

自然人风险评分查询接口

接口地址: `https://tx.hz-hanghui.com:8088/hanghui-server-platform/api/v1/risk/query`

请求方式: `POST`

请求数据类型: `application/json`

响应数据类型: `*/*`

接口描述: 请求和响应报文参数均需要使用国密4进行加解密, 涉及的appId、appKey、appSecret、privateKey管理员提供

请求示例:

```
1 //国密m4加密后密文 也就是三方接口接收到的数据
2 {
3     "appId": "",
4     "bizContent":
5     "NI4ry+D01kDXrNf50cmAzMGaB0td9kXfa321xmIS3qw5homFdZT4xaJu8Vqid37NDivdqj1ADzd0+v+QDKNovFSBkm
6     OyOQt20bJ4HW43i0dKKHAK1I3FtZA0F9URhlqXbckbpGMD1hev3XS/b9uxRw+QLOnTzhuzgoxpiOXNA9rmQs3V7prpF
7     9Wpaetts2pv",
8     "reqTimestamp": 1692693665800,
9     "sign": "签名方式见文末加密算法"
10 }
11
12 //国密m4加密前 也就是三方服务解密后的数据
13 {
14     "appId": "",
15     "bizContent": {
16         "sn": "",
17         "name": "",
18         "idNumber": "",
19         "phone": ""
20     },
21     "reqTimestamp": 1692693665800,
22     "sign": "签名方式见文末加密算法"
23 }
```

请求参数:

| 参数名称 | 参数说明 | 请求类型 | 是否必须 | 数据类型 | schema |
|--------------|---|------|------|--------|--------|
| bizContent | 加密内容 | body | true | | |
| sn | 设备号 | | true | string | |
| name | 姓名 | | true | string | |
| idNumber | 身份证号 | | true | string | |
| phone | 手机号 | | true | string | |
| appld | 商户唯一标识 | body | true | string | |
| reqTimestamp | 时间戳 毫秒 | body | true | long | |
| sign | 签名md5(appKey+appSecret+timestamp) 小写32位 | body | true | string | |

响应状态:

| 状态码 | 说明 | schema |
|-------------|----|--------|
| 200 | OK | |
| 除200以外都是异常码 | | |

响应参数:

| 参数名称 | 参数说明 | 类型 | schema |
|------------------|---|----------------|----------------|
| code | 状态码 | integer(int32) | integer(int32) |
| data | 对象 | | |
| bizContent | 加密后的数据 | string | |
| result | true/false 是否命中风险 | boolean | |
| resultInfo | 结果信息 | | |
| personExpfjxCode | 风险状态 200 除200以外都是异常 | integer(int32) | integer(int32) |
| personExpfjx | 风险值: "001010" 命中结果由 6 位0 1 字符串组成 分别为 第 1 位-其他、 第 2 位-前科、 第 3 位-涉毒、 第 4 位-吸毒、 第 5 位-在逃、 第 6 位-涉案(包括在逃撤销)。 值为 1 表示命中, 0 表示未 命中 | string | |
| reqTimestamp | 时间戳毫秒 1666881956513 | long | |
| sign | 签名md5(appKey+appSecret+timestamp) 小写32位 | string | |
| msg | 返回消息 | string | |

响应示例:

```
1 //国密m4加密后密文
2 {
3   "code": 200,
4   "data": {
```

```

5         "bizContent":
"NI4ry+D01kDXrNf50cmAzMGaB0td9kXfa321xmIS3qw5homFdZT4xaJu8Vqid37NDivdqj1ADzd0+v+QDKNovFSBkm
OyOQt20bJ4HW43i0dKKHAK1I3FtZA0F9URhlqXbckbpGMD1hev3XS/b9uxRw+QLOnTzhuzgoxpiOXNA9rmQs3V7prpF
9Wpaetts2pv",
6         "reqTimestamp": 1692693665800,
7         "sign": "签名方式见文末加密算法"
8     },
9     "msg": ""
10 }
11
12 //国密m4加密前
13 {
14     "code": 200,
15     "data": {
16         "bizContent": {
17             "result": false,
18             "resultInfo": {
19                 "personFxpfxjCode": 200,
20                 "personFxpfxj": "0",
21                 "personFxpfxjDesc": "001010" //命中结果由 6 位 0|1 的字符串组成，分别为 第 1
位-其他、第 2 位-前科、第 3 位-涉毒、第 4 位-吸毒、第 5 位-在逃、第 6 位-涉案(包括在逃撤销)。值
为 1 表示命中，0 表示未命中。
22             }
23         },
24         "reqTimestamp": 1692693665800,
25         "sign": "签名方式见文末加密算法"
26     },
27     "msg": ""
28 }

```

国密4加解密算法及签名认证

```

1 // pom.xml中引入
2 <dependency>
3     <groupId>cn.hutool</groupId>
4     <artifactId>hutool-all</artifactId>
5     <version>5.7.0</version>
6 </dependency>
7 <dependency>
8     <groupId>org.bouncycastle</groupId>
9     <artifactId>bcprov-jdk15to18</artifactId>
10    <version>1.70</version>
11 </dependency>

```

```

1 import cn.hutool.core.util.StrUtil;
2 import cn.hutool.crypto.SecureUtil;

```

```
3 import cn.hutool.crypto.SmUtil;
4 import cn.hutool.crypto.symmetric.SymmetricCrypto;
5
6 @Slf4j
7 public class Sm4Util {
8
9     /**
10      * 加密
11      *
12      * @param privateKey 管理员提供
13      * @param str
14      * @return
15      */
16     public static String encrypt(final String privateKey, final String str) {
17         if (StrUtil.isBlank((CharSequence) str)) {
18             return "";
19         }
20
21         try {
22             SymmetricCrypto sm4 = new SymmetricCrypto("SM4/ECB/PKCS5Padding",
privateKey.getBytes());
23             return sm4.encryptHex(str, Charset.forName("UTF-8"));
24         } catch (Exception e) {
25             log.error("加密失败", e);
26             return null;
27         }
28     }
29
30     /**
31      * 解密
32      *
33      * @param privateKey 管理员提供
34      * @param str
35      * @return
36      */
37     public static String decrypt(final String privateKey, final String str) {
38         if (StrUtil.isBlank((CharSequence) str)) {
39             return null;
40         }
41         try {
42             SymmetricCrypto sm4 = new SymmetricCrypto("SM4/ECB/PKCS5Padding",
privateKey.getBytes());
43             return sm4.decryptStr(str, Charset.forName("UTF-8"));
44         } catch (Exception e) {
45             log.error("解密失败", e);
46             return null;
47         }
48     }
49 }
```

```
50     /**
51      * 验证签名
52      *
53      * @param appKey
54      * @param appSecret
55      * @param sign
56      * @param timestamp
57      * @return
58      */
59     public static Boolean checkSign(String appKey, String appSecret, String sign, String
timestamp) {
60         String dbSign = SecureUtil.md5(new StringBuilder().append(appKey)
61             .append(appSecret)
62             .append(timestamp).toString()).toLowerCase();
63         if (!dbSign.equals(sign)) {
64             return Boolean.FALSE;
65         }
66         return Boolean.TRUE;
67     }
68
69     /**
70      * 获取签名
71      *
72      * @param appKey
73      * @param appSecret
74      * @param timestamp
75      * @return
76      */
77     public static String getSign(String appKey, String appSecret, String timestamp) {
78         String sign = SecureUtil.md5(new StringBuilder().append(appKey)
79             .append(appSecret)
80             .append(timestamp).toString()).toLowerCase();
81         return sign;
82     }
83
84 }
85
```