推呗标准接口 api

变更记录

版本	变更内容	变更日期
V1.1	初始版本	2023.05.19
V1.2	增加拓展参数	2023.07.05
V1.3	增加民生人工处理状态	2023.09.06

对接说明

1. 数据提交/接收方式: application/x-www-form-urlencoded
2. 正式域名: https://tuibei-mng.duiba.com.cn
3. 测试域名: https://tuibei-mng.duibatest.com.cn
4. 配置字段:

tid: 合作机构 id，推呗提供
salt: 合作机构加密盐值，推呗提供
cardCode: 合作机构卡片编号，推呗提供
callbackUrl：合作机构接口回调地址，合作机构提供

5. 回调接口：测试/生产环境对接前同步给运营配置
6. 参数处理：所有接口参数均需按如下步骤进行处理，可参照参数处理的实例参考
    0. 参数整理为键值数组格式(参数中某些敏感信息可能需要单独加密，可以参考具体接口的要求)。
    0. 对参数进行签名，参数的 token 用来存放签名的值。见附件签名算法
    0. 对参数进行 json 转换为字符串。
    0. 对 3 步骤的字符串转为十六进制值。
    0. 对 4 步骤的字符串使用 base64 加密。
    0. 对 5 步骤的字符串作为 data 参数的值，POST 提交数据。

相关接口

1. 获取银行申卡链接 url 接口

接口地址：

/downstream/api/getCardLink，请求方式：POST

接口入参：

参数	类型	长度	是否必传	备注
tid	string	50	是	第三方平台 id，对接后由接口提供方分配
timestamp	long	14	是	时间戳（毫秒）

cardCode	string	50	是	卡片编号，对接后由接口提供方分配
orderId	string	50	是	第三方订单流水号，唯一标识
mobile	int	11	否	手机号
name	string	50	否	申卡用户姓名，是否必填根据银行产品要求确定
idNumber	string	32	否	身份证号，是否必填根据银行产品要求确定
token	string	32	是	签名参数值，由接口提供方约定算法生成。第三方对接需要自己实现 见附件签名算法

参数处理过程示例

1. 初始参数

```
{
    "tid": "9999",
    "cardCode": "testCode",
    "orderId": "d2jhf348gh7jf",
    "callbackUrl": "https://www.baidu.com",
    "timestamp": 1683621431742
}
```

2. 签名结果：E83E2B94E09D979B571986BCE1FF161D
3. 加上签名的参数

```
{
    "tid": "9999",
    "cardCode": "testCode",
    "orderId": "d2jhf348gh7jf",
    "callbackUrl": "https://www.baidu.com",
    "timestamp": 1683621431742,
    "token": "E83E2B94E09D979B571986BCE1FF161D"
}
```

4. 转换成 json 字符串

```
{"tid":"9999","cardCode":"testCode","orderId":"d2jhf348gh7jf","callbackUrl":"https://www.baidu.com","timestamp":1683621431742,"token":"E83E2B94E09D979B571986BCE1FF161D"}
```

5. 转 16 进制字符串

```
7b22746964223a2239393939222c2263617264436f6465223a2274657374436f6465222c226f726465724964223a2264326a686633343867683736a66222c2263616c6c6261636b55726c223a2268747470733a2f2f7777772e62616964752e636f6d222c2274696d657374616d70223a31363833336323134333313734322c22746f6b656e223a22453833453242393445303944393739423537313938364243453146463136314227d
```

6. base64 处理

```
N2IyMjc0Njk2NDIyM2EyMjM5MzkzOTM5MjIyYzIyNjM2MTcyNjQ0MzZmNjQ2NTIyM2EyMjc0NjU3Mzc0NDM2ZjY0NjUyMjJjMjI2ZjcyNjQ2NTcyNDk2NDIyM2EyMjY0MzI2YTY4NjYzMzM0Mzg2NzY4Mzc2YTY2MjIyYzIyNjM2MTZjNmM2MjYxNjM2YjU1NzI2YzIyM2EyMjY4NzQ3NDcwNzMzYTJmMmY3Nzc3NzcyZTYyNjE2OTY0NzUyZTYzNmY2ZDIyMmMyMjc0Njk2ZDY1NzM3NDYxNmQ3MDIyM2EzMTM2MzgzMzM2MzIzMTM0MzMzMTM3MzQzMjJjMjI3NDZmNmI2NTZlMjIzYTIyNDUzODMzNDUzMjQyMzkzNDQ1MzAzOTQ0MzkzNzM5NDIzNTM3MzEzOTM4MzY0MjQzNDUzMTQ2NDYzMTM2MzE0NDIyN2Q
```

7. 将处理后的字符串作为 data 的值进行 post 提交

```
{{host}}/downstream/api/getCardLink?data=N2IyMjc0Njk2NDIyM2EyMjM5MzkzOTM5MjIyYzIyNjM2MTcyNjQ0MzZmNjQ2NTIyM2EyMjc0NjU3Mzc0NDM2ZjY0NjUyMjJjMjI2ZjcyNjQ2NTcyNDk2NDIyM2EyMjY0MzI2YTY4NjYzMzM0Mzg2NzY4Mzc2YTY2MjIyYzIyNjM2MTZjNmM2MjYxNjM2YjU1NzI2YzIyM2EyMjY4NzQ3NDcwNzMzYTJmMmY3Nzc3NzcyZTYyNjE2OTY0NzUyZTYzNmY2ZDIyMmMyMjc0Njk2ZDY1NzM3NDYxNmQ3MDIyM2EzMTM2MzgzMzM2MzIzMTM0MzMzMTM3MzQzMjJjMjI3NDZmNmI2NTZlMjIzYTIyNDUzODMzNDUzMjQyMzkzNDQ1MzAzOTQ0MzkzNzM5NDIzNTM3MzEzOTM4MzY0MjQzNDUzMTQ2NDYzMTM2MzE0NDIyN2Q
```

MTM3MzQzMjJjMjI3NDZmNmI2NTZlMjIzYTIyNDUzODMzNDUzMjQyMzkzNDQ1MzAzOTQ0MzkzNzM5NDIzNTM3MzEzOTM4MzY0MjQzNDUzMTQ2NDYzMTM2MzE0NDIyN2Q

返回参数：

名称	类型	是否必须	备注
success	boolean	非必须	是否成功
code	string	非必须	响应码
desc	string	非必须	错误描述
timestamp	integer	非必须	服务器当前时间戳
data	string	非必须	响应数据

返回示例

```
{
   "success": true,
   "code": 0000000000,
   "desc": "",
   "timestamp": 68256694,
   "data":
"https:\/\/xyk.cebbank.com\/icip\/icip-applypage\/info1?pro_code=FHTG023512SD0587CJSH&cardId=20960&coopinfo=49393259a1
   2c9e0677f5dc014329f7bfba8a88d4d1fb30e31799d08b09178d0c&corp_id=RONG20"
}
```

2. 用户订单查询接口

接口地址：

/downstream/api/findOrderInfo，请求方式：POST

接口入参：

参数名称	类型	是否必须	备注
tid	string	是	第三方平台 id，对接后由接口提供方分配
timestamp	long	是	时间戳（毫秒）
cardCodes[]	array[string]	否	卡片编号列表
orderId	string	否	第三方订单流水号，如果传该值，则只查询该订单信息
earliest	long	否	返回数据申请时间的最早时间戳，默认返回 1 个月之内的数据
token	string	是	签名参数值，由接口提供方约定算法生成。第三方对接需要自己实现 见附件签名算法

pageNum	int	否	分页数，默认 1
pageSize	int	否	分页大小，默认 20

参数处理过程示例

1. 初始参数

```
{
    "tid": "9999",
    "timestamp": 1683621804121,
    "cardCodes": [
        "testCode",
        "testCode1"
    ],
    "earliest": 1681029804121,
    "pageNum": 1,
    "pageSize": 20
}
```

2. 签名结果：D0C8D3D3C1BF187977D97D86FFA25895

3. 加上签名的参数

```
{
    "tid": "9999",
    "timestamp": 1683621804121,
    "cardCodes": [
        "testCode",
        "testCode1"
    ],
    "earliest": 1681029804121,
    "pageNum": 1,
    "pageSize": 20,
    "token": "D0C8D3D3C1BF187977D97D86FFA25895"
}
```

4. 转换成 json 字符串

```
{"tid":"9999","timestamp":1683621804121,"cardCodes":["testCode","testCode1"],"earliest":1681029804121,"pageNum":1,"pageSize":20,"token":"D0C8D3D3C1BF187977D97D86FFA25895"}
```

5. 转 16 进制字符串

```
7b22746964223a2239393939222c2274696d657374616d70223a313638333632313830343132312c2263617264436f646573223a5b2274657374436f6465222c2274657374436f646531225d2c226561726c69657374223a31363831303239383034313231 2c22706167654e756d223a312c227061676553697a65223a32302c22746f6b656e223a2244304338443344334331424631383739373744393744383646464132353839352 27d
```

6. base64 处理

```
N2IyMjc0Njk2NDIyM2EyMjM5MzkzOTM5MjIyYzIyNzQ2OTZkNjU3Mzc0NjE2ZDcwMjIzYTMxMzYzODMzMzYzMjMxMzgzMDM0MzEzMjMxMmMyMjYzNjE3MjY0NDM2ZjY0NjU3MzIyM2E1YjIyNzQ2NTczNzQ0MzZmNjQ2NTIyMmMyMjc0NjU3Mzc0NDM2ZjY0NjUzMTIyNWQyYzIyNjU2MTcyNmM2OTY1NzM3NDIyM2EzMTM2MzgzMTMwMzIzOTM4MzAzNDMxMzIzMTJjMjI3MDYxNjc2NTRlNzU2ZDIyM2EzMTJjMjI3MDYxNjc2NTUzNjk3YTY1MjIzYTMyMzAyYzIyNzQ2ZjZiNjU2ZTIyM2EyMjQ0MzA0MzM4NDQzMzQ0MzM0MzMxNDI0NjMxMzgzNzM5MzczNzQ0MzkzNzQ0MzgzNjQ2NDY0MTMyMzUzODM5MzUyMjdk
```

7. 将处理后的字符串作为 data 的值进行 post 提交

```
{{host}}/downstream/api/findOrderInfo?data=N2IyMjc0Njk2NDIyM2EyMjM5MzkzOTM5MjIyYzIyNjM2MTcyNjQ0MzZmNjQ2NTIyM2EyMjc0NjU3Mzc0NDM2ZjY0NjUyMjJjMjI2ZjcyNjQ2NTcyNDk2NDIyM2EyMjY0MzI2YTY4NjYzMzM0Mzg2NzY4Mzc2YTY2MjIyYzIyNjM2MTZjNmM2MjYxNjM2YjU1NzI2YzIyM2EyMjY4NzQ3NDcwNzMzYTJmMmY3Nzc3NzcyZTYyNjE2OTY0NzUyZTYzNmY2ZDIyMmMyMjc0Njk2ZDY1NzM3NDYxNmQ3MDIyM2EzMTM2MzgzMzM2MzIzMTM0MzMzMTM3MzQzMjJjMjI3NDZmNmI2NTZlMjIzYTIyNDUzODMzNDUzMjQyMzkzNDQ1MzAzOTQ0MzkzNzM5NDIzNTM3MzEzOTM4MzY0MjQzNDUzMTQ2NDYzMTM2MzE0NDIyN2Q
```

返回参数：

名称	类型	备注	其他信息
success	boolean	是否成功	
code	string	响应码	
desc	string	错误描述	
timestamp	long	服务器当前时间戳	
data	object	响应数据	**备注:** 响应数据
curPage	integer	当前页	
size	integer	数量	
totalPage	integer	总页数	
total	integer	总数量	
data	object []	数据集合	**item 类型:** object
tid	string	第三方平台 id	
cardCode	string	卡片编号	
orderId	string	第三方订单流水号，唯一标识	
cardType	string	卡种	
name	string	申卡用户姓名	
mobile	string	手机号	
idNumber	string	身份证号	
gmtApply	string	进件日期	
gmtExamine	string	核卡日期	
gmtActivation	string	激活日期	
gmtFirstBrush	string	首刷日期	
approvalStatus	string	审批状态（见附件状态定义）	
twoCard	int	一二卡（0 一卡 1 二卡 2 不符合资质卡）	
gmtCreate	string	创建时间	
extraData	object []	拓展参数集合	

monthCancelFlag	string	月注销标识，1：是； 0：否	
highQualityCustomer	string	优质客户，1：是； 0:否	
firstInstance	string	初审 A-初审通过 D-初审拒绝 M-人工处理	

返回示例


```
{
    "code": "0000000000",
    "data": {
        "curPage": 1,
        "data": [
            {
                "approvalStatus": "0",
                "cardType": "测试",
                "gmtActivation": "2023-11-11",
                "gmtApply": "2023-01-05",
                "gmtCreate": "2023-04-06 10:34:54",
                "gmtExamine": "2023-01-10",
                "gmtFirstBrush": "2023-12-21",
                "idNumber": "41032318889898",
                "ifNewUser": "0",
                "mobile": "15136361301",
                "name": "test",
                "orderId": "1",
                "tid": "testzhuanyong"
            },
            {
                "approvalStatus": "2",
                "cardCode": "e0s7e5eo3",
                "gmtActivation": "2023-04-27",
                "gmtApply": "2023-04-27",
                "gmtCreate": "2023-04-27 11:38:00",
                "gmtExamine": "2023-04-27",
                "mobile": "18989849377",
                "orderId": "db825658644812e6f13b",
                "tid": "ceshiliucheng03"
            }
        ],
        "size": 20,
        "total": 8,
        "totalPage": 1
    },
    "desc": "OK",
    "success": true,
    "timestamp": 1683626137591
}
```

3. 订单推送

接口说明：

- 合作机构需要提供回调地址
- 平台不定时推送订单状态到回调地址
- 合作机构需返回是否成功接收信息
- 主动推送与订单查询接口互不干扰
- 请求失败时自动重试 3 次，每次间隔 1.5s

请求方式：POST

接口入参：

参数名称	是否必须	备注
uuid	是	一次请求的唯一标识，接收方需做好幂等校验
timestamp	是	时间戳
token	是	签名 token (对 data 中的数据进行签名，见附件签名算法)
data	是	订单数据（使用参数处理说明的逆向处理可转换成 json 对象）

订单数据解析后参数：

参数名称	是否必须	备注
tid	是	第三方平台 id
cardCode	是	卡片编号
orderId	是	第三方订单流水号，唯一标识
cardType	否	卡种
name	否	姓名
mobile	否	手机号
idNumber	否	身份证号
gmtApply	否	进件日期
gmtExamine	否	核卡日期
gmtActivation	否	激活日期
gmtFirstBrush	否	首刷日期
approvalStatus	是	审批状态（见附件状态定义）

twoCard	否	一二卡（0 一卡 1 二卡 2 不符合资质卡）
gmtCreate	是	创建时间
extraData	否	拓展参数

拓展参数(extraData)：

名称	类型	是否必须	备注
monthCancelFlag	string	否	月注销标识，1：是； 0：否
highQualityCustomer	string	否	优质客户，1：是； 0:否
firstInstance	string	否	初审 A：初审通过 D：初审拒绝 M-人工处理

响应参数：

名称	类型	是否必须	备注
success	boolean	非必须	请求是否成功
code	string	非必须	业务响应码（成功返回 000000）
desc	string	非必须	错误描述

附件

状态定义

状态码（approvalStatus）	描述	说明
0	待进件	用户获取卡片链接，未提交申卡信息
1	待审核	用户填写申卡信息并提交-已经进件
2	已审核	银行审核通过
3	已激活	卡片已激活
4	已首刷	用户完成第一次消费
5	审核失败	银行审核未通过

签名算法

```
import com.alibaba.fastjson.JSON;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.lang.StringUtils;

import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Map;
import java.util.Objects;
import java.util.TreeMap;

/**
 * 签名工具类
 *
 * @author ykb
 * @date 2023-05-09
 */
@Slf4j
public class SignUtils {

    /**
     * 签名
     *
     * @param params 参数
     * @param salt   盐
     * @return {@link String}
     */
    public static String sign(Map<String, Object> params, String salt){
        params.entrySet().removeIf(entry -> Objects.isNull(entry.getValue()) ||
StringUtils.isEmpty(entry.getValue().toString()));
        Map<String, Object> noArraySortParams = sortParams(params);
        String noArraySignStr = joinParams(noArraySortParams);
        String noArrayToken = md5(md5(noArraySignStr) + salt);
        log.info("签名参数：{}，签名结果：{}", noArraySignStr, noArrayToken.toUpperCase());
        return noArrayToken.toUpperCase();
    }

    /**
     * MD5 加密
     *
     * @param plainText 文本
     * @return {@link String}
     */
    public static String md5(String plainText) {

        //定义一个字节数组
        byte[] secretBytes;

        try {

            // 生成一个 MD5 加密计算摘要
            MessageDigest md = MessageDigest.getInstance("MD5");
```

```
            //对字符串进行加密
            md.update(plainText.getBytes());

            //获得加密后的数据
            secretBytes = md.digest();

        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException("没有 md5 这个算法！ ");
        }

        //将加密后的数据转换为 16 进制数字
        StringBuilder md5code = new StringBuilder(new BigInteger(1, secretBytes).toString(16));

        // 如果生成数字未满 32 位，需要前面补 0
        int len = md5code.length();
        for (int i = 0; i < 32 - len; i++) {
            md5code.insert(0, "0");
        }

        return md5code.toString();
    }

    /**
     * 参数排序
     *
     * @param params 参数
     * @return {@link Map}<{@link String}, {@link Object}>
     */
    public static Map<String, Object> sortParams(Map<String, Object> params) {

        if (params == null || params.isEmpty()) {
            return null;
        }

        Map<String, Object> sortMap = new TreeMap<>(String::compareTo);
        sortMap.putAll(params);

        return sortMap;
    }

    /**
     * 获取 key 和值的拼接结果
     * 如果 value 为空，则 key 也不参与拼接
     *
     * @param params 参数
     * @return {@link String}
     */
    public static String joinParams(Map<String, Object> params){

        StringBuilder str = new StringBuilder();

        for (Map.Entry<String, Object> entry : params.entrySet()) {

            if (entry == null){
```

```
                continue;
            }

            if ("token".equals(entry.getKey())){
                continue;
            }

            if (entry.getValue().toString().length() > 0 || StringUtils.isNumeric(entry.getValue().toString())) {

                if (entry.getValue().getClass().equals(String[].class)) {
                    str.append(entry.getKey()).append(JSON.toJSONString(entry.getValue()));
                } else if(entry.getValue().getClass().equals(LocalDateTime.class)){
                    LocalDateTime value = (LocalDateTime) entry.getValue();
                    DateTimeFormatter dateTimeFormatter =
DateTimeFormatter.ofPattern("yyyy-MM-dd'T'HH:mm:ss");
                    str.append(entry.getKey()).append(dateTimeFormatter.format(value));
                } else {
                    str.append(entry.getKey()).append(entry.getValue());
                }
            }
        }
        return str.toString();
    }

  /**
     * 字符串转化成为 16 进制字符串
     * @param s
     * @return
     */
    public static String strTo16(String s) {
        byte[] bytes = s.getBytes(StandardCharsets.UTF_8);
        StringBuilder hex = new StringBuilder();
        for (byte b : bytes) {
            hex.append(String.format("%02x", b));
        }
        return hex.toString();
    }

    /**
     * 十六进制字符串转换成字符串
     *
     * @param s
     * @return {@link String}
     */
    public static String hexStringToString(String s) {
        if (s == null || "".equals(s)) {
            return null;
        }
        s = s.replace(" ", "");
        byte[] baKeyword = new byte[s.length() / 2];
        for (int i = 0; i < baKeyword.length; i++) {
            try {
                baKeyword[i] = (byte) (0xff & Integer.parseInt(s.substring(i * 2, i * 2 + 2), 16));
            } catch (Exception e) {
                e.printStackTrace();
```

```
                }
            }
            try {
                s = new String(baKeyword, StandardCharsets.UTF_8);
            } catch (Exception e1) {
                e1.printStackTrace();
            }
            return s;
    }
```

错误码

错误码	错误提示	错误原因和解决办法
100201	渠道产品不存在	联系运营核对 cardCode 值
100202	渠道不存在	联系运营核对 tid 值
100203	获取卡产品链接失败	请求卡产品链接失败，联系开发人员排查
100204	签名验证失败	token 生成方式有误
100205	请求已过期	请求时间超过 10 分钟，或请求时间在当前时间之后
100206	查询数量超出限制	订单查询接口数据量在 1000 以内
100000	系统错误	系统错误，联系开发人员排查