目　录

新颜科技产品服务接入指引

1.引言

1. 文档概述

新颜科技团队提供对外接口，供第三方机构系统与新颜科技系统对接。本文档拟对交互流程和接口规范进行描述，以指导机构系统开发人员顺利对接。

2. 联系我们

电话：18321124941
邮箱：support@xinyan.com
QQ： 3312259832

3. 阅读对象

1.接入新颜科技服务机构的系统开发人员。
2.新颜科技对外接口相关技术人员，包括开发和技术支持。

2.对接流程

1. 商务经理建立对接沟通群

商务建立对接群后双方商务会拉技术人员进群确定需要对接的产品，双方对接人员进行初步沟通。

2. 测试环境联调

确定需要对接的产品后，新颜会提供当前最新的接口文档和Demo供参考，开发人员可以参考Demo和接口文档进行对接，走通测试环境流程，尽量实现前期的基本逻辑，做好上线准备。

3. 上线准备

在协议流程完成后，新颜会根据协议给商户开通对应正式环境信息，客户获取信息后可以配置生产环境，将之前的测试环境 配置替换为生产信息，配置说明如下：
1.商户和新颜签约开通后会将相关信息发送到业务联系人邮箱。
2.终端号获取
3.替换信息：商户号、终端号、私钥文件、私钥密码、公钥文件和生产地址URL。
4.生产和测试环境地址可在接口文档获取，需要变更域名。
5.生产证书使用工具生成(支持windows和linux)，运行脚本自定义密钥名称和密码，生成后将cer公钥上传到开放平台（https://open.xinyan.com), 私钥pfx和密码代码中使用。

4. 切量后上线观察

正式上线后，新颜会协助一起观察交易稳定情况，核对双方请求数据是否同步，核对无误后上线完成。

3.Demo参考

3.1 说明

新颜一般会给出三种常见语言的Demo供参考，例如java、php、c#。

4.关于加密-RSA数字证书

4.1.说明

新颜api接口一般对业务参数进行加密传输，一般先把业务参数组装为json,再进行base64,最后用rsa加密是算法进行加密得到加密串。特别说明新颜提供的工具一般生成私钥文件和公钥文件为pfx和cer,适用于java、php、c#等语言，其他语言采用pem,详细请见pfx转pem.

4.2 java加密

Main.java

```java
public class Main {

    public static void main(String[] args) {

        String pfxpwd = "217526";// 私钥密码
        String pfxpath = "D:\\CER\\8000013189_pri.pfx";//商户私钥路径
        String base64str = "123456";// 待加密的字符串
        base64str.replaceAll("\r", "").replaceAll("\n", "");//重要 避免出现换行空格符
        String data_content = RsaCodingUtil.encryptByPriPfxFile(base64str, pfxpath, pfxpwd);//加密数据
        System.out.println(data_content);

    }

}
```

RsaCodingUtil.java

```java
import java.io.UnsupportedEncodingException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;

/**
 * <b>Rsa加解密工具</b><br>
 * <br>
 * 公钥采用X509,Cer格式的<br>
 * 私钥采用PKCS12加密方式的PFX私钥文件<br>
 * 加密算法为1024位的RSA，填充算法为PKCS1Padding<br>
 *
 * @author 行者
 * @version 4.1.0
 */
public final class RsaCodingUtil {

    // =======================================================================================
    // 公钥加密私钥解密段
    // =======================================================================================

    /**
     * 指定Cer公钥路径加密
     *
     * @param src
     * @param pubCerPath
     * @return hex串
     */
    public static String encryptByPubCerFile(String src, String pubCerPath) {

        PublicKey publicKey = RsaReadUtil.getPublicKeyFromFile(pubCerPath);
        if (publicKey == null) {
            return null;
        }
        return encryptByPublicKey(src, publicKey);
    }

    /**
     * 用公钥内容加密
     *
     * @param src
     * @param pubKeyText
     * @return hex串
     */
    public static String encryptByPubCerText(String src, String pubKeyText) {
        PublicKey publicKey = RsaReadUtil.getPublicKeyByText(pubKeyText);
        if (publicKey == null) {
            return null;
        }
        return encryptByPublicKey(src, publicKey);
    }

    /**
     * 公钥加密返回
     *
     * @param src
     * @param publicKey
     * @param encryptMode
     * @return hex串
     */
    public static String encryptByPublicKey(String src, PublicKey publicKey) {
        byte[] destBytes = rsaByPublicKey(src.getBytes(), publicKey, Cipher.ENCRYPT_MODE);

        if (destBytes == null) {
            return null;
        }

        return FormatUtil.byte2Hex(destBytes);

    }

    /**
     * 根据私钥文件解密
     *
     * @param src
     * @param pfxPath
     * @param priKeyPass
     * @return
     */
    public static String decryptByPriPfxFile(String src, String pfxPath, String priKeyPass) {
        if (FormatUtil.isEmpty(src) || FormatUtil.isEmpty(pfxPath)) {
            return null;
        }
        PrivateKey privateKey = RsaReadUtil.getPrivateKeyFromFile(pfxPath, priKeyPass);
        if (privateKey == null) {
            return null;
        }
        return decryptByPrivateKey(src, privateKey);
    }

    /**
     * 根据私钥文件流解密
     *
     * @param src
     * @param pfxPath
     * @param priKeyPass
     * @return
     */
    public static String decryptByPriPfxStream(String src, byte[] pfxBytes, String priKeyPass) {
        if (FormatUtil.isEmpty(src)) {
            return null;
        }
        PrivateKey privateKey = RsaReadUtil.getPrivateKeyByStream(pfxBytes, priKeyPass);
        if (privateKey == null) {
            return null;
        }
        return decryptByPrivateKey(src, privateKey);
    }

    /**
     * 私钥解密
     *
     * @param src
     * @param privateKey
     * @return
     */
    public static String decryptByPrivateKey(String src, PrivateKey privateKey) {
        if (FormatUtil.isEmpty(src)) {
            return null;
        }
        try {
            byte[] destBytes = rsaByPrivateKey(FormatUtil.hex2Bytes(src), privateKey, Cipher.DECRYPT_MODE);
            if (destBytes == null) {
                return null;
            }
            return new String(destBytes, RsaConst.ENCODE);
        } catch (UnsupportedEncodingException e) {
            // //log.error("解密内容不是正确的UTF8格式:", e);
        } catch (Exception e) {
            // //log.error("解密内容异常", e);
        }

```

```
            return null;
        }

        // ================================================================
=======================
        // 私钥加密公钥解密
        // ================================================================
=======================

        /**
         * 根据私钥文件加密
         *
         * @param src
         * @param pfxPath
         * @param priKeyPass
         * @return
         */
        public static String encryptByPriPfxFile(String src, String pfxPat
h, String priKeyPass) {

            PrivateKey privateKey = RsaReadUtil.getPrivateKeyFromFile(pfxP
ath, priKeyPass);
            if (privateKey == null) {
                return null;
            }
            return encryptByPrivateKey(src, privateKey);
        }

        /**
         * 根据私钥文件流加密
         *
         * @param src
         * @param pfxPath
         * @param priKeyPass
         * @return
         */
        public static String encryptByPriPfxStream(String src, byte[] pfxB
ytes, String priKeyPass) {
            PrivateKey privateKey = RsaReadUtil.getPrivateKeyByStream(pfxB
ytes, priKeyPass);
            if (privateKey == null) {
                return null;
            }
            return encryptByPrivateKey(src, privateKey);
        }

        /**
         * 根据私钥加密
         *
         * @param src
         * @param privateKey
         */
        public static String encryptByPrivateKey(String src, PrivateKey pr
ivateKey) {

            byte[] destBytes = rsaByPrivateKey(src.getBytes(), privateKey,
 Cipher.ENCRYPT_MODE);

            if (destBytes == null) {
                return null;
            }
            return FormatUtil.byte2Hex(destBytes);

        }

        /**
         * 指定Cer公钥路径解密
         *
         * @param src
         * @param pubCerPath
         * @return
         */
        public static String decryptByPubCerFile(String src, String pubCerPath) {
            PublicKey publicKey = RsaReadUtil.getPublicKeyFromFile(pubCerPath);
            if (publicKey == null) {
                return null;
            }
            return decryptByPublicKey(src, publicKey);
        }

        /**
         * 根据公钥文本解密
         *
         * @param src
         * @param pubKeyText
         * @return
         */
        public static String decryptByPubCerText(String src, String pubKeyText) {
            PublicKey publicKey = RsaReadUtil.getPublicKeyByText(pubKeyText);
            if (publicKey == null) {
                return null;
            }
            return decryptByPublicKey(src, publicKey);
        }

        /**
         * 根据公钥解密
         *
         * @param src
         * @param publicKey
         * @return
         */
        public static String decryptByPublicKey(String src, PublicKey publicKey) {

            try {
                byte[] destBytes = rsaByPublicKey(FormatUtil.hex2Bytes(src), publicKey, Cipher.DECRYPT_MODE);
                if (destBytes == null) {
                    return null;
                }
                return new String(destBytes, RsaConst.ENCODE);
            } catch (UnsupportedEncodingException e) {
                log("解密出错" + e);

            }
```

```
            return null;
        }

        // ==============================================================
=======================
        // 公私钥算法
        // ==============================================================
=======================
        /**
         * 公钥算法
         *
         * @param srcData    源字节
         * @param publicKey 公钥
         * @param mode       加密 OR 解密
         * @return
         */
        public static byte[] rsaByPublicKey(byte[] srcData, PublicKey publ
icKey, int mode) {
            try {
                Cipher cipher = Cipher.getInstance(RsaConst.RSA_CHIPER);
                cipher.init(mode, publicKey);
                // 分段加密
                int blockSize = (mode == Cipher.ENCRYPT_MODE) ? RsaConst.E
NCRYPT_KEYSIZE : RsaConst.DECRYPT_KEYSIZE;
                byte[] encryptedData = null;
                for (int i = 0; i < srcData.length; i += blockSize) {
                    // 注意要使用2的倍数，否则会出现加密后的内容再解密时为乱码
                    byte[] doFinal = cipher.doFinal(subarray(srcData, i, i
 + blockSize));
                    encryptedData = addAll(encryptedData, doFinal);
                }
                return encryptedData;

            } catch (NoSuchAlgorithmException e) {
                // //log.error("公钥算法-不存在的解密算法:", e);
            } catch (NoSuchPaddingException e) {
                // //log.error("公钥算法-无效的补位算法:", e);
            } catch (IllegalBlockSizeException e) {
                // //log.error("公钥算法-无效的块大小:", e);
            } catch (BadPaddingException e) {
                // //log.error("公钥算法-补位算法异常:", e);
            } catch (InvalidKeyException e) {
                // //log.error("公钥算法-无效的私钥:", e);
            }
            return null;
        }

        /**
         * 私钥算法
         *
         * @param srcData     源字节
         * @param privateKey 私钥
         * @param mode        加密 OR 解密
         * @return
         */
        public static byte[] rsaByPrivateKey(byte[] srcData, PrivateKey pr
ivateKey, int mode) {
            try {
                Cipher cipher = Cipher.getInstance(RsaConst.RSA_CHIPER);
                cipher.init(mode, privateKey);
                // 分段加密
                int blockSize = (mode == Cipher.ENCRYPT_MODE) ? RsaConst.ENCRYPT_KEYSIZE : RsaConst.DECRYPT_KEYSIZE;
                byte[] decryptData = null;

                for (int i = 0; i < srcData.length; i += blockSize) {
                    byte[] doFinal = cipher.doFinal(subarray(srcData, i, i + blockSize));

                    decryptData = addAll(decryptData, doFinal);
                }
                return decryptData;
            } catch (NoSuchAlgorithmException e) {
                // //log.error("私钥算法-不存在的解密算法:", e);
            } catch (NoSuchPaddingException e) {
                // log.error("私钥算法-无效的补位算法:", e);
            } catch (IllegalBlockSizeException e) {
                // log.error("私钥算法-无效的块大小:", e);
            } catch (BadPaddingException e) {
                // log.error("私钥算法-补位算法异常:", e);
            } catch (InvalidKeyException e) {
                // log.error("私钥算法-无效的私钥:", e);
            }
            return null;
        }

        // ///////////////==========================
        public static byte[] subarray(byte[] array, int startIndexInclusive, int endIndexExclusive) {
            if (array == null) {
                return null;
            }
            if (startIndexInclusive < 0) {
                startIndexInclusive = 0;
            }
            if (endIndexExclusive > array.length) {
                endIndexExclusive = array.length;
            }
            int newSize = endIndexExclusive - startIndexInclusive;

            if (newSize <= 0) {
                return new byte[0];
            }

            byte[] subarray = new byte[newSize];

            System.arraycopy(array, startIndexInclusive, subarray, 0, newSize);

            return subarray;
        }

        public static byte[] addAll(byte[] array1, byte[] array2) {
            if (array1 == null) {
                return clone(array2);
            } else if (array2 == null) {
                return clone(array1);
            }
            byte[] joinedArray = new byte[array1.length + array2.length];
            System.arraycopy(array1, 0, joinedArray, 0, array1.length);
            System.arraycopy(array2, 0, joinedArray, array1.length, array2
    .length);
            return joinedArray;
        }

        public static byte[] clone(byte[] array) {
            if (array == null) {
                return null;
            }
            return (byte[]) array.clone();
        }

        public static void log(String msg) {
            System.out.println(new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")
    .format(new Date()) + "\t: " + msg);
        }
    }
```

RsaReadUtil.java

```
import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.io.StringReader;
import java.security.KeyStore;
import java.security.KeyStoreException;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.UnrecoverableKeyException;
import java.security.cert.Certificate;
import java.security.cert.CertificateException;
import java.security.cert.CertificateFactory;
import java.util.Enumeration;

import sun.misc.BASE64Decoder;

//import com.jweb.//log.Logger;

/**
 * <b>公私钥读取工具</b><br>
 * <br>
 *
 * @author 行者
 * @version 4.1.0
 */
public final class RsaReadUtil {

    /**
     * 根据Cer文件读取公钥
     *
     * @param pubCerPath
     * @return
     */
    public static PublicKey getPublicKeyFromFile(String pubCerPath) {
        FileInputStream pubKeyStream = null;
        try {
            pubKeyStream = new FileInputStream(pubCerPath);
            byte[] reads = new byte[pubKeyStream.available()];
            pubKeyStream.read(reads);
            return getPublicKeyByText(new String(reads));
        } catch (FileNotFoundException e) {
            // //log.error("公钥文件不存在:", e);
        } catch (IOException e) {
            // log.error("公钥文件读取失败:", e);
        } finally {
            if (pubKeyStream != null) {
                try {
                    pubKeyStream.close();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
        return null;
    }

    /**
     * 根据公钥Cer文本串读取公钥
     *
     * @param pubKeyText
     * @return
     */
    public static PublicKey getPublicKeyByText(String pubKeyText) {
        try {
            CertificateFactory certificateFactory = CertificateFactory
.getInstance(RsaConst.KEY_X509);
            BufferedReader br = new BufferedReader(new StringReader(pu
bKeyText));
            String line = null;
            StringBuilder keyBuffer = new StringBuilder();
            while ((line = br.readLine()) != null) {
                if (!line.startsWith("-")) {
                    keyBuffer.append(line);
                }
            }
            Certificate certificate = certificateFactory.generateCerti
ficate(
                    new ByteArrayInputStream(new BASE64Decoder().decod
eBuffer(keyBuffer.toString())));
            return certificate.getPublicKey();
        } catch (Exception e) {
            // log.error("解析公钥内容失败:", e);
        }
        return null;
    }

    /**
     * 根据私钥路径读取私钥
     *
     * @param pfxPath
     * @param priKeyPass
     * @return
     */
    public static PrivateKey getPrivateKeyFromFile(String pfxPath, Str
ing priKeyPass) {
        InputStream priKeyStream = null;
        try {
            priKeyStream = new FileInputStream(pfxPath);
            byte[] reads = new byte[priKeyStream.available()];
            priKeyStream.read(reads);
            return getPrivateKeyByStream(reads, priKeyPass);
        } catch (Exception e) {
            // log.error("解析文件，读取私钥失败:", e);
        } finally {
            if (priKeyStream != null) {
                try {
                    priKeyStream.close();
                } catch (Exception e) {
                    //
                }
            }
        }
        return null;
    }

    /**
     * 根据PFX私钥字节流读取私钥
     *
     * @param pfxBytes
     * @param priKeyPass
     * @return
     */
    public static PrivateKey getPrivateKeyByStream(byte[] pfxBytes, String priKeyPass) {
        try {
            KeyStore ks = KeyStore.getInstance(RsaConst.KEY_PKCS12);
            char[] charPriKeyPass = priKeyPass.toCharArray();
            ks.load(new ByteArrayInputStream(pfxBytes), charPriKeyPass);
            Enumeration<String> aliasEnum = ks.aliases();
            String keyAlias = null;
            if (aliasEnum.hasMoreElements()) {
                keyAlias = (String) aliasEnum.nextElement();
            }
            return (PrivateKey) ks.getKey(keyAlias, charPriKeyPass);
        } catch (IOException e) {
            // 加密失败
            // log.error("解析文件，读取私钥失败:", e);
        } catch (KeyStoreException e) {
            // log.error("私钥存储异常:", e);
        } catch (NoSuchAlgorithmException e) {
            // log.error("不存在的解密算法:", e);
        } catch (CertificateException e) {
            // log.error("证书异常:", e);
        } catch (UnrecoverableKeyException e) {
            // log.error("不可恢复的秘钥异常", e);
        }
        return null;
    }
}
```

RsaConst.java

```
public final class RsaConst {

    /** 编码 */
    public final static String ENCODE = "UTF-8";

    public final static String KEY_X509 = "X509";
    public final static String KEY_PKCS12 = "PKCS12";
    public final static String KEY_ALGORITHM = "RSA";
    public final static String CER_ALGORITHM = "MD5WithRSA";

    public final static String RSA_CHIPER = "RSA/ECB/PKCS1Padding";

    public final static int KEY_SIZE = 1024;
    /** 1024bit 加密块 大小 */
    public final static int ENCRYPT_KEYSIZE = 117;
    /** 1024bit 解密块 大小 */
    public final static int DECRYPT_KEYSIZE = 128;
}
```

FormatUtil.java

```

import java.math.BigDecimal;
import java.text.DecimalFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.Set;

public final class FormatUtil {
    /** ==============IS Base=================== */
    /** 判断是否为整数(包括负数) */
    public static boolean isNumber(Object arg) {
        return NumberBo(0, toString(arg));
    }

    /** 判断是否为小数(包括整数,包括负数) */
    public static boolean isDecimal(Object arg) {
        return NumberBo(1, toString(arg));
    }

    /** 判断是否为空 ,为空返回true */
    public static boolean isEmpty(Object arg) {
        return toStringTrim(arg).length() == 0 ? true : false;
    }

    /** ==============TO Base=================== */
    /**
     * Object 转换成 Int 转换失败 返回默认值 0 <br>
     * 使用:toInt(值,默认值[选填])
     */
    public static int toInt(Object... args) {
        int def = 0;
        if (args != null) {
            String str = toStringTrim(args[0]);
            // 判断小数情况。舍弃小数位
            int stri = str.indexOf('.');
            str = stri > 0 ? str.substring(0, stri) : str;
            if (args.length > 1)
                def = Integer.parseInt(args[args.length - 1].toString());
            if (isNumber(str))
                return Integer.parseInt(str);
        }
        return def;
    }

    /**
     * Object 转换成 Long 转换失败 返回默认值 0 <br>
     * 使用:toLong(值,默认值[选填])
     */
    public static long toLong(Object... args) {
        Long def = 0L;
        if (args != null) {
            String str = toStringTrim(args[0]);
            if (args.length > 1)
                def = Long.parseLong(args[args.length - 1].toString());
            if (isNumber(str))
                return Long.parseLong(str);
        }
        return def;
    }

    /**
     * Object 转换成 Double 转换失败 返回默认值 0 <br>
     * 使用:toDouble(值,默认值[选填])
     */
    public static double toDouble(Object... args) {
        double def = 0;
        if (args != null) {
            String str = toStringTrim(args[0]);
            if (args.length > 1)
                def = Double.parseDouble(args[args.length - 1].toString());
            if (isDecimal(str))
                return Double.parseDouble(str);
        }
        return def;
    }

    /**
     * Object 转换成 BigDecimal 转换失败 返回默认值 0 <br>
     * 使用:toDecimal(值,默认值[选填]) 特别注意: new BigDecimal(Double) 会有误差，得先转String
     */
    public static BigDecimal toDecimal(Object... args) {
        return new BigDecimal(Double.toString(toDouble(args)));
    }

    /**
     * Object 转换成 Boolean 转换失败 返回默认值 false <br>
     * 使用:toBoolean(值,默认值[选填])
     */
    public static boolean toBoolean(String bool) {
        if (isEmpty(bool) || (!bool.equals("1") && !bool.equalsIgnoreCase("true") && !bool.equalsIgnoreCase("ok")))
```

```java
                return false;
            else
                return true;
        }

        /**
         * Object 转换成 String 为null 返回空字符 <br>
         * 使用:toString(值,默认值[选填])
         */
        public static String toString(Object... args) {
            String def = "";
            if (args != null) {
                if (args.length > 1)
                    def = toString(args[args.length - 1]);
                Object obj = args[0];
                if (obj == null)
                    return def;
                return obj.toString();
            } else {
                return def;
            }
        }

        /**
         * Object 转换成 String[去除所以空格]; 为null 返回空字符 <br>
         * 使用:toStringTrim(值,默认值[选填])
         */
        public static String toStringTrim(Object... args) {
            String str = toString(args);
            return str.replaceAll("\\s*", "");
        }

        /** ==============Other Base=================== */
        public static String getNowTime() {
            return new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(new
Date());
        }

        /** 数字左边补0 ,obj:要补0的数， length:补0后的长度 */
        public static String leftPad(Object obj, int length) {
            return String.format("%0" + length + "d", toInt(obj));
        }

        /** 小数 转 百分数 */
        public static String toPercent(Double str) {
            StringBuffer sb = new StringBuffer(Double.toString(str * 100.0
000d));
            return sb.append("%").toString();
        }

        /** 百分数 转 小数 */
        public static Double toPercent2(String str) {
            if (str.charAt(str.length() - 1) == '%')
                return Double.parseDouble(str.substring(0, str.length() -
1)) / 100.0000d;
            return 0d;
        }

        /**
         * 将byte[] 转换成字符串
```

```
     */
    public static String byte2Hex(byte[] srcBytes) {
        StringBuilder hexRetSB = new StringBuilder();
        for (byte b : srcBytes) {
            String hexString = Integer.toHexString(0x00ff & b);
            hexRetSB.append(hexString.length() == 1 ? 0 : "").append(h
exString);
        }
        return hexRetSB.toString();
    }

    /**
     * 将16进制字符串转为转换成字符串
     */
    public static byte[] hex2Bytes(String source) {
        try {
            byte[] sourceBytes = new byte[source.length() / 2];
            for (int i = 0; i < sourceBytes.length; i++) {
                sourceBytes[i] = (byte) Integer.parseInt(source.substr
ing(i * 2, i * 2 + 2), 16);
            }
            return sourceBytes;

        } catch (NumberFormatException e) {
            log("转换出错" + e);
            throw new NumberFormatException();

        }
    }

    /** String 转 Money */
    public static String toMoney(Object str, String MoneyType) {
        DecimalFormat df = new DecimalFormat(MoneyType);
        if (isDecimal(str))
            return df.format(toDecimal(str)).toString();
        return df.format(toDecimal("0.00")).toString();
    }

    /** 获取字符串str 左边len位数 */
    public static String getLeft(Object obj, int len) {
        String str = toString(obj);
        if (len <= 0)
            return "";
        if (str.length() <= len)
            return str;
        else
            return str.substring(0, len);
    }

    /** 获取字符串str 右边len位数 */
    public static String getRight(Object obj, int len) {
        String str = toString(obj);
        if (len <= 0)
            return "";
        if (str.length() <= len)
            return str;
        else
            return str.substring(str.length() - len, str.length());
    }

```

```java
    /**
     * 首字母变小写
     */
    public static String firstCharToLowerCase(String str) {
        Character firstChar = str.charAt(0);
        String tail = str.substring(1);
        str = Character.toLowerCase(firstChar) + tail;
        return str;
    }

    /**
     * 首字母变大写
     */
    public static String firstCharToUpperCase(String str) {
        Character firstChar = str.charAt(0);
        String tail = str.substring(1);
        str = Character.toUpperCase(firstChar) + tail;
        return str;
    }

    /**
     * List集合去除重复值 只能用于基本数据类型，。 对象类集合，自己写
     */
    @SuppressWarnings({ "rawtypes", "unchecked" })
    public static List delMoreList(List list) {
        Set set = new HashSet();
        List newList = new ArrayList();
        for (Iterator iter = list.iterator(); iter.hasNext();) {
            Object element = iter.next();
            if (set.add(element))
                newList.add(element);
        }
        return newList;
    }

    public static String formatParams(String message, Object[] params) {
        StringBuffer msg = new StringBuffer();
        String temp = "";
        for (int i = 0; i < params.length + 1; i++) {
            int j = message.indexOf("{}") + 2;
            if (j > 1) {
                temp = message.substring(0, j);
                temp = temp.replaceAll("\\{\\}", FormatUtil.toString(params[i]));
                msg.append(temp);
                message = message.substring(j);
            } else {
                msg.append(message);
                message = "";
            }
        }
        return msg.toString();
    }

    /** ============== END =================== */
    public final static class MoneyType {
        /** * 保留2位有效数字，整数位每3位逗号隔开 （默认） */
        public static final String DECIMAL = "#,##0.00";
        /** * 保留2位有效数字 */
        public static final String DECIMAL_2 = "0.00";
        /** * 保留4位有效数字 */
        public static final String DECIMAL_4 = "0.0000";
    }

    private static boolean NumberBo(int type, Object obj) {
        if (isEmpty(obj))
            return false;
        int points = 0;
        int chr = 0;
        String str = toString(obj);
        for (int i = str.length(); --i >= 0;) {
            chr = str.charAt(i);
            if (chr < 48 || chr > 57) { // 判断数字
                if (i == 0 && chr == 45) // 判断 - 号
                    return true;
                if (i >= 0 && chr == 46 && type == 1) { // 判断 . 号
                    ++points;
                    if (points <= 1)
                        continue;
                }
                return false;
            }
        }
        return true;
    }

    public static void log(String msg) {
        System.out.println(new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")
.format(new Date()) + "\t: " + msg);
    }
}
```

4.3 php加密

EncryptUtil.php

```
<?php

/**
 * Created by PhpStorm.
 * User: BF100311
 * Date: 2018/7/29
 * Time: 17:59
 */
class EncryptUtil
{

    private $private_key;
    private $public_key;

    function __construct($private_key_path, $public_key_path, $private_key_password, $Debug = FALSE)
    {
        if (!$Debug) {
            ob_start();
        }
        // 初始化商户私钥
        $pkcs12 = file_get_contents($private_key_path);
        $private_key = array();
        openssl_pkcs12_read($pkcs12, $private_key, $private_key_password);
        //echo "私钥是否可用:", empty($private_key) == true ? '不可用':'可用', "\n";
        $this->private_key = $private_key["pkey"];

        /**
         * $keyFile = file_get_contents($public_key_path);
         * $this->public_key = openssl_get_publickey($keyFile);
         * LOG::LogWirte(empty($this->public_key) == true ? "公钥是否可用:不可用" : "公钥是否可用:可用");
         */

        if (!$Debug) {
            ob_end_clean();
        }
    }

    // 私钥加密
    function encryptedByPrivateKey($data_content)
    {
        $data_content = base64_encode($data_content);
        $encrypted = "";
        $totalLen = strlen($data_content);
        $encryptPos = 0;
        while ($encryptPos < $totalLen) {
            openssl_private_encrypt(substr($data_content, $encryptPos, 117), $encryptData, $this->private_key);
            $encrypted .= bin2hex($encryptData);
            $encryptPos += 117;
        }
        return $encrypted;
    }

    // 公钥解密
    function decryptByPublicKey($encrypted)
    {
        if (!function_exists('hex2bin')) {
            throw new Exception("请启用hex2bin方法");
        }

        $decrypt = "";
        $totalLen = strlen($encrypted);
        $decryptPos = 0;
        while ($decryptPos < $totalLen) {
            openssl_public_decrypt(hex2bin(substr($encrypted, $decryptPos, 256)), $decryptData, $this->public_key);
            $decrypt .= $decryptData;
            $decryptPos += 256;
        }
        $decrypt = base64_decode($decrypt);
        return $decrypt;
    }

    /**
     * function hex2bin( $str ) {
     * $sbin = "";
     * $len = strlen( $str );
     * for ( $i = 0; $i < $len; $i += 2 ) {
     * $sbin .= pack( "H*", substr( $str, $i, 2 ) );
     * }
     * return $sbin;
     * } */
}
```

开始调用

```
// **** 先BASE64进行编码再RSA加密 ***  $pfx_pwd私钥路径  $pfx_pwd私钥密码
$encryptUtil = new EncryptUtil($pfxpath, "", $pfx_pwd, TRUE); //实例化加密类。
$data_content = $encryptUtil->encryptedByPrivateKey($data_content);
echo $data_content;
```

4.4 c#代码片段

```csharp
string pfxPath = "D:\\CER\\8000013189_pri.pfx";//私钥路径
string pfxPwd = "217526";//私钥密码
string XmlOrJson = "123456";//待加密数据
string data_content = RSAUtil.EncryptRSAByPfx(XmlOrJson, HttpContext.Current.Server.MapPath(pfxPath), pfxPwd);
Log.LogWrite("====加密串:" + data_content);
```

具体请参考附件：

附件

- RSACode.rar

4.5 python加密

备注：如果使用python3.* 在Windows进行开发，不要用windows系统安装M2Crypto，基本不会安装成功。可以使用vmware workstation或者virtualbox安装liunx或者Mac OS系统进行开发。

Python 2.*版本

```python
import base64
import M2Crypto
class RsaUtil:

    @staticmethod
    def encrypt(digest, private_key):
        digest=base64.b64encode(digest)
        result = ""
        while (len(digest) > 117):
            some = digest[0:117]
            digest = digest[117:]
            result += private_key.private_encrypt(some, M2Crypto.RSA.pkcs1_padding).encode("hex")

        result += private_key.private_encrypt(digest, M2Crypto.RSA.pkcs1_padding).encode("hex")

        return result

if __name__ == "__main__":
    private_key = M2Crypto.RSA.load_key('8000013189_pri.pem')
    result= RsaUtil.encrypt("123456",private_key)
    print result
```

Python 3.*版本

```python
import base64
import M2Crypto
class RsaUtil:

    private_key = M2Crypto.RSA.load_key('8000013189_test.pem')

    @staticmethod
    def encrypt(digest, private_key):
        digest=base64.b64encode(digest.encode('utf-8'))
        result = ""
        while (len(digest) > 117):
            some = digest[0:117]
            digest = digest[117:]
            result += private_key.private_encrypt(some, M2Crypto.RSA.pkcs1_padding).hex()

        result += private_key.private_encrypt(digest, M2Crypto.RSA.pkcs1_padding).hex()

        return result
```

```

if __name__ == "__main__":
    rsaUtil=RsaUtil()
    result=rsaUtil.encrypt("123456",RsaUtil.private_key)
    print(result)
```

4.6 node js加密

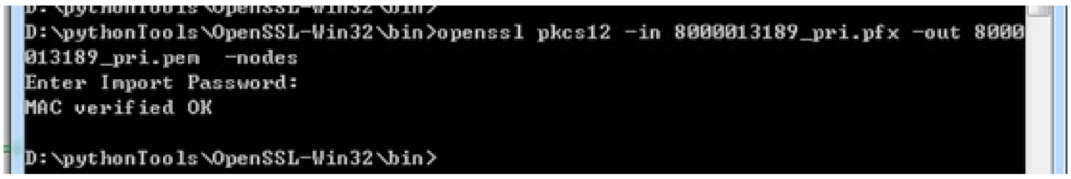
```
const fs=require('fs');
const crypto=require('crypto');
const privateKey=fs.readFileSync('8000013189_pri.pem','utf-8');
console.log(privateKey);

function encrypt(stuff) {
    let encryptString = Buffer.from(Buffer.from(stuff).toString('base64'));
    let result ='';
    while (encryptString.length > 117) {
        const chunk = encryptString.slice(0, 117);
        encryptString = encryptString.slice(117);
        result += crypto.privateEncrypt(privateKey, chunk).toString('hex');
    }
    result += crypto.privateEncrypt(privateKey, encryptString).toString('hex');
    return result;
}

console.log(encrypt("123456"));
```

4.7 pfx转pem

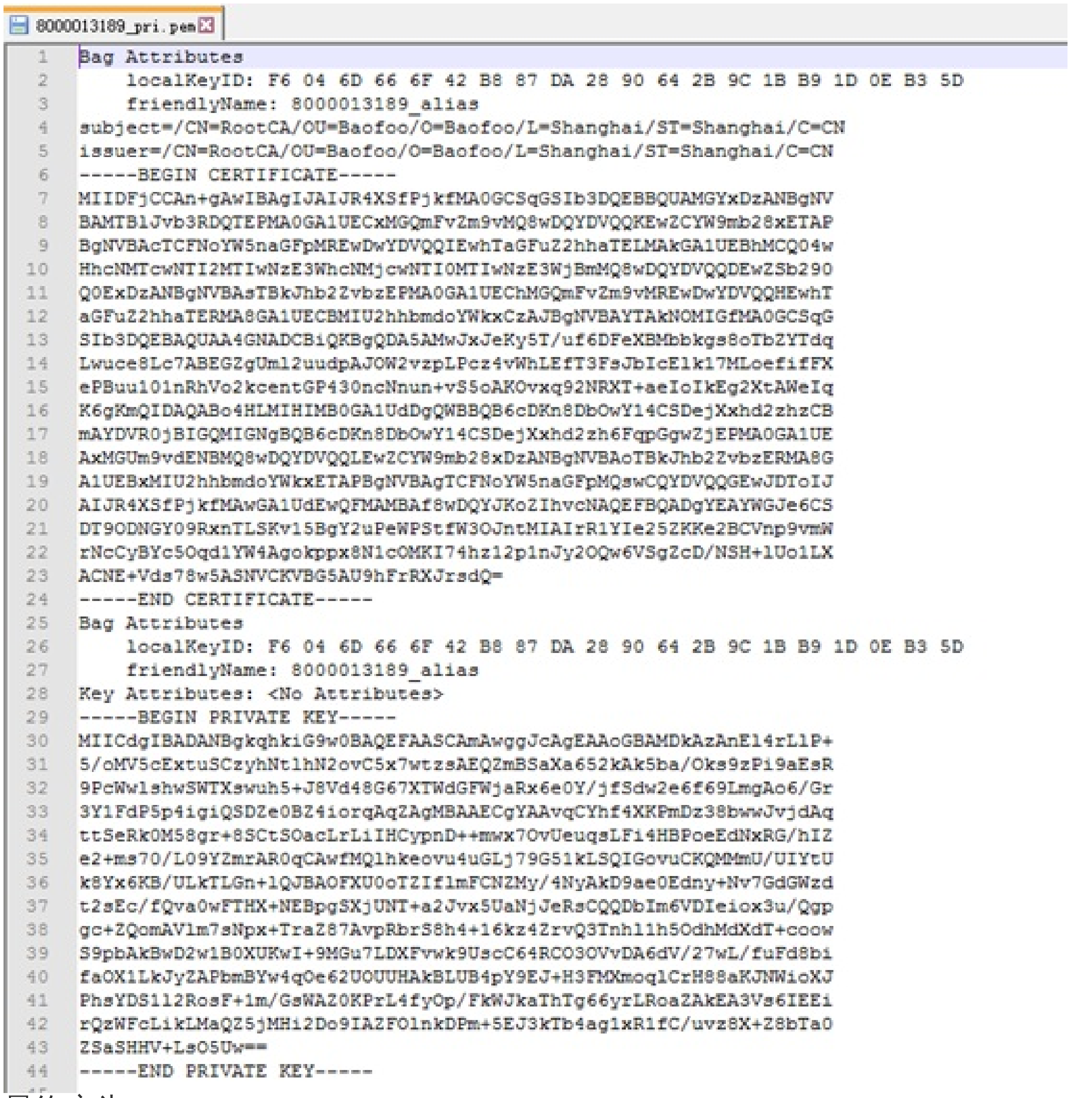
1.在openssl下使用命令：openssl pkcs12 -in 8000013189_pri.pfx -out 8000013189_pri.pem -nodes ，输入密码 例如：



```
D:\pythonTools\OpenSSL-Win32\bin>
D:\pythonTools\OpenSSL-Win32\bin>openssl pkcs12 -in 8000013189_pri.pfx -out 8000
013189_pri.pem  -nodes
Enter Import Password:
MAC verified OK

D:\pythonTools\OpenSSL-Win32\bin>
```

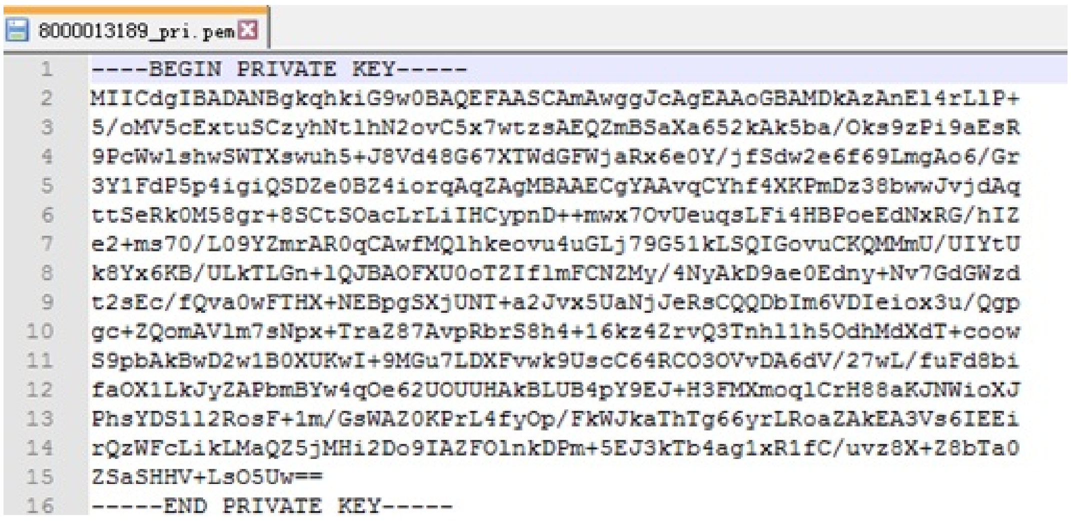
2.需要去掉多余的字符串
开始为：



8000013189_pri.pem

```
Bag Attributes
    localKeyID: F6 04 6D 66 6F 42 B8 87 DA 28 90 64 2B 9C 1B B9 1D 0E B3 5D
    friendlyName: 8000013189_alias
subject=/CN=RootCA/OU=Baofoo/O=Baofoo/L=Shanghai/ST=Shanghai/C=CN
issuer=/CN=RootCA/OU=Baofoo/O=Baofoo/L=Shanghai/ST=Shanghai/C=CN
-----BEGIN CERTIFICATE-----
MIIDFjCCAn+gAwIBAgIJAIJR4XSfPjkfMA0GCSqGSIb3DQEBBQUAMGYxDzANBgNV
BAMTBlJvb3RDQTEPMA0GA1UECxMGQmFvZm9vMQ8wDQYDVQQKEwZCYW9mb28xETAP
BgNVBAcTCFNoYW5naGFpMREwDwYDVQQIEwhTaGFuZ2hhaTELMAkGA1UEBhMCQ04w
HhcNMTcwNTI2MTIwNzE3WhcNMjcwNTI0MTIwNzE3WjBmMQ8wDQYDVQQDEwZSb290
Q0ExDzANBgNVBAsTBkJhb2ZvbzEPMA0GA1UEChMGQmFvZm9vMREwDwYDVQQHEwhT
aGFuZ2hhaTERMA8GA1UECBMIU2hhbmdoYWkxCzAJBgNVBAYTAkNOMIGfMA0GCSqG
SIb3DQEBAQUAA4GNADCBiQKBgQDA5AMwJxJeKy5T/uf6DFeXBMbbkgs8oTbZYTdq
Lwuce8Lc7ABEGZgUml2uudpAJOW2vzpLPcz4vWhLEfT3FsJbIcElk17MLoefifFX
ePBuu101nRhVo2kcentGP430ncNnun+vS5oAKOvxq92NRXT+aeIoIkEg2XtAWeIq
K6gKmQIDAQABo4HLMIHIMB0GA1UdDgQWBBQB6cDKn8DbOwY14CSDejXxhd2zhzCB
mAYDVR0jBIGQMIGNgBQB6cDKn8DbOwY14CSDejXxhd2zh6FqpGgwZjEPMA0GA1UE
AxMGUm9vdENBMQ8wDQYDVQQLEwZCYW9mb28xDzANBgNVBAoTBkJhb2ZvbzERMA8G
A1UEBxMIU2hhbmdoYWkxETAPBgNVBAgTCFNoYW5naGFpMQswCQYDVQQGEwJDToIJ
AIJR4XSfPjkfMAwGA1UdEwQFMAMBAf8wDQYJKoZIhvcNAQEFBQADgYEAYWGJe6CS
DT9ODNGY09RxnTLSKv15BgY2uPeWPStfW3OJntMIAIrR1YIe25ZKKe2BCVnp9vmW
rNcCyBYc5Oqd1YW4AgokppxBN1cOMKI74hz12p1nJy2OQw6VSgZcD/NSH+1Uo1LX
ACNE+Vds78w5ASNVCKVBG5AU9hFrRXJrsdQ=
-----END CERTIFICATE-----
Bag Attributes
    localKeyID: F6 04 6D 66 6F 42 B8 87 DA 28 90 64 2B 9C 1B B9 1D 0E B3 5D
    friendlyName: 8000013189_alias
Key Attributes: <No Attributes>
-----BEGIN PRIVATE KEY-----
MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBAMDkAzAnEl4rL1P+
5/oMV5cExtuSCzyhNtlhN2ovC5x7wtzsAEQZmBSaXa652kAk5ba/Oks9zPi9aEsR
9PcWwlshwSWTXswuh5+J8Vd48G67XTWdGFWjaRx6e0Y/jfSdw2e6f69LmgAo6/Gr
3Y1FdP5p4igiQSDZe0BZ4iorqAqZAgMBAAECgYAAvqCYhf4XKPmDz38bwwJvjdAq
ttSeRk0M58gr+8SCtSOacLrLiIHCypnD++mwx7OvUeuqsLFi4HBPoeEdNxRG/hIZ
e2+ms70/L09YZmrAR0qCAwfMQlhkeovu4uGLj79G51kLSQIGovuCKQMMmU/UIYtU
k8Yx6KB/ULkTLGn+lQJBAOFXU0oTZIflmFCNZMy/4NyAkD9ae0Edny+Nv7GdGWzd
t2sEc/fQva0wFTHX+NEBpgSXjUNT+a2Jvx5UaNjJeRsCQQDbIm6VDIeiox3u/Qgp
gc+ZQomAVlm7sNpx+TraZ87AvpRbr38h4+16kz4ZrvQ3Tnh11h5OdhMdXdT+coow
S9pbAkBwD2w1B0XUKwI+9MGu7LDXFvwk9UscC64RCO3OVvDA6dV/27wL/fuFd8bi
faOX1LkJyZAPbmBYw4qOe62UOUUHAkBLUB4pY9EJ+H3FMXmoqlCrH88aKJNWioXJ
PhsYDS112RosF+1m/GsWAZ0KPrL4fyOp/FkWJkaThTg66yrLRoaZAkEA3Vs6IEEi
rQzWFcLikLMaQZ5jMHi2Do9IAZFO1nkDPm+5EJ3kTb4ag1xR1fC/uvz8X+Z8bTa0
ZSaSHHV+Ls05Uw==
-----END PRIVATE KEY-----
```

最终变为：

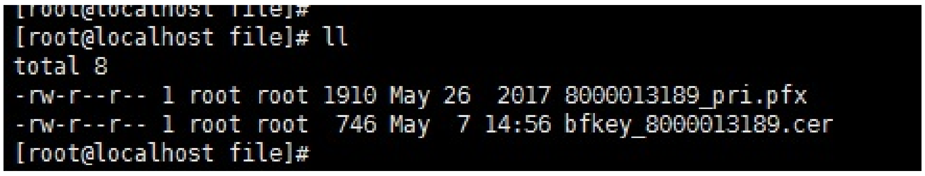


```
----BEGIN PRIVATE KEY-----
MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBAMDkAzAnEl4rLlP+
5/oMV5cExtuSCzyhNtlhN2ovC5x7wtzsAEQZmBSaXa652kAk5ba/Oks9zPi9aEsR
9PcWwlshwSWTXswuh5+J8Vd48G67XTWdGFWjaRx6e0Y/jfSdw2e6f69LmgAo6/Gr
3Y1FdP5p4igiQSDZe0BZ4iorqAqZAgMBAAECgYAAvqCYhf4XKPmDz38bwwJvjdAq
ttSeRk0M58gr+8SCtSOacLrLiIHCypnD++mwx7OvUeuqsLFi4HBPoeEdNxRG/hIZ
e2+ms70/L09YZmrAR0qCAwfMQlhkeovu4uGLj79G51kLSQIGovuCKQMMmU/UIYtU
k8Yx6KB/ULkTLGn+lQJBAOFXU0oTZIflmFCNZMy/4NyAkD9ae0Edny+Nv7GdGWzd
t2sEc/fQva0wFTHX+NEBpgSXjUNT+a2Jvx5UaNjJeRsCQQDbIm6VDIeiox3u/Qgp
gc+ZQomAVlm7sNpx+TraZ87AvpRbrS8h4+16kz4ZrvQ3Tnhl1h5OdhMdXdT+coow
S9pbAkBwD2w1B0XUKwI+9MGu7LDXFvwk9UscC64RCO3OVvDA6dV/27wL/fuFd8bi
faOX1LkJyZAPbmBYw4qOe62UOUUHAkBLUB4pY9EJ+H3FMXmoq1CrH88aKJNWioXJ
PhsYDS112RosF+1m/GsWAZ0KPrL4fyOp/FkWJkaThTg66yrLRoaZAkEA3Vs6IEEi
rQzWFcLikLMaQZ5jMHi2Do9IAZFOlnkDPm+5EJ3kTb4ag1xR1fC/uvz8X+Z8bTa0
ZSaSHHV+LsO5Uw==
-----END PRIVATE KEY-----
```

5.注意事项

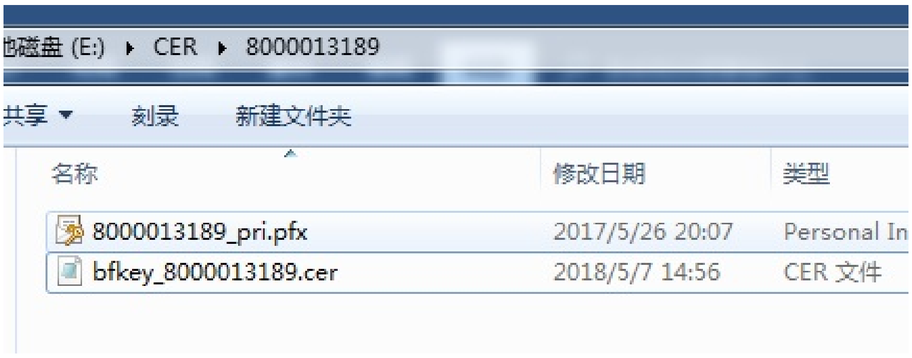
5.1 私钥（.pfx）文件存放路径

建议把私钥文件的存储路径放在独立的目录或单独的存放在文件服务器，这样的好处：
1、需要更换密钥不需要发布项目或者影响其他的配置。
2、对于需要用到maven打包的项目时不会因为打包改变文件导致加密失败。
例如：
Liunx系统: /data/file



```
[root@localhost file]# ll
total 8
-rw-r--r-- 1 root root 1910 May 26  2017 8000013189_pri.pfx
-rw-r--r-- 1 root root  746 May  7 14:56 bfkey_8000013189.cer
[root@localhost file]#
```

Windows：E:\CER\8000013189

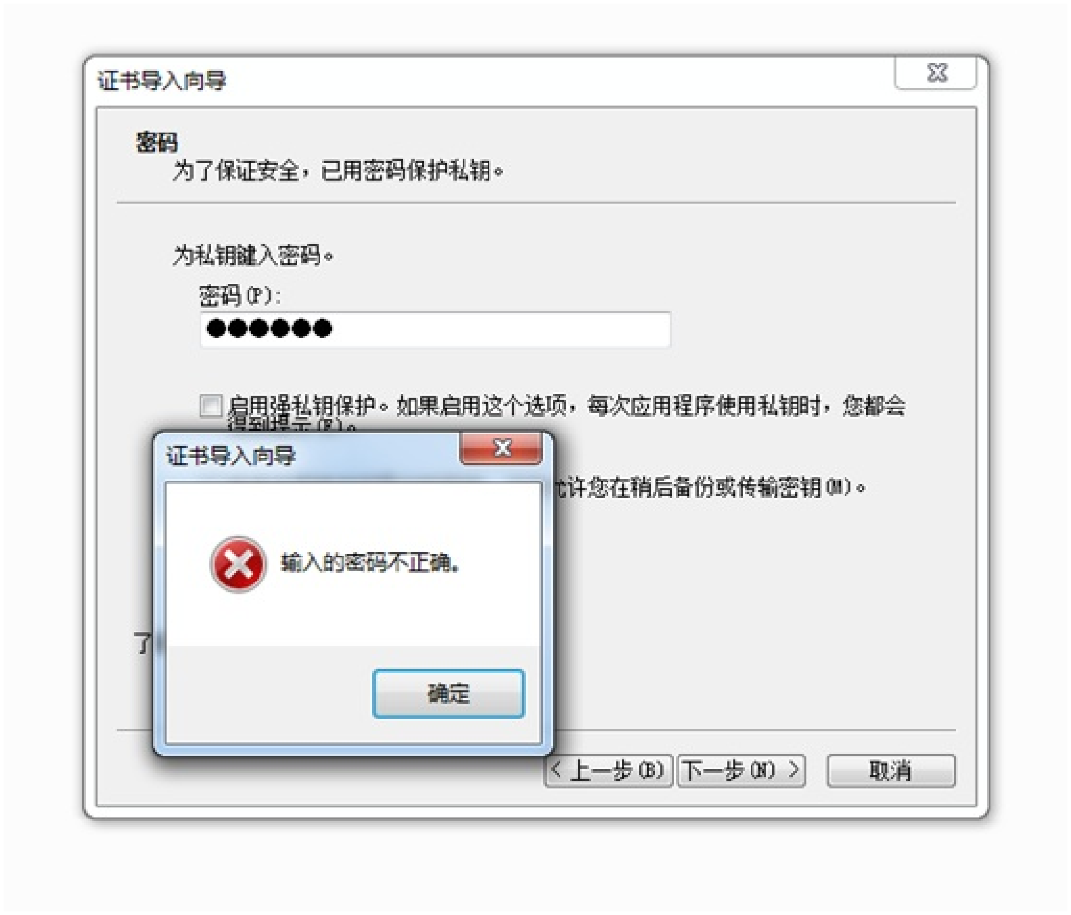


名称	修改日期	类型
8000013189_pri.pfx	2017/5/26 20:07	Personal In
bfkey_8000013189.cer	2018/5/7 14:56	CER 文件

5.2 加密失败或者加密内容为空

1.检查私钥文件和私钥密码是否是一对，如果不是一对不能加密出数据。

在windows下可以采用双击私钥文件输入私钥密码导入，如果能导入则为一致，如提示【输入的密码不正确】则不能加密成功，例如：


证书导入向导

密码
为了保证安全，已用密码保护私钥。

为私钥键入密码。
密码(P):
●●●●●●

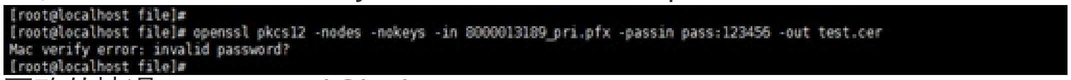
启用强私钥保护。如果启用这个选项，每次应用程序使用私钥时，您都会得到提示(E)。

…许您在稍后备份或传输密钥(M)。

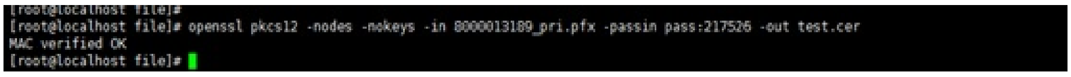
证书导入向导

输入的密码不正确。

在liunx下则可以在openssl 执行命令：openssl pkcs12 -nodes -nokeys -in 8000013189_pri.pfx -passin pass:123456 -out test.cer
错误的情况：Mac verify error: invalid password?


```
[root@localhost file]#
[root@localhost file]# openssl pkcs12 -nodes -nokeys -in 8000013189_pri.pfx -passin pass:123456 -out test.cer
Mac verify error: invalid password?
[root@localhost file]#
```

正确的情况：MAC verified OK


```
[root@localhost file]#
[root@localhost file]# openssl pkcs12 -nodes -nokeys -in 8000013189_pri.pfx -passin pass:217526 -out test.cer
MAC verified OK
[root@localhost file]# 
```

2.报错：**DerInputStream.getLength(): lengthTag=111, too big**和加密数据为空

可能在maven项目中密钥文件统一放到项目src/main/resources的某个目录下，打包部署运行中出现。原因为maven打包中没有排除密钥文件，证书就会被修改。

办法一：推荐把密钥文件存在单独的目录，不参与maven打包。
办法二：可以采用过滤证书文件的方式，但是不推荐。

```xml
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-resources-plugin</artifactId>
    <configuration>
        <encoding>UTF-8</encoding>
        <!-- 过滤后缀为cer、pfx的证书文件 -->
        <nonFilteredFileExtensions>
                    <nonFilteredFileExtension>cer</nonFilteredFileExtension>
        <nonFilteredFileExtension>pfx</nonFilteredFileExtension>
        </nonFilteredFileExtensions>
    </configuration>
</plugin>
```

5.3 ****不能为空

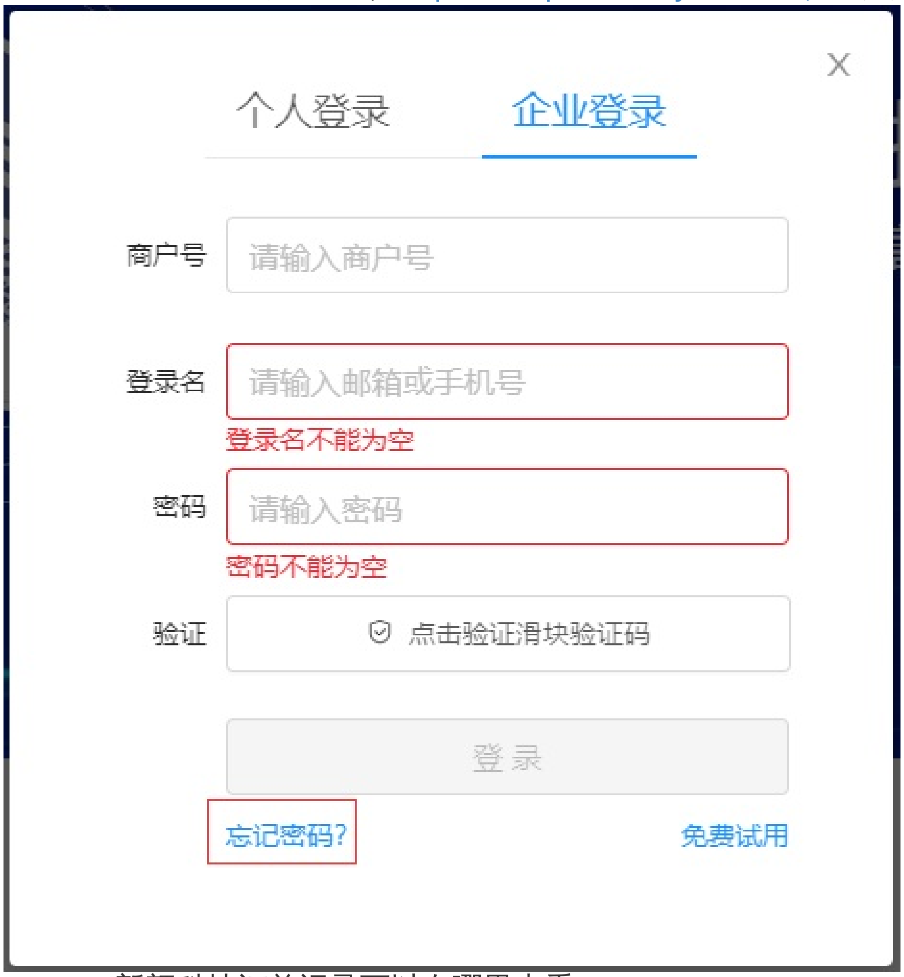
如果在调用中出现：商户号不能为空、加密数据不能为空、终端号不能为空、加密数据类型不能为空循环出现。这样的报错是由于post提交数据的方式不对导致新颜服务端没有接收到数据随机提示错误,具体参考文档和Demo。具体提交方式为：
认证类产品、信用评估类产品提交方式为：Content-Type: application/x-www-form-urlencoded;charset=utf-8
参考资料：https://blog.csdn.net/tycoon1988/article/details/40080691

5.4 新颜科技接口地址和ip

新颜科技正式环境地址和测试环境地址一般的区别二级域名不一致，具体可以查看接口文档。新颜科技不需要上报ip设置白名单。域名对应的ip具体请查看在线文档。
https://docs.qq.com/sheet/DSWF5TVdlZWhNZU9l?tab=BB08J2&coord=B14%24B14%240%240%241%240ss

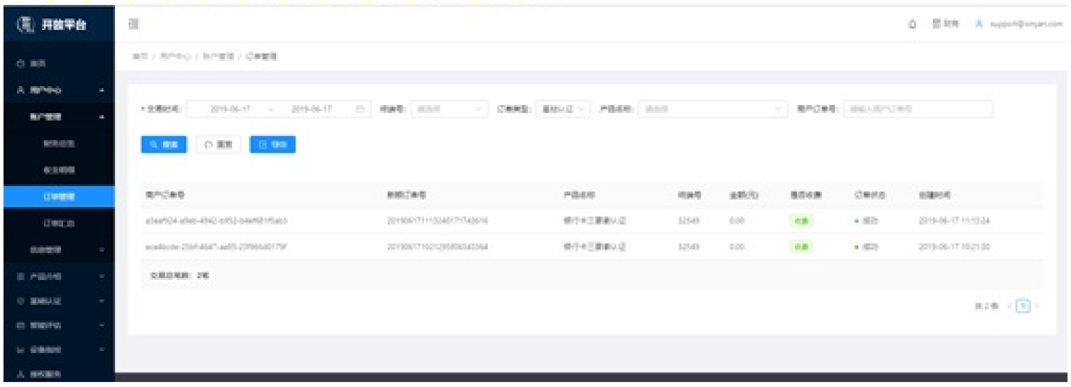
6. 常见Q&A

6.1 Q:新颜科技测试环境数据是真实校验么？
A:测试环境数据都是模拟的数据，为了前期实现业务逻辑代码走通流程。
6.2 Q:出现请求报文解析失败有哪些可能？
A:因为我们数据采用加密传输，解密失败为不能解密到传输过来的密文。一般可能的原因是：1、用测试环境密钥加密请求生产环境，生成环境没有上传解密用的公钥。2、对参数进行base64后有换行符，需要把换行符替换掉。3、加密的私钥和解密的公钥不是一对。
6.3 Q:商户号不存在有哪些情况？
A:一般为商户号与请求环境对应不上，先检查下商户号对应的相应环境，测试商户号只能请求测试环境，正式商户号只能请求正式环境 。
6.4 Q:为什么出现订单不能重复提交？
A:在同一商户号下，每笔请求的商户订单号需要保证唯一性，只能请求一次。
6.5 Q:商户号、终端号是什么？在哪里能找到？
A:商户号、终端号是新颜为了区别每个商户分配的系统参数。测试环境在Demo的配置文件中，正式环境会在签订协议后，发送到业务联系人的邮箱中。
6.6 Q:行业类别这个参数该怎么填？是否影响返回结果？
A:行业类别参数可以到接口文档上找到自己公司对应的类别，传入相应的参数，不影响返回结果。
6.7 Q:忘记登录密码可以在哪里找回？
A:可以登录新颜开放平台（https://open.xinyan.com） 修改密码和查看订单。



6.8 Q:新颜科技订单记录可以在哪里查看？

A：在开发平台（https://open.xinyan.com），账户管理---->订单管理查看。



7.关于加密-其他

AES加解密

AES的key需要到开放平台去找到密钥类型为AES的终端号，先保存在上传，具体路径如下：
首页->用户中心->信息设置->安全证书管理

java版本

```java
    /**
     * 密钥算法
     */
    private static final String KEY_ALGORITHM = "AES";

    /**
     * 加密/解密算法 / 工作模式 / 填充方式
     * PKCS5Padding填充方式
     */
    private static final String CIPHER_ALGORITHM = "AES/CBC/PKCS5Padding";

    private static final String IV_STRING = "DEVICE-AES000000";

    /**
     * 使用AES算法进行加密
     * @param source 加密源
     * @param key 秘钥-可在新颜开放平台获取，**详见：常见问题FAQ-AES加密密钥如何获取**
     * @return 密文
     */
    public static String encrypt(String source, String key){
        try {
            if (key == null || "".equals(source)) {
                return null;
            }
            byte[] keyBytes = key.getBytes();
            SecretKeySpec skeySpec = new SecretKeySpec(keyBytes, KEY_ALGORITHM);
            byte[] initParam = IV_STRING.getBytes();
            IvParameterSpec ivParameterSpec = new IvParameterSpec(initParam);
            Cipher cipher = Cipher.getInstance(CIPHER_ALGORITHM);
            cipher.init(Cipher.ENCRYPT_MODE, skeySpec,ivParameterSpec);
            byte[] encrypted = cipher.doFinal(source.getBytes(Charset.forName("UTF-8")));
            return Base64.encode(encrypted);
        } catch (Exception e) {
            log.warn("encrypt Exception source {}, key {}",source,key);
            return null;
        }
    }

    /**
     *  AES算法解密
     * @param source 密文
     * @param key 秘钥-可在新颜开放平台获取，**详见：常见问题FAQ-AES加密密钥如何获取**
     * @return 明文
     */
    public static String decrypt(String source, String key) {
        try {
            if (key == null || "".equals(source)) {
                return null;
            }
            byte[] keyBytes =key.getBytes();
            SecretKeySpec skeySpec = new SecretKeySpec(keyBytes, KEY_ALGORITHM);
            byte[] initParam = IV_STRING.getBytes();
            IvParameterSpec ivParameterSpec = new IvParameterSpec(initParam);
            Cipher cipher = Cipher.getInstance(CIPHER_ALGORITHM);
            cipher.init(Cipher.DECRYPT_MODE,skeySpec,ivParameterSpec);
            byte[] sourceBytes = new BASE64Decoder().decodeBuffer(source);
            byte[] original = cipher.doFinal(sourceBytes);
            return new String(original,Charset.forName("UTF-8"));
        } catch (Exception e) {
            log.error("decrypt Exception source {}, key {}",source,key);
            return null;
        }
    }
```

PHP版本 $IV=”DEVICE-AES000000”

```
    /**
     * AES/CBC/PKCS5Padding Encrypter
     *
     * @param $str
     * @param $key
     * @param $iv
     * @return string
     */
    function encryptNew($str, $key,$iv)
    {
        return base64_encode(openssl_encrypt($str, 'AES-128-CBC', $key, OPENSSL_RAW_DATA, $iv));
    }

    /**
     * AES/CBC/PKCS5Padding Decrypter
     *
     * @param $encryptedStr
     * @param $key
     * @param $iv
     * @return string
     */
    function decryptNew($encryptedStr,$key,$iv)
    {
        return openssl_decrypt(base64_decode($encryptedStr), 'AES-128-CBC', $key, OPENSSL_RAW_DATA, $iv);
    }
```

C#版本 IV=”DEVICE-AES000000”

```
    /**
     * 加密
     */
    public static string Encrypt(string toEncrypt, string key, string iv)
        {
            byte[] keyArray = UTF8Encoding.UTF8.GetBytes(key);
            byte[] ivArray = UTF8Encoding.UTF8.GetBytes(iv);
            byte[] toEncryptArray = UTF8Encoding.UTF8.GetBytes(toEncrypt);

            RijndaelManaged rDel = new RijndaelManaged();
            rDel.Key = keyArray;
            rDel.IV = ivArray;
            rDel.Mode = CipherMode.CBC;
            rDel.Padding = PaddingMode.PKCS7;

            ICryptoTransform cTransform = rDel.CreateEncryptor();
            byte[] resultArray = cTransform.TransformFinalBlock(toEncryptArray, 0, toEncryptArray.Length);

            return Convert.ToBase64String(resultArray, 0, resultArray.Length);
        }

        /**
         * 解密
         */
        public static string Decrypt(string toDecrypt, string key, string iv)
        {
            byte[] keyArray = UTF8Encoding.UTF8.GetBytes(key);
            byte[] ivArray = UTF8Encoding.UTF8.GetBytes(iv);
            byte[] toEncryptArray = Convert.FromBase64String(toDecrypt);

            RijndaelManaged rDel = new RijndaelManaged();
            rDel.Key = keyArray;
            rDel.IV = ivArray;
            rDel.Mode = CipherMode.CBC;
            rDel.Padding = PaddingMode.PKCS7;

            ICryptoTransform cTransform = rDel.CreateDecryptor();
            byte[] resultArray = cTransform.TransformFinalBlock(toEncryptArray, 0, toEncryptArray.Length);

            return UTF8Encoding.UTF8.GetString(resultArray);
        }
```

MD5加密(32位小写)

java实现

```java
package com.xinyan.credit.util;

import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.text.SimpleDateFormat;
import java.util.Arrays;
import java.util.Date;
import sun.misc.BASE64Encoder;

public class MD5Utils {

    private static final String HEX_NUMS_STR = "0123456789ABCDEF";
    private static final Integer SALT_LENGTH = Integer.valueOf(12);
    private static final String KEY_MD5 = "MD5";
    private static final String[] STR_DIGITS = { "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "a", "b", "c", "d",
            "e", "f" };

    public static byte[] hexStringToByte(String hex) {
        int len = hex.length() / 2;
        byte[] result = new byte[len];
        char[] hexChars = hex.toCharArray();
        for (int i = 0; i < len; i++) {
            int pos = i * 2;
            result[i] = (byte) ("0123456789ABCDEF".indexOf(hexChars[pos]) << 4
                    | "0123456789ABCDEF".indexOf(hexChars[(pos + 1)]));
        }

        return result;
    }

    public static String byteToHexString(byte[] b) {
        StringBuffer hexString = new StringBuffer();
        for (int i = 0; i < b.length; i++) {
            String hex = Integer.toHexString(b[i] & 0xFF);
            if (hex.length() == 1) {
                hex = new StringBuilder().append('0').append(hex).toString();
            }
            hexString.append(hex.toUpperCase());
        }
        return hexString.toString();
    }

    public static boolean validPassword(String password, String passwordInDb)
            throws NoSuchAlgorithmException, UnsupportedEncodingException {
        byte[] pwdInDb = hexStringToByte(passwordInDb);

        byte[] salt = new byte[SALT_LENGTH.intValue()];

        System.arraycopy(pwdInDb, 0, salt, 0, SALT_LENGTH.intValue());

        MessageDigest md = MessageDigest.getInstance("MD5");

        md.update(salt);

        md.update(password.getBytes("UTF-8"));

        byte[] digest = md.digest();

        byte[] digestInDb = new byte[pwdInDb.length - SALT_LENGTH.intValue()];

        System.arraycopy(pwdInDb, SALT_LENGTH.intValue(), digestInDb, 0, digestInDb.length);

        return Arrays.equals(digest, digestInDb);
    }

    public static String getEncryptedPwd(String password)
            throws NoSuchAlgorithmException, UnsupportedEncodingException {
        byte[] pwd = null;

        SecureRandom random = new SecureRandom();

        byte[] salt = new byte[SALT_LENGTH.intValue()];

        random.nextBytes(salt);

        MessageDigest md = null;

        md = MessageDigest.getInstance("MD5");

        md.update(salt);

        md.update(password.getBytes("UTF-8"));

        byte[] digest = md.digest();

        pwd = new byte[digest.length + SALT_LENGTH.intValue()];

        System.arraycopy(salt, 0, pwd, 0, SALT_LENGTH.intValue());

        System.arraycopy(digest, 0, pwd, SALT_LENGTH.intValue(), digest.length);

        return byteToHexString(pwd);
    }

    public static String encryptMD5(byte[] data) {
        try {
            MessageDigest md5 = MessageDigest.getInstance("MD5");
            md5.update(data);
            return byteToString(md5.digest());
        } catch (Exception e) {
        }
        return "";
    }

    private static String byteToArrayString(byte bByte) {
        int iRet = bByte;
        if (iRet < 0) {
            iRet += 256;
        }
        int iD1 = iRet / 16;
        int iD2 = iRet % 16;
        return new StringBuilder().append(STR_DIGITS[iD1]).append(STR_DIGITS[iD2]).toString();
    }

    private static String byteToString(byte[] bByte) {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < bByte.length; i++) {
            sb.append(byteToArrayString(bByte[i]));
        }
        return sb.toString();
    }

    public static String encryptMD5(String inputText) {
        return encrypt(inputText, "MD5");
    }

    private static String encrypt(String inputText, String algorithmName) {
        if ((inputText == null) || ("".equals(inputText.trim()))) {
            return "";
        }
        if ((algorithmName == null) || ("".equals(algorithmName.trim())))
            algorithmName = "md5";
        try {
            MessageDigest m = MessageDigest.getInstance(algorithmName);
            m.update(inputText.getBytes("UTF-8"));
            byte[] s = m.digest();
            return hex(s);
        } catch (NoSuchAlgorithmException e) {
            error("MD5加密异常:" + e);
            return null;
        } catch (UnsupportedEncodingException e) {
            error("MD5加密异常:{}" + e);
        }
        return null;
    }

    private static String hex(byte[] arr) {
        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < arr.length; i++) {
            sb.append(Integer.toHexString(arr[i] & 0xFF | 0x100).substring(1, 3));
        }

        return sb.toString();
    }

    public static String ecodeByMD5(String originstr) {
        String result = null;

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

        if (originstr != null) {
            try {
                MessageDigest md = MessageDigest.getInstance("MD5");

                byte[] source = originstr.getBytes("utf-8");

                md.update(source);

                byte[] tmp = md.digest();

                char[] str = new char[32];

                int i = 0;
                for (int j = 0; i < 16; i++) {
                    byte b = tmp[i];

                    str[(j++)] = hexDigits[(b >>> 4 & 0xF)];

                    str[(j++)] = hexDigits[(b & 0xF)];
                }

                result = new String(str);
            } catch (NoSuchAlgorithmException e) {
                return null;
            } catch (UnsupportedEncodingException e) {
                return null;
            }
        }

        return result;
    }

    public static String encryptMd5(String originStr) {
        String result = null;
        if (null != originStr) {
            try {
                MessageDigest md = MessageDigest.getInstance("md5");
                byte[] mess = originStr.getBytes("utf-8");
                md.reset();
                byte[] hash = md.digest(mess);
                BASE64Encoder encoder = new BASE64Encoder();
                result = encoder.encode(hash);
            } catch (Exception e) {
                error("encryptMd5 error:" + e);
            }
        }
        return result;
    }

    public static void error(String msg) {
        System.out.println(new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(new Date()) + "\t: " + msg);
    }

    public static void main(String[] args) {

        String idno = MD5Utils.encryptMD5("110102200101017072");//1b0b0dbf7650414b7117fcaadce9ddad
        System.out.println(idno);
        String name = MD5Utils.encryptMD5("李四");//36c942351ec9cc3ad124e288a5c9cf0b
        System.out.println(name);
    }
}
```