYPT_MODE, publicKey);
        return cipher.doFinal(plainTextData);
    }

    //rsa解密，此函数直接使用即可
    public static byte[] rsaDecrypt(RSAPrivateKey
                                            privateKey,
byte[] cipherData)
            throws Exception {
        Cipher cipher = Cipher.getInstance("RSA");
        cipher.init(Cipher.DECRYPT_MODE, privateKey);
        return cipher.doFinal(cipherData);
    }
    //此函数直接使用即可
    public static String rsaSign(PrivateKey privateKey,
```

```
                                    String plainText)
throws Exception {
        Signature sig =
                Signature.getInstance("SHA1WithRSA");
        sig.initSign(privateKey);
        byte[] plainTextData = plainText.getBytes("UTF-
8");
        sig.update(plainTextData);
        byte[] signature = sig.sign();
        return new
String(Base64.encodeBase64Chunked(signature));
        //return Base64.encodeBase64String(signature);
    }
    //检查公私钥是否配对，输出true代表配对，否则请与友盟+一起检查
公私钥是否给错或者配错，此函数直接使用即可
    public static boolean rsaSignCheck(PublicKey

publicKey, String plainText, String sign) {
        try {
            //sign = URLDecoder.decode(sign, "UTF-8");
            Signature sig =

 Signature.getInstance("SHA1WithRSA");
            sig.initVerify(publicKey);
            byte[] plainTextData =
                    plainText.getBytes("UTF-8");
            sig.update(plainTextData);
            byte[] signature =

 Base64.decodeBase64(sign.getBytes());
            return sig.verify(signature);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return false;
    }

    /**
     * hmacSha1加密
     * @param datas 参与加密的数据
     * @param key 参与加密的密钥
     * @return
     * 此函数直接使用即可
     */
    public static byte[] hmacSha1(String[] datas, byte[]
key) {
```

```
        SecretKeySpec signingKey = new
SecretKeySpec(key, "HmacSHA1");
        Mac mac = null;
        try {
            mac = Mac.getInstance("HmacSHA1");
            mac.init(signingKey);
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e.getMessage(),
e);
        } catch (InvalidKeyException e) {
            throw new RuntimeException(e.getMessage(),
e);
        }
        try {
            for (String data : datas) {

 mac.update(data.getBytes(Charset.forName("utf-8")));
            }
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage(),
e);
        }
        return mac.doFinal();
    }

    /**
     * HmacSHA1 加密
     * @param urlPath 参与加密的url部分，从协议（param2）开始
截取，到“?”为止
     * @param params 请求的参数列表
     * @param apiSecurity 网关平台提供的 apiSecurity
     * @return
     * 此函数直接使用即可
     */
    public static String encodeWithHmacSha1(String
urlPath, Map<String,String> params, String apiSecurity)
{
        List<String> paramValueList = new
ArrayList<String>();
        for (Map.Entry<String, String> entry :
params.entrySet()) {
            paramValueList.add(entry.getKey() +
entry.getValue());
        }
        final String[] datas = new String[1 +
paramValueList.size()];
        datas[0] = urlPath;
```

```
        //key按字典序排序
        Collections.sort(paramValueList);
        for (int i = 0; i < paramValueList.size(); i++)
{
            datas[i+1] = paramValueList.get(i);
        }
        final byte[] signature = hmacSha1(datas,
apiSecurity.getBytes());
        return encodeHexStr(signature);
    }

    /**
     * 字节流转十六进制字符串
     * @param bytes
     * @return
     * 此函数直接使用即可
     */
    public static String encodeHexStr(final byte[]
bytes){
        if(bytes == null){
            return null;
        }
        char[] result = new char[bytes.length * 2];
        for (int i = 0; i < bytes.length; i++) {
            result[i*2] = digital[(bytes[i] & 0xf0) >>
4];
            result[i*2 + 1] = digital[bytes[i] & 0x0f];
        }
        return new String(result);
    }



    public static String GetUrlNew(String app_id, String
id_type,String id_value,String req_id,String
publicKeyPath,String privateKeyPath,String apiKey,String
apiSecurity)
            throws Exception
    {
        //【1】 注意! paramstr的所有【key】需要按字典序拼接,
app_id,id_type,id_value,req_id
        String paramStr =
"app_id="+app_id+"&id_type="+id_type+"&id_value="+id_val
ue+"&req_id="+req_id;
        String signStr =
rsaSign(loadPrivateKey(readKeyFile(privateKeyPath)),
paramStr);
```

```
        if(

 rsaSignCheck(loadPublicKey(readKeyFile(publicKeyPath)),
paramStr, signStr)//生产使用时，此公私钥检验配对的代码可以去掉
        ){


            String signRes = URLEncoder.encode(signStr,
"UTF-8");
            //【3】 注意！最终的sign结果,必须要做URLEncoder!
            //System.out.println("signRes\t"+signRes);
            //【4】 注意！sigh签名完后长这样，如果长度与以下示例
不是一个格式，请重新检查签名过程，以下为示例
            // signRes
LwbybbFpO3jNFd%2FSoxzW%2BDthDXkqs%2B6OPSVCaKxFvmKsBbO5NU
XkheMq3tOCyRn9GEvaTjDCnr23%0D%0ADEalDnmnRtWdq7xmUaJM7%2B
fbubvJLMMLJweBryL6RwQZv83im6%2Bg3jQ%2BdCMvU9uZK9ghuxd%2F
gNtx%0D%0A0xd9sQk33QH8W6Kngxg%3D%0D%0A


            //OpenAPI签名需要的参数
            Map<String, String> params = new
HashMap<String, String>();
            params.put("app_id", app_id);
            params.put("req_id", req_id);
            params.put("id_value", id_value);
            params.put("id_type", id_type);
            //【6】 注意！在api签名的时候还需要加入sigh，参与签
名的是UrlEncoder之前的原始值signStr
            params.put("sign", signStr);

            String
apiUrlPath="param2/1/com.umeng.finplus/umeng.cplus.credi
tTrace/"+apiKey;//注意这个头的固定的，apiKey是根据在友盟+后台申
请的
            //encodeWithHmacSha1里已经实现对params的key按字
典序排序
            String aopSignature =
encodeWithHmacSha1(apiUrlPath, params,apiSecurity);

 //System.out.println("aopSignature\t"+aopSignature);
            //【7】 注意！aopSignature示例长这样，
58F12888265C5779B0A1CDCF240C1270C35A9E48，如果不是这样的格
式，请重新检查aopSignature生成逻辑

            String newApiFinalUrl =
                    openApiHead+apiUrlPath+"?"
```

```
                                +paramStr
                                + "&sign="
                                + signRes
                                + "&_aop_signature="
                                + aopSignature;

            return newApiFinalUrl;
            //【8】 注意！最终可以放到浏览器里访问的url长这样

 //https://gateway.open.umeng.com/openapi/param2/1/com.umeng.finplus/umeng.cplus.creditTrace/ur_apiKey?app_id=ur_app_id&id_type=imei&id_value=860529038487085&req_id=20190304211111&sign=LwbybbFpO3jNFd%2FSoxzW%2BDthDXkqs%2B6OPSVCaKxFvmKsBbO5NUXkheMq3tOCyRn9GEvaTjDCnr23%0D%0ADEalDnmnRtWdq7xmUaJM7%2BfbubvJLMMLJweBryL6RwQZv83im6%2Bg3jQ%2BdCMvU9uZK9ghuxd%2FgNtx%0D%0A0xd9sQk33QH8W6Kngxg%3D%0D%0A&_aop_signature=58F12888265C5779B0A1CDCF240C1270C35A9E48
            //【9】 注意！若以上url可以访问，但返回不是code0，请根据文档中的【http状态码】和【业务状态码】章节排查问题


        }else{
            String errMsg="请确认公钥是否给错，或者公私钥配对关系出现错误";
            return errMsg;
        }




    }

    //需要您详细调的main函数
    public static void main(String[] args) throws
            Exception {
        //需要您配置的参数-----------------------------------------
        String app_id="ur_app_id";  //请在此填新开通的app_id

        String apiKey="ur_apiKey"; //OPENAPI apiKey
        String apiSecurity="ur_apiSecurity";//OPENAPI apiSecurity
```

```
        //apiKey和apiSecurity需授权开通，首次开通需登录友盟+账号，前往https://developer.umeng.com/open-api/state点击“获取身份认证信息”，不需提交其他信息，友盟+开放平台将给您分配apiKey和apiSecurity；已开通账户也可通过该网址查看apiKey和apiSecurity。

        String req_id = UUID.randomUUID().toString();//客户请求流水号
        //req_id是客户指定的查询流水号， 返回结果biz_id是友盟+返回的查询流水号
        // req_id与biz_id用于对账或问题追查。
        // 建议req_id推荐生成方式为uuid v1，或者与时间有关的全局唯一字符串

        //公钥文件与私钥文件的位置，请注意修改这里
        String publicKeyPath="ur_publicKeyPath";
        String privateKeyPath="ur_privateKeyPath";

        //String id_type="idfa";
        //String id_value="EC3B219D-881C-4E6C-BF58-1A51D3B37D36";// 示例idfa号，此idfa号能拿到code 0的结果

        String id_type="imei";//"imei";
        String id_value="860529038487085";//"351637012970806370";// 示例imei号，此imei号能拿到code 0的结果

        //需要您配置的参数-----------------------------------------

        System.out.println("newApiFinalUrl是finplus接口的访问方式，将以下url贴到浏览器里即可访问");
        String newApiFinalUrl=GetUrlNew( app_id, id_type, id_value, req_id, publicKeyPath, privateKeyPath, apiKey, apiSecurity);
        System.out.println(newApiFinalUrl);


    }

}
```

# 五、接口响应示例

## HTTP状态码

用户通过OpenAPI访问Finplus服务，将会返回以下4类HTTP响应状态码。

1. status code=200：API成功响应。
2. status code=400：API使用错误导致的异常，以下是常见的错误码类型和解决方法。

| ERROR_CODE | REASON | RESOLUTION |
| --- | --- | --- |
| IPAPPWLDecline | Request IP 不在IP白名单 | 联系友盟+，将访问公网IP添加IP白名单列表中。 |
| APIACLDecline | apiKey未授权 | 联系友盟+，申请apiKey授权。 |
| SignatureMissing | 签名缺失或网络超时 | 签名缺失：参照签名过程实现示例章节进行接口签名。 网络超时：增加超时设置 |
| ParamMissing | 参数缺失 | 检查API入参。 |
| ParamIllegal | 参数非法 | 检查API入参。 |
| 未返回错误信息 | apiKey错误或apiSecurity错误或签名方式非法 | 参照OpenAPI调用依次检查三者信息，如仍不能解决，请联系我们。 |

3. status code=403：网关限流异常，请第一时间联系我们。
4. status code=500：服务器内部异常，请第一时间联系我们。

注：部分请求失败可能是网络抖动引起客户端socket连接断开，建议用户增加超时处理。

## 响应字段

用户通过状态码作为调用返还状态的依据，当调用成功且计费时，状态码固定为0，其他值表示调用出现了错误或者异常，查阅附录中的状态码对照表可以确定错误原因并采取相应对策。

| NAME | TYPE | SAMPLE | COMMENT |
| --- | --- | --- | --- |
| code | Integer | 0/108/109/... | 响应状态码，建议作为查询状态 |

| | | | |
|---|---|---|---|
| | | | **的唯一标识，以下success与message仅作为辅助** |
| success | Boolean | true/false | 业务逻辑是否计费的标记 |
| message | String | OK | 响应状态消息，仅供内部使用和联调时参考信息，不建议生产服务使用 |
| response | Object | {tags: null,biz_id: "57901bd3-b699-40a4-92f6-f7778aa8afb9",req_id: "your unique request id"} | 响应内容，其中tags字段为查询内容（空字段key存在但value为null），biz_id为finplus生成流水号，req_id为客户指定的流水号原样返回 |

## 业务状态码说明

状态码用于标识调用成功与否以及错误类型。共有三种状态分类：

1. 成功调用；
2. 调用失败（调用错误）；
3. 调用失败（Finplus系统错误）。

Finplus会对产生的所有状态码进行监控和报警，当出现系统错误时，我们会第一时间进行处理。

### 成功调用且计费

| CODE | MESSAGE | REASON | SOLUTION | 对应SUCCESS取值 |
|---|---|---|---|---|
| 0 | OK | 调用成功且计费 | 取response中的tags字段使用 | true |

### 成功调用但不计费

当出现以下状态码时候，代表调用成功。**经验来看，返回”108”，”109”时，违约风险会高于正常返回（code:0）20%以上，建议给返回”108”，”109”的查询一个稍低的评分，或者将”108”，”109”作为binary变量融合进下一层模型**

| CODE | MESSAGE | REASON | SOLUTION | 对应SUCCESS取值 |
|---|---|---|---|---|
| 108 | Credit.id_unconverted | 调用成功但不计费 | 输入ID未覆盖，一般108占比在10%以内 | false |
| 109 | Credit.empty_content | 调用成功 | 180天内行为不足以建模 | false |

| | | |
|---|---|---|
| | 但不计费 | 或计算字段（模型预测或明细），注意对于含明细的app_id只要有一个字段可计算出就不会返回109，一般109占比在20%以内 |

## 返回结果样例

### code=0样例

code值为0，success标记为true。response中存储了返回的结果。包含字段结构如下：

- tags：荷载数据，格式为tag.name: model.value或tag.id: model.value。即标签英文名或id与其模型取值构成的对象，具体是显示英文名还是id显示请与商务沟通；tag处理空字段时，key存在但value为null
    - 预测分对应的tag为credit_score，如果不是开放平台，取值范围为[300,850]的int，两侧双闭区间；如果预测分credit_score来自开放平台，取值为[0,1]的四位小数float，两侧双闭区间；预测分的内容和版本等，是根据与商务沟通得到的app_id来唯一指定
    - 其他tag根据双方商务合同的文档进行说明
- req_id:透传用户请求中的req_id，是用于对账的流水凭证。
- biz_id:由finplus服务生成的业务流水ID，uuid格式。

```
{
code: 0,
success: true,
message: "OK",
response: {
tags: {
credit_score: 697
},
biz_id: "80e249e4-3198-4fdb-9ad9-0aec4aa11e75",
req_id: "your unique request id"
}
}
```

### code=108样例

```
{
code: 108,
success: false,
message: "credit.id_unconverted",
response: {
tags: null,
biz_id: "b560b107-5e6a-40c3-a9e6-f7695ea63744",
req_id: "your unique request id"
}
}
```

**code=109样例**

```
{
code: 109,
success: false,
message: "credit.empty_content",
response: {
tags: null,
biz_id: "57901bd3-b699-40a4-92f6-f7778aa8afb9",
req_id: "your unique request id"
}
}
```

**code=22样例**

```
{
code: 22,
success: false,
message: "cplus.invalid_id_type",
response: {
tags: null,
biz_id: "d7e3d0d5-a82d-4f94-9cab-e044ccff56f6",
req_id: "your unique request id"
}
}
```

## 超时设置

建议设置超时时间为3秒，若出现code1或超时3s请马上重试一次，若还是超时3s，请记录对账id（req_id）和biz_id后马上与我们联系。

# 六、常见问题和处理办法

## 调用失败（调用错误）

当出现以下状态码时，一般是由于您在调用接口时的配置或参数错误造成的，请您根据Reason中的描述，按solution中给出的方案进行重试，如果还未解决问题，您可与我们联系，发邮件至：finplus-support@service.umeng.com。

| CODE | MESSAGE | REASON | SOLUTION | 对应SUCCESS取值 |
|---|---|---|---|---|
| 10 | cplus.invalid_parameters | 参数错误 | 不存在的参数、参数没有按照字典序降序、参数未进行urlencode处理 | false |
| 18 | Cplus.verification_failed | 签名错误 | 检查签名逻辑或私钥是否正确，一般是签名key没有字典序或者没有urlencode造成 | false |
| 19 | Cplus.insufficient.quota | 调用量限制 | 天调用量达到上限 | false |
| 22 | Cplus.invalid.id.type | 无效的id类型 | 参数id_type需要是idfa,imei中的一项 | false |
| 105 | Credit.invalid.id.type | id type缺失 | 参数id_type缺失 | false |

## 调用失败（Finplus系统错误）

当出现以下状态码时，一般是由于Finplus系统产生的错误造成的，不会计费，建议您首先进行重试，间隔1分钟，若重试3次还未解决问题，您可发邮件至：finplus-support@service.umeng.com**与我们联系。

| CODE | MESSAGE | REASON | SOLUTION | 对应SUCCESS取值 |
|---|---|---|---|---|
| 1 | Cplus.error | Finplus系统内部 | 若在联调阶段，请直接联系友盟+商务查看app_id的配置问题；若为在 | false |

| | | |
|---|---|---|
| | 错误 | 线使用阶段，请在**收到code1后马上重试一次**，若还是返回code1，请记录对账id（req_id）后与我们联系 |

由于系统迭代需求，未来我们可能会增加更多的状态码，届时我们会通过您注册时留下的邮箱通知到您。

# 七、附录

## IMEI/IDFA 采集方法

imei，idfa的详细介绍和采集方法，见文档《IMEI_MEID_IDFA查询方法V1.0》