流动人口自主申报接口

接口地址:`https://tx.hz-hanghui.com:8088/hanghui-server-platform/api/v1/open/recurrent/population/push`

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:

```
//国密m4加密后密文 也就是三方接口接收到的数据
{
  "appId":"",
  "bizContent":
"NI4ry+D01kDXrNf5OcmAzMGaB0td9kXfa321xmIS3qw5homFdZT4xaJu8Vqid37NDivdqj1ADzd0+v+QDKNovFSBkm
OyOQt2ObJ4HW43i0dKKHAKlI3FtZA0F9URhlqXbckbpGMD1hev3XS/b9uxRw+QLOnTzhuzgoxpiOXNA9rmQs3V7prpF
9Wpaetts2pv",
  "reqTimestamp": 1692693665800,
  "sign": "签名方式见文末加密算法"
}

//国密m4加密前 也就是三方服务解密后的数据
{
  "appId":"",
  "bizContent": {
      "xm": "姓名",
      "sfzh": "身份证号",
      "lxdh": "联系电话",
      "pcs": "派出所",
      "jwh": "居委会",
      "xzdxz": "现住地详址",
      "sblx": "申报类型"
  },
  "reqTimestamp": 1692693665800,
  "sign": "签名方式见文末加密算法"
}
```

请求参数:

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		
xm	姓名		true	string	
sfzh	身份证号		true	string	
lxdh	联系电话		true	string	
pcs	派出所		true	string	
jwh	居委会区域代码		true	string	举例330114001
xzdxz	现住地详址		true	string	
sblx	申报类型 1、登记 2、 注销		true	string	
hksx	户口省县		false	string	
hkxz	户口详址		false	string	
whcd	文化程度		false	string	
hyzk	婚姻状况		false	string	
xy	政治面貌		false	string	
xzdqh	现住地区划		false	string	
fh	房号		false	string	
fdxm	房东姓名		false	string	
fdgmsfhm	房东身份证号		false	string	
fdlxdh	房东联系电话		false	string	
gzdw	工作单位		false	string	
dwfzr	单位负责人		false	string	
dwlxdh	单位联系电话		false	string	
gzdz	工作地址		false	string	
szdzbm	实战地址编码		false	string	
stdzbm	省厅地址编码		false	string	
ryzp	照片base64		false	string	
xb	性别		false	string	
mz	民族		false	string	
csrq	出生日期		false	string	
cszy	从事职业		false	string	
czfwdah	出租房屋档案号		false	string	
bz	备注		false	string	
zzsy	暂(居)住事由		false	string	
zzcs	暂(居)住处所		false	string	
qzrq	启租日期		false	string	
tzrq	退租日期		false	string	
dwbm	单位编码		false	string	
appId	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	返回类«PdPersonBasicVO»
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			
msg	返回消息	string	

响应示例:

```
//国密m4加密后密文
{
    "code": 200,
    "data": null,
    "msg": ""
}

//国密m4加密前
{
    "code": 200,
    "data": null,
    "msg": ""
}
```

国密4加解密算法及签名认证

```
// pom.xml中引入
<dependency>
    <groupId>cn.hutool</groupId>
    <artifactId>hutool-all</artifactId>
    <version>5.7.0</version>
</dependency>
```

```
import cn.hutool.core.util.StrUtil;
import cn.hutool.crypto.SecureUtil;
import cn.hutool.crypto.SmUtil;
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
```