目录

[引言 二](#_Toc519581338)

[修订历史 二](#_Toc519581339)

[1.总体说明 二](#_Toc519581340)

[1.1 概要 二](#_Toc519581341)

[1.2接入准备 二](#_Toc519581342)

[2. 接口总体说明 三](#_Toc519581343)

[2.1 接口URL 三](#_Toc519581344)

[2.2数据类型 三](#_Toc519581345)

[2.3数据输入格式规范 三](#_Toc519581346)

[2.4数据返回格式规范 三](#_Toc519581347)

[3. 接口总体说明 四](#_Toc519581348)

[3.1获取商户的调用密钥接口 四](#_Toc519581349)

[3.2获取灵芝分接口 四](#_Toc519581350)

[4. ErrorCode定义 六](#_Toc519581351)

[5. 加密算法（Java） 八](#_Toc519581352)

[6. Http Post调用范例(Java) 十一](#_Toc519581353)

# 引言

风控服务平台为合作机构提供风控服务，合作机构金融产品通过服务接口推送必要的业务数据到风控服务平台，风控服务平台负责从第三方获取相应的数据进行数据建模分析，并将模型分析风控结果反馈给合作机构金融产品系统。

## 修订历史

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **修订人** | **修订日期** | **修订描述** |
| V1.0 | 陈晓亮 | 2018-07-03 | 无 |
| V1.1 | 陈晓亮 | 2018-07-17 | 增加返回值类型说明；增加errorcode说明 |
| V1.2 | 罗礼龙 | 2018-07-18 | 将调用范例打成jar包；添加jar包调用范例；更新了商户的调用密钥接口 |
|  |  |  |  |
|  |  |  |  |

# 1.总体说明

## 1.1 概要

该文档是**浙江孚临科技有限公司**对外统一接口使用说明文档。

该文档适用户以下读者：

项目管理者：依据该文档接口说明，进行项目开发管理工作；

软件开发人员：依据该文档接口说明，完成相关功能代码的编写工作；

软件测试人员：依据该文档接口说明，完成相关功能代码的测试工作。

## 1.2接入准备

1）合作机构金融产品系统；

2）向风控服务平台申请合作机构金融产品系统的商户ID（customerId）和商户产品Id（customerProdId）；

3）获取风控服务平台为该商户分配的秘钥（可调用5.1接口进行重新生成）

4）向风控服务平台获取传输OSS文件的子账号信息

# 2. 接口总体说明

## 2.1 接口URL

测试URL: AppPath = http://saas-test.fullink.tech:端口号

正式环境: AppPath = http://saas.fullink.tech/ (注意：不需要加端口号)

Http POST调用请参考：[第6章](#_6._Http_Post调用范例(Java)_1)

## 2.2数据类型

|  |  |
| --- | --- |
| 术语 | 说明 |
| Integer | 整型 |
| String | 字符串 |
| Long | 长整型 |
| JSON | JSON格式字符串 |
| M | Must，表示必传，如果没有表示非必传 |

## 2.3数据输入格式规范

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名称 | 类型 | 必传 | 说明 |
| prodId | String | M | 合作机构金融产品系统编号  （风控平台定义提供） |
| prodNode | String | M | 金融产品业务节点 |
| data | String | M | 经过des加密后的业务数据 |
| version | String | M | API协议版本，默认值：v1 |
|  |  |  |  |

## 2.4数据返回格式规范

|  |  |  |
| --- | --- | --- |
| 参数名称 | 类型 | 说明 |
| status | Integer | 返回状态码 |
| msg | String | 返回提示信息 |
| data | String | 经过des加密后的业务数据 |
| internalErrorCode | String | 内部错误码 |

# 3. 接口总体说明

## 3.1获取商户的调用密钥接口

请求方式: REST接口，GET请求

接口URL：AppPath:30062/eaglehorn-application/api/v1/credit/company/key

AppPath参看[2.1定义](#_2.1_接口URL)

**请求参数**：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 字段名 | 必填 | 类型 | 说明 |
| customerId | 客户ID | M | String | 风控服务平台为客户分配的id |
| customerProdId | 客户产品ID | M | String | 商户的某个产品ID |

**返回值**：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 字段名 | 必填 | 类型 | 说明 |
| status | 状态码 | M | Integer | 200: 正常  非200: 异常 |
| msg | 返回信息 | M | String | 失败时，回执异常信息；  成功时，回执success |
| internalErrorCode | 内部错误码 | M | String | 正常：0 不正常：类似2001001的错误码 |
| data | encryptKey | M | String | 平台为客户分配的秘钥 |
|  |  |  |  |

**注意：此接口调用会修改该商户的秘钥，影响该商户所有的产品的节点调用，所以请不要频繁调用此接口**

## 3.2获取灵芝分接口

请求方式: REST接口，POST请求

接口URL：AppPath:30062/eaglehorn-application/api/v1/lzf/report

AppPath参看[2.1定义](#_2.1_接口URL)

**请求参数**：以下参数作为Http请求的RequestBody的参数（不要用表单提交）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 字段名 | 必填 | 类型 | 说明 |
| customerId | 商户Id | M | String | 商户ID |
| customerProdId | 商户产品Id | M | String | 商户产品ID |
| name | 姓名 | M | String | 姓名 |
| mobile | 手机号 | M | String | 手机号 |
| idCardNo | 身份证号 | M | String | 身份证号 |
| timestamp | 时间戳 | M | Long | 调用接口时的时间戳，ms级别 |
| sign | 签名信息 | M | String | 前面所有字段按顺序做Des加密后的值 |

示例如下：

如果customerId = CTXD1804281050, customerProdId = PROD1805123065825020054, name=张，mobile=13344445555，idCardNo = 330106200010102420, timestamp=1500947912936, 则sign是根据“customerId= CTXD1804281050 &customerProdId= PROD1805123065825020054 &name=张&mobile=13344445555&idCardNo=330106200010102420&timestamp=1500947912936”进行Des加密生成的，具体加密算法参看：[第5章](#_加密算法（Java）)

**返回值**：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 字段名 | 必填 | 类型 | 说明 |
| status | 状态码 | M | Integer | 200: 正常  非200: 异常 |
| msg | 返回信息 | M | String | 失败时，回执异常信息；  成功时，回执SUCCESS |
| internalErrorCode | 内部错误码 | M | String | 正常：0 不正常：类似2001001的错误码 |
| data | 报告数据 | M | 字符串 | 包含三要素信息和灵芝分字段的加密字符串，需要商户用自己的key解密 |

解密data后的内容为：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 字段名 | 必填 | 类型 | 说明 |
| name | 名字 | M | String | 姓名 |
| mobile | 手机号 | M | String | 手机号 |
| idCardNo | 身份证号 | M | String | 身份证号 |
| score | 灵芝分 |  | String | 正常返回区间为300至900，返回-1表示未命中，没有值表示服务器异常 |

# ErrorCode定义

|  |  |
| --- | --- |
| InternelErrorCode | Msg |
| 9002001 | 系统异常 |
| 2002001 | 服务异常 |
| 2002002 | 消息队列服务异常 |
| 2002103 | 参数xxx不正确 |
| 2002204 | 该商户信息不存在 |
| 2002205 | 商户姓名与商户ID不匹配，您无权获取该商户的密钥信息 |
| 2002206 | 用户三要素信息保存失败 |
| 2002207 | 请求信息保存失败，请稍后重试。 |
| 2002208 | 认证失败, 请检查参数是否正确 |
| 2002209 | ip地址参数非法 |
| 2002210 | 目前还没有生成相关报告，请稍后查询！ |
| 2002211 | 未查询到该请求信息，请确认id或者requestCustomerId参数是否正确 |
| 2002212 | 该请求的时间戳非法或已失效 |
| 2002213 | id或者customerRequestId必须传一个 |
| 2002214 | 发送消息到消息队列失败 |
| 2002215 | 无法生成报告，请联系孚临技术人员 |
| 2002216 | 该节点不存在或者未开启 |
| 2002217 | 该商户已停用 |
| 2002218 | 该商户已注销 |
| 2002219 | 该产品已停用 |
| 2002220 | 该产品已注销 |
| 其他 | 通常是http调用的问题 |

# 加密算法（Java）

1. import javax.crypto.Cipher;  
   import javax.crypto.SecretKey;  
   import javax.crypto.SecretKeyFactory;  
   import javax.crypto.spec.DESKeySpec;  
   import javax.crypto.spec.IvParameterSpec;  
     
   /\*\*  
    \* DES CBC 加密、解密算法  
    \*   
    \* @author xiaoliang.chen  
    \* @version $Id: DesCbcSecurity.java, v 0.1 2017年12月16日 下午12:55:29 xiaoliang.chen Exp $  
    \*/  
   public class DesCbcSecurity {  
     
    public static final String ***SECRET\_DES*** = "DES";  
    public static final String ***ALGORITHM\_DES*** = "DES/CBC/PKCS5Padding";  
     
    public static void main(String[] args) throws Exception {  
      
    String key = "IKo4FGe1";  
    String bbbString = "customerId=HJGS1806081013&customerProdId=PROD1806159021508163743&name=吴仁彪&mobile=1582716758&idCardNo=42092319910204529&timestamp=1528527568482";  
     
    System.***out***.println("最终加密后： " + encrypt(bbbString, key));  
    }  
    /\*\*  
    \* 加密  
    \* @author xiaoliang.chen  
    \* 2017年12月16日 下午12:59:28  
    \* @param content  
    \* @param key  
    \* @return  
    \*/  
    public static String encrypt(String content, String key) {  
    return byteToHexString(encrypt(content.getBytes(), key.getBytes()));  
    }  
     
    public static byte[] encrypt(byte[] content, byte[] keyBytes) {  
    try {  
    DESKeySpec keySpec = new DESKeySpec(keyBytes);  
    SecretKeyFactory keyFactory = SecretKeyFactory.getInstance(***SECRET\_DES***);  
    SecretKey key = keyFactory.generateSecret(keySpec);  
     
    Cipher cipher = Cipher.getInstance(***ALGORITHM\_DES***);  
    cipher.init(Cipher.***ENCRYPT\_MODE***, key, new IvParameterSpec(keySpec.getKey()));  
    byte[] result = cipher.doFinal(content);  
    return result;  
    } catch (Exception e) {  
    e.printStackTrace();  
    }  
    return null;  
    }  
    /\*\*  
    \* 解密  
    \* @author xiaoliang.chen  
    \* 2017年12月16日 下午1:01:01  
    \* @param content  
    \* @param key  
    \* @return  
    \*/  
    public static String decrypt(String content, String key) {  
    byte[] contentBytes = hexStringToBytes(content);  
    return decrypt(contentBytes, key.getBytes());  
    }  
      
    public static String decrypt(byte[] content, byte[] keyBytes) {  
    try {  
    DESKeySpec keySpec = new DESKeySpec(keyBytes);  
    SecretKeyFactory keyFactory = SecretKeyFactory.getInstance(***SECRET\_DES***);  
    SecretKey key = keyFactory.generateSecret(keySpec);  
     
    Cipher cipher = Cipher.getInstance(***ALGORITHM\_DES***);  
    cipher.init(Cipher.***DECRYPT\_MODE***, key, new IvParameterSpec(keyBytes));  
    byte[] result = cipher.doFinal(content);  
    String contentString = new String(result);  
    return contentString;  
    } catch (Exception e) {  
    e.printStackTrace();  
    }  
    return null;  
    }  
     
    public static String byteToHexString(byte[] bytes) {  
    StringBuffer sb = new StringBuffer(bytes.length);  
    String sTemp;  
    for (int i = 0; i < bytes.length; i++) {  
    sTemp = Integer.toHexString(0xFF & bytes[i]);  
    if (sTemp.length() < 2)  
    sb.append(0);  
    sb.append(sTemp.toUpperCase());  
    }  
    return sb.toString();  
    }  
      
    public static byte[] hexStringToBytes(String hexString) {  
    if (hexString == null || hexString.equals("")) {  
    return null;  
    }  
    hexString = hexString.toUpperCase();  
    int length = hexString.length() / 2;  
    char[] hexChars = hexString.toCharArray();  
    byte[] d = new byte[length];  
    for (int i = 0; i < length; i++) {  
    int pos = i \* 2;  
    d[i] = (byte) (charToByte(hexChars[pos]) << 4 | charToByte(hexChars[pos + 1]));  
     
    }  
    return d;  
    }  
     
    private static byte charToByte(char c) {  
    return (byte) "0123456789ABCDEF".indexOf(c);  
    }  
   }

# 6. Http Post调用范例(Java)

1. import com.alibaba.fastjson.JSON;  
   import org.apache.commons.collections.map.HashedMap;  
     
   import java.io.\*;  
   import java.net.HttpURLConnection;  
   import java.net.URL;  
   import java.net.URLConnection;  
   import java.util.Map;  
     
   /\*\*  
    \* @Description:  
    \* @Author: xiaoliang.chen  
    \* @Date: Created in 下午5:26 2018/6/11  
    \* @Modified  
    \*/  
   public class MyHttpUtil {  
     
     
    /\*\*  
    \* HTTP Post 获取内容  
    \* @return 页面内容  
    \*/  
    public static String doPost(String url, String param) throws Exception {  
    URL localURL = new URL(url);  
    URLConnection connection = localURL.openConnection();  
    HttpURLConnection httpURLConnection = (HttpURLConnection) connection;  
    httpURLConnection.setDoOutput(true);  
    httpURLConnection.setRequestMethod("POST");  
    httpURLConnection.setRequestProperty("Accept-Charset", "utf-8");  
    httpURLConnection.setRequestProperty("Content-Type", "application/json");  
    httpURLConnection.setRequestProperty("Content-Length", String.valueOf(param.length()));  
    httpURLConnection.setReadTimeout(60000);// 读取数据超时时间  
    httpURLConnection.setConnectTimeout(60000);// 连接主机超时时间  
    OutputStream outputStream = null;  
    OutputStreamWriter outputStreamWriter = null;  
    InputStream inputStream = null;  
    InputStreamReader inputStreamReader = null;  
    BufferedReader reader = null;  
    StringBuffer resultBuffer = new StringBuffer();  
    String tempLine = null;  
     
    try {  
    outputStream = httpURLConnection.getOutputStream();  
    outputStreamWriter = new OutputStreamWriter(outputStream, "UTF-8");  
    outputStreamWriter.write(param.toString());  
    outputStreamWriter.flush();  
    if (httpURLConnection.getResponseCode() >= 300) {  
    throw new Exception(  
    "HTTP Request is not success, Response code is " + httpURLConnection.getResponseCode());  
    }  
     
    inputStream = httpURLConnection.getInputStream();  
    inputStreamReader = new InputStreamReader(inputStream, "UTF-8");  
    reader = new BufferedReader(inputStreamReader);  
     
    while ((tempLine = reader.readLine()) != null) {  
    resultBuffer.append(tempLine);  
    }  
     
    } finally {  
     
    if (outputStreamWriter != null) {  
    outputStreamWriter.close();  
    }  
     
    if (outputStream != null) {  
    outputStream.close();  
    }  
     
    if (reader != null) {  
    reader.close();  
    }  
     
    if (inputStreamReader != null) {  
    inputStreamReader.close();  
    }  
     
    if (inputStream != null) {  
    inputStream.close();  
    }  
    }  
    return resultBuffer.toString();  
    }  
     
    public static void main(String[] args) throws Exception {  
    String url ="http://testd.fltechfin.com:30062/eaglehorn-application/api/v1/lzf/report";  
    Map<String, Object> params = new HashedMap();  
    params.put("customerId", "DSFZF1805161114");  
    params.put("customerProdId", "PROD1805136031548334892");  
    params.put("mobile", "15927236181");  
    params.put("name", "毛俊邱");  
    params.put("idCardNo", "421022199505210053");  
    params.put("timestamp", 1513389389346L);  
    params.put("sign", "D5A2310B92F911220878D21FDFAF5B4204475D7958AAA24DE3CD658C95EBD195E300072F0063BBD24FBBBD20EDBC7837C590AF1BAFE1850C9446075F1C7B1830EC6D66EE92016203938940B31A986F3880B0F431F132973FD1859EB645428A0BB91FAC453F159CDE0BCF9EA9F58EE6D3E97AEC9F30F334071F977667175FD4538DC9CCB6C2CC92F8B482FE31362F28AC57F30801D9B38D21");  
     
     
    System.***out***.println(doPost(url, JSON.toJSONString(params)));  
    }

}

# Jar包调用范例

jar包与文档一并给出、调用jar包的demo如下：

import com.alibaba.fastjson.JSON;  
import org.apache.commons.collections.map.HashedMap;  
import tech.fullink.lzf.common.utils.DesCbcSecurity;  
import tech.fullink.lzf.common.utils.MyHttpUtil;  
  
import java.util.Map;  
  
/\*\*  
 \* lilong.luo  
 \* 2018.7.18  
 \*/  
public class Main {  
 public static void main(String[] args) throws Exception {  
 //加密的key  
 String key = "WQ1s7rce";  
 //商户id  
 String customerId = "HJGS1806081013";  
 //产品id  
 String customerProdId = "PROD1806159021508163743";  
 //手机号、姓名、身份证号、时间戳  
 String mobile = "1582716758";  
 String name = "吴仁彪";  
 String idCardNo = "42092319910204529";  
 long timestamp = 1528527568482L;  
  
 //加密  
 String bbbString = "customerId=" + customerId +  
 "&customerProdId="+ customerProdId +  
 "&name=" + name +  
 "&mobile=" + mobile +  
 "&idCardNo=" + idCardNo +  
 "&timestamp="+ timestamp;  
 //签名  
 String sign = DesCbcSecurity.encrypt(bbbString, key);  
 System.out.println("最终加密后： " + DesCbcSecurity.encrypt(bbbString, key));  
  
 //调用地址，见文档  
 String url = "http://saas-test.fullink.tech:30062/eaglehorn-application/api/v1/lzf/report";  
  
 Map<String, Object> params = new HashedMap();  
 params.put("customerId", customerId);  
 params.put("customerProdId", customerProdId);  
 params.put("mobile", mobile);  
 params.put("name", name);  
 params.put("idCardNo", idCardNo);  
 params.put("timestamp", timestamp);  
 params.put("sign", sign);  
  
 //http调用  
 System.out.println("调用结果：" + MyHttpUtil.doPost(url, JSON.toJSONString(params)));  
  
 }  
}

附件：jar包