腾梭身份两要素接口文档

1. 接口公共说明

约定

- 所有数据都以 JSON 格式传递
- 所有数据都走 http/https 接口，调试环境为http，正式环境为https
- 字符编码默认用 UTF-8

调试

- 申请并妥善保管您的secretId和secretKey
- 调试地址：http://118.126.96.200:8801

调用方式

协议

> *HTTP/HTTPS*

接口定义

POST 有效 RAW

```
/{productCode}/request
```

接口描述

地址参数

参数名	描述
productCode	产品编码，请查询实际购买的产品指定

请求头

名称	类型	必填	默认值	描述
X-TS-Key	string	true	true	请求标识，建议使用唯一值。32位长度
X-TS-Api	string	true	true	接口名称，请查询实际购买的产品指定
X-TS-Timestamp	string	true	true	当前东八区毫秒时间戳，可记录发起 API 请求的时间。注意：如果与服务器时间相差超过5分钟，会引起签名过期错误。
Authorization	string	true	true	HTTP 标准身份认证头部字段，详见"签名方式"

请求体

```
** json格式，根据具体产品产品传入
```

响应参数(Body)

200: OK　　响应数据格式：JSON

公共响应参数，不同的接口还存在其他参数。

名称	类型	必填	默认值	描述
code	int32	true		响应码，0标识成功。其他值见错误码列表
codeDesc	string	false		响应提示信息
message	string	false		错误码信息描述

签名方式

说明

> *为了防止请求合法性和数据有效性，需要在请求头中添加认证数据，包含请求方，请求签名。*

认证头格式设计

```
MD5 Credential=${secretId},Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f10
```

格式说明

1. MD5
   > *签名方法，目前固定取该值*

2. Credential
   > *签名凭证，${secretId}是用户在平台分配的secretId*

3. Signature
   > *签名摘要*

签名摘要计算步骤说明

1. 待签名字符串组装

   ```
   ${productNo}${X-TS-Key}${X-TS-Api}${X-TS-Timestamp}${secretKey}${requestBody}
   ```

   a. productNo
      产品编码
   b. X-TS-Key
      请求标识
   c. X-TS-Api
      接口名称
   d. X-TS-Timestamp
      当前东八区毫秒时间戳。
   e. secretKey
      用户的secretKey,
   f. requestBody
      请求体
2. 签名
   在完成第一步后，对生成的字符串进行签名:
   `Lowercase(HexEncode(MD5(stringToSign)))`
3. 示例(JavaScript)

```
// 引入md5函数 <script src="https://cdn.bootcss.com/blueimp-md5/2.10.0/js/md5.js"></script>
function md5Str() {
        let productCode = 'anti';
        let requestKey= '1629373888664';
        let apiCode= 'anti-api-v1';
        let timestamp= '1629373888664';
        let secretKey= 'your secret key';
        let str = productCode + requestKey + apiCode + timestamp+ secretKey
        let data = {};
        data.idNumber = '110123456789012345';
        data.phoneNumber = '13012345678';
        data.bankCardNumber= '6222020000000000000';
        let dataStr = JSON.stringify(data);
        let requestStr = str + dataStr;
        console.log(requestStr)
        let result = md5(requestStr);
        console.log(result);
}
```

4. 示例（Java）

```
package demo.sign;

import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

/**
 * 签名示例
 *
 * @author luofei
 * @date 2021/8/20 17:02
 */
class SignDemo {

    private static final char[] DIGITS_LOWER =
            {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f'};

    public static void main(String[] args) throws NoSuchAlgorithmException {
        // 产品编码
        String productCode = "anti";
        // 请求标识
        String requestKey = "1629373888664";
        // 接口编码
        String apiCode = "anti-api-v1";
        // 当前时间戳
        String timestamp = "1629373888664";
        // secretKey
        String secretKey = "your secret key";
        String strToSign = productCode + requestKey + apiCode + timestamp + secretKey;
        String idNumber = "110123456789012345";
        String phoneNumber = "13012345678";
        String bankCardNumber = "6222020000000000000";
        // 请求体信息，！！！注意：签名使用的请求体请与实际发送的请求体一致！！！
        String data =
                "{\"idNumber\":\"" + idNumber + "\",\"phoneNumber\":\"" + phoneNumber + "\",\"
                bankCardNumber + "\"}";
        // 最终待签名字符串
        strToSign = strToSign + data;
        System.out.println("strToSign:[" + strToSign + "]");
        // 签名结果
        String result = md5(strToSign);
        System.out.println("result:[" + result + "]");
    }

    private static String md5(String strToSign) throws NoSuchAlgorithmException {
        MessageDigest messageDigest = MessageDigest.getInstance("MD5");
        if (messageDigest != null) {
            byte[] bytes = strToSign.getBytes(StandardCharsets.UTF_8);
            return encodeHexString(messageDigest.digest(bytes));
        }
        throw new NoSuchAlgorithmException("MessageDigest get MD5 instance error");
    }
```

```
    private static String encodeHexString(byte[] bytes) {
        int l = bytes.length;
        char[] out = new char[l << 1];
        for (int i = 0, j = 0; i < l; i++) {
            out[j++] = DIGITS_LOWER[(0xF0 & bytes[i]) >>> 4];
            out[j++] = DIGITS_LOWER[0x0F & bytes[i]];
        }
        return new String(out);
    }
}
```

公共响应码

响应码	说明	处理措施
0	成功	
4000	参数校验失败	请检查接口非空参数是否有值。
4102、4103、4104	未获取到内部接口的配置信息	请联系管理员。
4100	签名验证失败	确认签名的生成方式是否正确。
4101	接口权限不足	可能是余额/次数不足，请联系管理员。
4500	请求已失效	签名的有效期是10分钟，过期后需要重新生成签名。
6000	系统错误	请联系管理员。

2．身份两要素接口

POST 有效 JSON

```
/factor/request
```

接口描述

> *支持校验身份二要素验证，支持明文和md5。*

请求体

名称	类型	必填	默认值	描述
idNumber	string	true		身份证号
name	string	true		姓名

响应参数(Body)

200: OK　　响应数据格式：JSON

名称	类型	必填	默认值	描述
code	int32	true		公共错误码 0：表示查询成功 其他值：表示失败
codeDesc	string	false		业务侧错误码：对 code 码的英文描述 成功时返回 Success 错误时返回具体业务错误原因
message	string	false		模块错误信息描述，与接口相关。
verifyResult	object	true		
verifyCode	string	true		验证结果码，请参考后面的说明
verifyMessage	string	true		验证结果信息

API Code说明

Code	描述	备注
IdVerify_v1	两要素验证	两要素：身份证号码，姓名
IdVerify_md5_v1	两要素验证md5	两要素：身份证号码，姓名，传md5值

验证结果码(verifyCode)说明

验证结果码	验证详情	验证结果码解释	是否计费
200	一致	提交的两要素验证一致	是
404	不一致	提交的两要素验证不一致	是
405	无效的证件号		否
405	参数异常		否
500	系统错误		否
502	不存在		否
503	无法验证		否