华泰互联网核心接口文档

简介: 华泰互联网核心接口文档

Version: 5.2

1.环境地址

测试环境地址: https://test.pc.ehuatai.com:9003/inco-gateway/commonOrder/接口地址

生产环境地址: 请联系华泰IT提供

调用方式:

使用协议：HTTPS请求响应模式 数据格式：JSON 传参方式：POST 响应模式：同步 POST参数：原始JSON数据 幂等性：由于存在网络重发等各种异常情况，需要被调用方做好幂等性控制，相同输入返回相同输出。

2.报文结构

请求报文: 数据文件采用标准的JSON格式，字符编码为UTF-8；下列业务接口中的出入参，均为request中的内容。

报文示例:

```
{
    <!– 待加密报文，可不传 –>
    "request"{
        "requestHeadVo":{
        },
        "XXXX":{}
    }
    <!– 密文 –>
    "signature": "",
    <!– 渠道代码 –>
    "chlCode": ""
}
```

回执报文格式: 数据文件采用标准的JSON格式，字符编码为UTF-8；

报文示例：

```
{
    <!– 待加密报文，可不传 –>
    "response"{
        "XXXX":{}
    }
    <!– 密文 –>
    "signature": "",
    <!– 渠道代码 –>
    "chlCode": ""
}
```

3.数据加密

加密说明: 采用RSA方式，对报文体request中的字符串(不包含request标签)进行Base64编码后生成的待加签内容。对待加签内容按照约定私钥和算法进行加签，签名信息放在signature节点中。（为防止空格、换行等字符影响，建议先生成JSONObject后，再转为String生成待价签内容）

测试环境私钥: 请联系华泰IT提供

生产环境私钥: 请联系华泰IT提供

jar包:

```
<dependency>
  <groupId>commons-codec</groupId>
  <artifactId>commons-codec</artifactId>
  <version>1.14</version>
</dependency>
```

加密示例:

```
package com.sinosoft.common;

import cn.hutool.core.util.CharsetUtil;
import cn.hutool.core.util.HexUtil;
import cn.hutool.core.util.StrUtil;
import cn.hutool.crypto.asymmetric.AsymmetricCrypto;
import cn.hutool.crypto.asymmetric.KeyType;
import cn.hutool.crypto.asymmetric.RSA;
import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONObject;
import com.google.gson.JsonObject;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.InputStream;

import java.math.BigInteger;
```

```
import java.security.GeneralSecurityException;
import java.security.Key;
import java.security.KeyFactory;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.SecureRandom;
import java.security.Signature;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;
import java.security.interfaces.RSAPrivateKey;
import java.security.interfaces.RSAPublicKey;
import java.security.spec.EncodedKeySpec;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import java.util.HashMap;
import java.util.Map;



import javax.crypto.Cipher;

import org.apache.commons.codec.binary.Base64;
import org.apache.commons.lang.StringUtils;
import sun.misc.BASE64Decoder;

/**
 *
 * @author simon.xxm
 * @version $Id: RSA.java, v 0.1 2016年1月25日 上午10:34:58 simon.xxm Exp $
 */
public class RsaUtil {

  /** 指定key的大小 */
  private static int          KEYSIZE        = 1024;
  private static final String encoding       = "UTF-8";
  private static final String RSA_ALGORITHM  = "RSA";
  /**
   * RSA最大加密明文大小
   */
   private static final int MAX_ENCRYPT_BLOCK = 117;
   /**
    * RSA最大解密密文大小
    */
   private static final int MAX_DECRYPT_BLOCK = 128;

  //私钥




  public static final String RSA_privateKey
```

```
="MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBAM2VGsDYC4h8ZZQMffETP4iDtPEgRduLOoyeD+YUY0PnYxD792Hx9T7gEm1AVg1x+QK5jQH4cXy/3XkEf7Nd5RWridju1y4mMDPGdFiSVWXtcucc9KsSPg0hbMq3dTY+KGRk1YfwaH5UUteh4kJaFaIqJM58qj7IvFP4O1p9rMcdAgMBAAECgYEAt4lPai03FrHgSe1hHrHNfcX/62mhlGBXdCTFEubOvFe+VPJuKA5IocqQCONwL+65ndoj7kdsoi/0vM7sZykDk9unHOwlRGhqVV3sGB9SkIfYuRU/6DDi7jq5fmNE2H6B7yHOX/Tp0fW70ZP//5R0eErl5NqEl6vP4HhOa9l8Na0CQQD0vXIZIco46oXgOcTTbaNWyFxY0f8XS62D7yvxHqmNK1o8q6Fpt+uFgsCJ7Qetfvh+7MKT9Pz/PgqflWGuZqlHAkEA1wp1tKSSwhIXB7vLpNnogz4g+lwY0JMtca08tca0gqI1QpJDcfSp9uNYT0TnES/5LUkV3HoTFjNJYEirmPP+ewJAeg7lka0terdUL2EATeX3OXfRvqZ0z3x5vDwTMTz2mKZPacS7SstkVgDA38jsNFYHvt17qWjcqLubdr18qwseTwJAQ/hnahjW1ob3RpeCb/H8v3ck31267jqHE7ZpSR+ssNnqsccfkGaATqxfnnat/s3GGh1Ozqi7XboKSGfP7YG5/wJAMPcD9PZf5o2T59gyBb2T0WZoaU7CNoZImfH8QkznB1a+FpKHzwOqmRGHzecbFYJguD3AYq19vNNHsNdrdDrYLQ==";
  //公钥
  public static final String RSA_publicKey
="MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDNlRrA2AuIfGWUDH3xEz+Ig7TxIEXbizqMng/mFGND52MQ+/dh8fU+4BJtQFYNcfkCuY0B+HF8v915BH+zXeUVq4nY7tcuJjAzxnRYklVl7XLnHPSrEj4NIWzKt3U2PihkZNWH8Gh+VFLXoeJCWhWiKiTOfKo+yLxT+DtafazHHQIDAQAB";



  public static String message = "{\"publicOrderVo\":{\"policyDynamicVos\":
[{\"dynamicValue\":\"开\",\"dynamicId\":\"1\",\"dynamicMean\":\"旅行1目的地
\",\"insuredNo\":\"1\",\"itemNo\":\"1\",\"dynamicKey\":\"travleDes\"}],\"policyProgrammeVos\":
[{\"policyRdrCategoryVos\":
[{\"itemNo\":\"1\",\"personDecimal\":\"1\",\"plan\":\"01MD0001\"}]}],\"policyMainVo\":
{\"amount\":\"145000\",\"effectiveTm\":\"2021-07-31
00:00:00\",\"premium\":\"45\",\"productCode\":\"01MD\",\"chlCode\":\"HT100042\",\"proTm\":\"2021
-07-30 15:18:28\",\"terminalTm\":\"2021-09-29
00:00:00\",\"paymentWayCode\":\"1\",\"copy\":\"1\",\"dataProducer\":\"PB\"},\"policyInsuredVos\"
:
[{\"insuredNum\":\"110101199901010023X\",\"insuredGender\":\"1\",\"insuredTelNum\":\"15001132199\
",\"islegal\":\"1\",\"isHolder\":\"0\",\"insuredIdType\":\"01\",\"insuredNo\":\"1\",\"insuredTyp
e\":\"1\",\"itemNo\":\"1\",\"insuredBirthday\":\"1999-01-01\",\"insuredName\":\"核酸
\",\"relationship\":\"03\"}],\"policyApplicantVo\":
{\"appEmail\":\"11111@qq.com\",\"appNum\":\"110101199901010096\",\"apptelNum\":\"15001132199\",\
"appGender\":\"1\",\"appName\":\"验证
\",\"appType\":\"1\",\"appidType\":\"01\",\"visaType\":\"2\",\"appBirthday\":\"1999-01-01
00:00:00\"}},\"requestHeadVo\":{\"requestType\":\"HTIC002\",\"channelCode\":\"HT100042\"}}";
  public static void main(String[] args) throws Exception {
    JSONObject inputObj = JSONObject.parseObject(message);
    message= inputObj.toJSONString();

    String s1 = rsaEncrypt(message, RSA_publicKey,"UTF-8");
    System.out.println(s1);

    String s = rsaDecrypt(s1, RSA_privateKey,"UTF-8");

    System.out.println(s);


  }



  public static PrivateKey getPrivateKey(String privateKey) throws Exception {
    KeyFactory keyFactory = KeyFactory.getInstance("RSA");
    byte[] decodedKey = Base64.decodeBase64(privateKey.getBytes());
    PKCS8EncodedKeySpec keySpec = new PKCS8EncodedKeySpec(decodedKey);
    return keyFactory.generatePrivate(keySpec);
  }
```

```
  public static PublicKey getPublicKey(String publicKey) throws Exception {
    KeyFactory keyFactory = KeyFactory.getInstance("RSA");
    byte[] decodedKey = Base64.decodeBase64(publicKey.getBytes());
    X509EncodedKeySpec keySpec = new X509EncodedKeySpec(decodedKey);
    return keyFactory.generatePublic(keySpec);
  }


  /**
   * 公钥加密<br>
   * 对于<b>开发者</b>，publicKey是指公钥，privateKey是指开发者自己的私钥<br>
   * 对于，publicKey是指开发者的公钥，privateKey是指私钥<br>
   *
   * @param content 待加密内容
   * @param publicKey 公钥
   * @param charset 字符集，如UTF-8
   * @return 密文内容
   * @throws Exception
   */
  public static String rsaEncrypt(String content, String publicKey, String charset) throws
Exception {
    PublicKey pubKey = getPublicKey(publicKey);
    Cipher cipher = Cipher.getInstance("RSA");
    cipher.init(Cipher.ENCRYPT_MODE, pubKey);
    byte[] data = StringUtils.isEmpty(charset) ? content.getBytes() : content.getBytes(charset);
    int inputLen = data.length;
    ByteArrayOutputStream out = new ByteArrayOutputStream();
    int offSet = 0;
    byte[] cache;
    int i = 0;
    // 对数据分段加密
    while (inputLen - offSet > 0) {
      if (inputLen - offSet > 117) {
        cache = cipher.doFinal(data, offSet, 117);
      } else {
        cache = cipher.doFinal(data, offSet, inputLen - offSet);
      }
      out.write(cache, 0, cache.length);
      i++;
      offSet = i * 117;
    }
    byte[] encryptedData = Base64.encodeBase64(out.toByteArray());
    out.close();

    return StringUtils.isEmpty(charset) ? new String(encryptedData) : new String(encryptedData,
charset);
  }

  /**
   * 私钥解密<br>
   *
   * @param content 待解密内容
   * @param privateKey 私钥
   * @param charset 字符集，如UTF-8
   * @return 明文内容
   * @throws Exception
```

```
   */
  public static String rsaDecrypt(String content, String privateKey, String charset) throws
Exception {
    try {
      PrivateKey priKey = getPrivateKey(privateKey);
      Cipher cipher = Cipher.getInstance("RSA");
      cipher.init(Cipher.DECRYPT_MODE, priKey);
      byte[] encryptedData =
          StringUtils.isEmpty(charset) ? Base64.decodeBase64(content.getBytes()) : Base64
              .decodeBase64(content.getBytes(charset));
      int inputLen = encryptedData.length;
      ByteArrayOutputStream out = new ByteArrayOutputStream();
      int offSet = 0;
      byte[] cache;
      int i = 0;
      // 对数据分段解密
      while (inputLen - offSet > 0) {
        if (inputLen - offSet > 128) {
          cache = cipher.doFinal(encryptedData, offSet, 128);
        } else {
          cache = cipher.doFinal(encryptedData, offSet, inputLen - offSet);
        }
        out.write(cache, 0, cache.length);
        i++;
        offSet = i * 128;
      }
      byte[] decryptedData = out.toByteArray();
      out.close();

      return StringUtils.isEmpty(charset) ? new String(decryptedData)
          : new String(decryptedData, charset);
    } catch (Exception e) {
      throw new Exception("EncodeContent = " + content + ",charset = " + charset, e);
    }
  }


}
```

- ***以下业务接口均采用通用的报文结构***

6.互联网核心中小微企业险数据回传接口(互联网核心调用渠道)

接口说明:用于使用华泰收银台支付，支付完成生成保单号后，由互联网核心调用渠道，通知报文为RSA加密密文;

接口地址:联系华泰IT提供渠道调用地址

请求方式:POST

请求数据类型:application/json

响应数据类型:application/json

接口类型:RequestHeadVo.requestType = "CHL_NOTICE_01"

明文请求示例:

```
{
    "policyMainVo": {
        "orderNo": "PA877610670029176832",
        "policyNo": "PB2516440121001009 80",
        "policyUrl": "https://test.pc.ehuatai.com:9003/inco-gateway/printPdf?
key=2EEBAA8C9CBAAC39FBC4AB14EB509912",
       "effectiveTm":"2022-12-01 00:00:00",
       "terminalTm":"2023-12-01 00:00:00",
       "insuranceCode":"3909",
       "productCode":"0K42",
        "productName":"餐饮业综合保险",
        "plan": "0K420001",
        "planName": "尊享版",
        "chlDptCode": "ZY",
        "chlUserCode": "0043002",
        "proTm":"2022-11-11 10:54:00",
        "premium: 2000.2,
        "amount: 200000,
        "status": "1"
    },
    "policyApplicantVo": {
        "appName": "测试机构名称",
        "appNum": "97XXXXXXXXXXXXXXX",
        "appidType": "97",
        "apptelNum": "15611679194",
        "appEmail": "yyl048@qq.coom"
    },
    "policyInsuredVos": [{
        "insuredIdType": "97",
        "insuredName": "测试机构名称",
        "insuredNum": "97XXXXXXXXXXXXXXX",
        "relationship": "03"
    }],
    "rdrGroupVos": [{
        "groupNo": 1
        "groupName": "财产一切险",
        "premium": 100.00,
        "amount": 100.00
    }],
   "policyItemVo": [{
        "industryClass": "387207_2",
        "industrySubclass": "01",
        "insuranceCoverage": "110000",
```

```
            "InsuranceMarket": "110100",
            "InsuranceArea": "110101",
            "InsuranceAddress": "门牌号XXX"
        }],
        "requestHeadVo": {
            "channelCode": "HT100220",
            "thirdTransId": "POR00000000017294"
        }
}
```

密文请求示例:

```
{
    <!– 密文 –>
    "signature": ""
}
```

请求参数:

参数名称	参数说明	是否必须	数据类型
paySuccessNoticeVo	paySuccessNoticeVo	true	VO
policyMainVo			VO
orderNo	订单号	true	string
policyNo	保单号	true	string
policyUrl	保单下载地址	true	string
insuranceCode	险种编码	true	string
productCode	产品代码	true	string
productName	产品名称	true	string
plan	计划代码	true	string
planName	计划名称	true	string
chlDptCode	渠道出单机构	false	string
chlUserCode	渠道出单员	false	string
effectiveTm	保险起期	true	date
terminalTm	保险止期	true	date
proTm	投保日期	true	date
premium	总保额(单位元)	true	long

参数名称	参数说明	是否必须	数据类型
amount	总保费(单位元)	true	long
status	保单状态1投保 2退保	true	long
requestHeadVo		true	VO
channelCode	渠道编码	false	string
thirdTransId	请求id	false	string
policyInsuredVos			List
insuredName	被保人姓名	true	string
insuredIdType	被保人证件类型	true	string
insuredNum	被保人证件号	true	string
relationship	被保人与投保人关系	true	string
policyApplicantVo		true	VO
appName	投保人姓名	true	string
appidType	投保人证件类型	true	string
appNum	投保人证件号	true	string
apptelNum	投保人电话号	false	string
appEmail	投保人邮箱	false	string
policyItemVo			List
industryClass	行业大类(参见码表 行业类别)	true	string
industrySubclass	行业小类(参见码表 行业类别)	true	string
insuranceCoverage	省（参照地区国标码）	true	string
insuranceMarket	市（参照地区国标码）	true	string
insuranceArea	区（参照地区国标码）	true	string
insuranceAddress	详细地址	true	string
rdrGroupVos			List
groupName	险类名称	true	string
groupNo	序号	true	int
amount	险类保额	true	long

参数名称	参数说明	是否必须	数据类型
premium	险类保费	true	long

响应参数:

参数名称	参数说明	类型
responseDate		返回时间
status		integer(int64) 0成功
statusText		请求结果

响应为明文示例:

```
{
    "responseDate": "",
    "status": 0,
    "statusText": ""
}
```

码表

投保人与被保人关系

00：本人 01：配偶 02：父母 03：子女 05：兄弟姐妹 06：雇主 07：雇员 08：祖父母、外祖父母 09：祖孙、外祖孙 10：监护人 11：被监护人 12：朋友 17：雇佣 99：其他

行业类别

387207_1：办公室 01：咨询服务公司 02：律师事务所 03：金融机构 04：IT信息技术 05：贸易公司

387207_2：餐饮业 01：中餐 02：西餐 03：粥/面点 04：咖啡/茶 05：快餐 06：自助餐 07：甜品 08：其他餐饮店（不包含火锅店）

387207_3：医疗齿科诊所 01：齿科诊所 02：专科诊所 03：中医诊所 04：一般诊所

387207_4：美容美发业 01：美容 02：美发 03：纤体（瘦身） 04：SPA 05：美甲 06：推拿按摩

387207_5：零售业 01：纺织品、时装服装及其饰品 02：人造珠宝装饰品 03：箱包、皮革制品 04：鞋店 05：日化用品 06：婴儿用品/成人用品 07：眼镜及其他光学器材 08：家具、家用电器 09：窗帘、地毯、床上用品 10：照明用具及配件 11：陶器、餐具及其厨房用具 12：运动休闲用品、器材及配件便利店 13：乐器店、音像制品 14：书店、文具店 15：玩具及游戏用品 16：礼品、纪念品 17：办公设备用品（不包括电脑或移动电子设备）

18：便利店（小于300平米，不含加油站内零售店） 19：水果、蔬菜 20：糖果、饼干 21：特产及日用杂货 22：花店、人造花饰以及人造盆景（不包括烟草、酒类、茶叶专卖店）

387207_6：教育培训机构 06：学前教育