



银联商务 POS 通安全支付插件 商户接入接口规范

第一部分 移动端接口 V1.2.2

技术支持: qmf_support@chinaums.com

中国银联商务有限公司

2015. 12

版本控制信息

版本	日期	拟稿和修改	说明
V1.0.0	2015/2/10	初稿	
V1.0.1	2015/3/3	修改	调整联合登录接口、调整
	2015/3/9	修改	调整文档格式
V1.1.0	2015/3/13	新增	增加远程支付、签名机制
	2015/3/17	修改	远程支付：增加商户通知、单笔订单查询
V1.2.0	2015/10/17	修改	公私钥对格式由模+指数统一调整为公私 X509，私钥 PKCS8 DES 密钥加解密优化升级 待签名串由指定顺序转按字典顺序排序
V1.2.1	2016/1/14	新增	远程快捷支持代理商交易模式
V1.2.2	2016/2/18	新增	Apple Pay 调用接口

本文档中的所有内容为银联商务有限公司专属所有。未经银联商务有限公司的明确书面许可，任何组织或个人不得以任何目的、任何形式及任何手段复制或传播本文档部分或全部内容。

目 录

目 录.....	3
1.1 概述.....	5
1.2 适用范围.....	5
1.3 流程说明.....	5
2、 参数规范说明.....	5
2.1 序号.....	5
2.2 数据项.....	6
2.3 类型.....	6
2.4 长度.....	6
2.5 输入/选择.....	6
2.6 备注.....	7
3、 远程支付调用接口.....	7
3.1 Android 客户端调用远程支付插件.....	7
3.1.1 Android 客户端 apk 签名信息检测.....	7
3.1.2 服务 API 定义.....	8
3.1.3 绑定服务.....	8
3.1.4 解绑服务.....	9
3.1.5 返回信息.....	9
3.1.6 调用方法.....	9
3.1.7 参数说明.....	11
3.1.8 签名机制.....	12
3.2 iOS 客户端调用远程支付插件.....	15
3.2.1 调用方法.....	15
3.2.2 参数说明.....	16
3.2.3 备注.....	17
3.2.4 签名机制.....	17
4、 Apple Pay 调用接口.....	19
4.1 iOS 客户端调用 Apple Pay 接口.....	19
4.1.1 Apple Pay 支付接口&预授权接口.....	19
4.1.2 Apple Pay 商户通知.....	22
4.1.3 Apple Pay 备注.....	22
4.2 服务端调用 Apple Pay 接口.....	22
4.2.1 总则.....	22
4.2.2 获取 accessToken 流程.....	23

4.2.3	业务接口	24
5、	商户通知	30
5.1	接口介绍	30
5.2	使用方法	30
5.2.1	发送及接收方式	30
5.2.2	API 参数	30
5.2.3	回调通知 URL	32
5.2.4	3DES 密钥	32
5.2.5	RSA 公钥（2048 位）	32
5.2.6	商户接收设置	33
5.2.7	商户返回	35
6、	单笔订单查询	36
6.1	规则介绍	36
6.2	界面原型	36
6.3	请求、响应参数说明	37
6.4	POST 消息体	39
6.5	消息签名	39
7、	RSA 公私钥	40
7.1	OpenSSL 工具安装	40
7.2	RSA 私钥及公钥生成	40
8、	附录	42
8.1	订单状态表	42
8.2	支付状态表	43
8.3	货币代码表	43
8.4	Android 客户端 APK 签名表	43

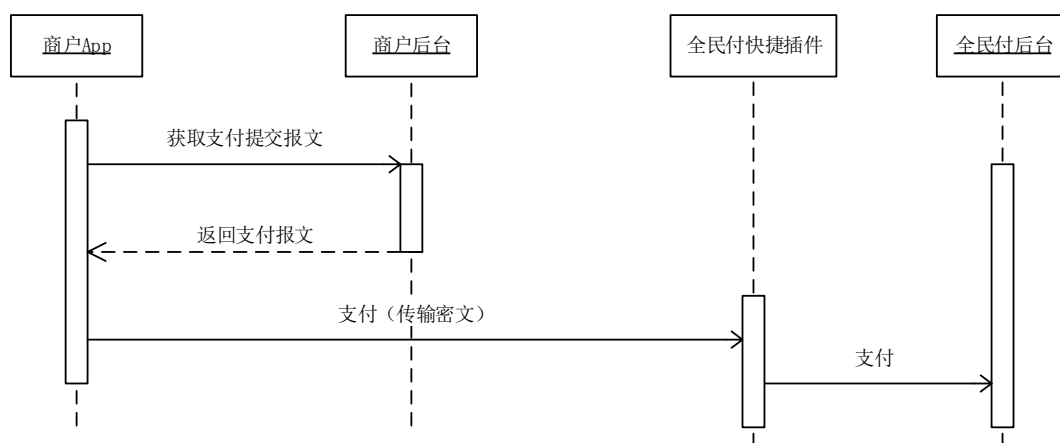
1.1 概述

本接口定义第三方接入银联商务全民付快捷插件的接口标准，主要是对第三方传入手机号、商户号、密钥验签等要素的定义。

1.2 适用范围

此接口适用第三方接入银联商务全民付快捷插件的开发。

1.3 流程说明



2、 参数规范说明

2.1 序号

报文中每个字段的顺序

2.2 数据项

报文中此字段所代表的属性

2.3 类型

报文中此字段的类型

字符	含义
A	字母字符，A至Z，a至z
N	数值，0至9，右靠，首位有效数字前填零。若表示人民币金额，则最右二位为角、分。
P	填充字符，如空格。
S	特殊符号。
C	字符类型
An	字母和数字字符
As	字母和特殊字符
Ns	数字和特殊字符
Ans	字母、数字和特殊字符。
MM	月份，01至12。
DD	日期，01至31。
YY	年份，00至99。
hh	时，00至23。
mm	分，00至59。
ss	秒，00至59。

2.4 长度

报文中此字段允许的最大长度

2.5 输入/选择

报文中此字段输入要求：

M 此字段必须出现，且不能为空

- O 此字段可选
- C 在特定环境下，此字段必须出现，且不为空

2.6 备注

对字段的一些解释

3、 远程支付调用接口

3.1 Android 客户端调用远程支付插件

3.1.1 Android 客户端 apk 签名信息检测

在调用 Android 的客户端插件之前，为了防止访问伪造的插件 APK，需要对客户端 apk 进行签名信息校验，正确客户端 APK 签名信息请查阅 8.4 Android 客户端 APK 签名表，获取 apk 签名信息的示例代码如下：

```
public Signature getPackageSignature(Context context, String packageName) {  
  
    PackageManager pm = context.getPackageManager();  
  
    List<PackageInfo> apps =  
  
    pm.getInstalledPackages(PackageManager.GET_SIGNATURES);  
  
    Iterator<PackageInfo> it = apps.iterator();  
  
    while(it.hasNext()) {  
  
        PackageInfo info = it.next();  
  
        if(info.packageName.equals(packageName)) {  
  
            return info.signatures[0];  
  
        }  
  
    }  
  
}
```

```
    }  
  
    }  
  
    return null;  
}
```

3.1.2 服务 API 定义

IUmsQuickPayService.aidl 文件:

```
package com.chinaums.pppay.quickpay.service;  
  
import android.os.Bundle;  
  
import com.chinaums.pppay.quickpay.service.IUmsQuickPayResultListener;  
  
oneway interface IUmsQuickPayService {  
  
    void payOrder(in Bundle args, in IUmsQuickPayResultListener listener);  
  
}
```

回调接口定义 IUmsQuickPayResultListener.aidl 文件:

```
package com.chinaums.pppay.quickpay.service;  
  
oneway interface IUmsQuickPayResultListener {  
  
    void umsServiceResult(in Bundle result);  
  
}
```

3.1.3 绑定服务

使用银联商户各服务接口前，首先商户应用要绑定银联商务的服务。

举例代码:

```
private IUmsQuickPayService mUmsQuickPayService;  
  
private ServiceConnection mConnection = new ServiceConnection() {
```



```
public void onServiceConnected(ComponentName className, IBinder service) {  
    mUmsQuickPayService = IUmsQuickPayService.Stub.asInterface(service);  
}  
  
public void onServiceDisconnected(ComponentName className) {  
    mUmsQuickPayService = null;  
}  
};  
  
private void bindQuickPayService() {  
    Intent intent = new Intent();  
    intent.setClassName("com.chinaums.pppay",  
        "com.chinaums.pppay.quickpay.service.QuickPayService");  
    bindService(intent, mConnection, Context.BIND_AUTO_CREATE);  
}
```

3.1.4 解绑服务

商户退出应用后，要解绑服务：
举例代码：

```
private void unbindQuickPayService() {  
    unbindService(mConnection);  
}
```

3.1.5 返回信息

在每个服务接口回调的 bundle 里面，都有 resultStatus 字段和 resultInfo 字段。resultStatus 值可能为 success、fail、timeout、cancel、nosupport。其他返回状态各个接口自行定义。

3.1.6 调用方法

分为两步：首先绑定服务，然后调用服务的接口。调用订单支付接口进行支

付。

```
// 定义订单支付回调

class PayOrderResultListener extends IUmsQuickPayResultListener.Stub {

    @Override

    public void umsServiceResult (final Bundle result) throws RemoteException {

        runOnUiThread(new Runnable() {

            public void run() {

                String payStatus = result.getString("payStatus");

                if ("success".equals(payStatus) && !"success".equals(printStatus)) {

                    Toast.makeText(MainActivity.this, "支付成功，但请重新签名",

                        Toast.LENGTH_SHORT).show();

                } else if ("success".equals(payStatus)) {

                    Toast.makeText(MainActivity.this, "支付成功",

                        Toast.LENGTH_SHORT).show();

                } else {

                    String resultInfo = result.getString("resultInfo");

                    Toast.makeText(MainActivity.this, "支付失败：" + resultInfo,

                        Toast.LENGTH_SHORT).show();

                }

            }

        });

    }

}

public void payOrder() {

    Bundle args = new Bundle();

    args.putString("merchantId ", "123456789111117");

    args.putString("merchantUserId", "mMerchantUserId");

    args.putString("mobile", "12345678901");

    args.putString("merOrderId", "622014040126637643");

}
```

```

args.putString("amount", "1");

args.putString("sign", "mSign");

args.putString("mode", "2");

args.putString("notifyUrl", "mNotifyUrl");

args.putString("signType", "RSA");


try {

    mUmsQuickPayService.payOrder(args, new PayOrderResultListener());

} catch (RemoteException e) {

    e.printStackTrace();

}

}

```

3.1.7 参数说明

3.1.7.1 启动参数

参数名	数据项	类型	长度	输入/选择	备注
agentMerchantId	代理商户号	N		O	由银商分配
amount	交易金额	N		M	支付金额，单位为分
merOrderId	商户订单号	An		M	
merchantId	商户号	N		M	由银商分配
merchantUserId	商户唯一id	An		M	商户自己系统用户号，字母和数字构成
mobile	用户手机号	N	11	M	11 位数字
mode	启动模式	N	1	M	插件模式值（2 远程快捷模式）
notifyUrl	支付成功	Ans		M	支付成功的通知接口

	通知接口				(详见 4.商户通知)
sign	签名	C		M	RSA，用第三方商户 privateKey 对参数字符串拼接后进行签名(详见 3.1.8 签名机制)
signType	签名方式	C		M	签名方式值(值为 RSA;表示参数以字典顺序排列后签名)(本参数不做验签)

3.1.7.2 返回参数

参数名	数据项	类型	长度	输出/选择	备注
errCode	错误码	C		C	标识接口请求是否成功，若返回“0000”则标识成功
errInfo	错误信息	C		C	接口返回的状态描述

若根据第三方商户传入的信息进行联合登录失败，商户可在 onActivityResult 方法中根据插件返回的 resultCode (RESULT_OK 为-1 时) 和 Intent 封装的 errCode 和 errInfo 信息进行提示，主要 errCode 和 errInfo 对应的场景如下：

- a、errCode="0000",errInfo="支付成功";
- b、errCode="1000",errInfo="用户取消支付";
- c、errCode="1001",errInfo="手机号不能为空";
- d、errCode="1002",errInfo="手机号格式不正确";
- e、errCode="1003",errInfo="商户号不能为空";
- f、errCode="1004",errInfo="商户用户号不能为空";
- g、errCode="1005",errInfo="签名信息不能为空";
- h、errCode="1006",errInfo="网络连接超时";
- i、errCode="8002",errInfo="验签失败";
- j、errCode="8003",errInfo="无此商户";

其他异常：

- k、errCode="2000",errInfo 为其他异常信息;

注意：订单支付状态请以后台通知为准，前端返回状态仅作页面展示所用。

3.1.8 签名机制

商户应用通过调用银商支付插件接口来完成支付功能，为保证交易安全性，支付接口采用 RSA 验签机制，银商通过商户提供的公钥验证消息来源。

3.1.8.1 待签名字符串

商户需要额外提供签名串 sign，签名串由商户自己的私钥对支付参数进行签名。支付参数详见列表 3.1.7.2（区分大小写），key 与 value 值以“=”符号连接，不同的参数对以“&”符号拼接。最终将所有的参数（sign 除外）按 key 的字典排序，组成待签名字符串。示例：

```
agentMerchantId=898330572300000&amount=100&merOrderId=1442803121615&merchantId=898330572300006&merchantUserId=00000001&mobile=18601770573&mode=2&notifyUrl=http://172.16.26.178:8080/connectDemo/NotifyOperServlet
```

3.1.8.2 支付参数

参数	参数名称	类型	必填	描述	范例
agentMerchantId	代理商户号	String(15)	否	银商分配商户号	898330572300000
amount	交易金额	Long	是	支付金额，单位为分	100
merOrderId	商户订单号	String(64)	是	商户系统唯一订单号	1442803121615
merchantId	商户号	String(15)	是	银商分配商户号	898330572300006
merchantUserId	商户唯一 id	String	是	商户系统自己的用户号，字母和数字构成	00000001
mobile	用户手机号	String(11)	是	11 位手机号	18601770573
mode	启动模式	String(1)	是	模式值（1：近场模	2

参数	参数名称	类型	必填	描述	范例
				式 2： 远程快捷模式)	
notifyUrl	通知地址	String(256)	否	支付成功的通知服务地址	http://172.16.26.178:8080/connectDemo/NotifyOperServlet
sign	签名	String(172)	是	商户请求参数的签名串	详见实例 3.1.8.3
signType	签名格式	String	是	商户请求参数的签名串格式	RSA

3.1.8.3 签名示例

使用 OpenSSL 的加密算法对待签名字符串进行签名并进行 **Base64** 编码，以下为 Java 代码的签名方法代码片段，供参考：

```
/**
 * 用私钥对信息生成数字签名
 * @param data 待签名数据
 * @param privateKey 私钥
 * @return
 * @throws Exception
 */
public static String sign(byte[] data, String privateKey) throws Exception {
    byte[] keyBytes = Base64Utils.decode(privateKey);
    PKCS8EncodedKeySpec pkcs8KeySpec = new PKCS8EncodedKeySpec
(keyBytes);
    //加密算法：RSA
    KeyFactory keyFactory = KeyFactory.getInstance("RSA");
    PrivateKey privateK = keyFactory.generatePrivate(pkcs8KeySpec);
    //用 SHA 算法签名，用 RSA 算法加密
    Signature signature = Signature.getInstance("SHA1withRSA");
    signature.initSign(privateK);
    signature.update(data);
    //对签名进行 Base64 编码，变为可读的字符串
```

```
        return new String(Base64Utils.encode(signature.sign()));
    }
}
```

提示：私钥由商户生成，如何生成私钥，请参考 [6.2](#) 内容。

3.2 iOS 客户端调用远程支付插件

3.2.1 调用方法

集成 iOS 版支付插件时，请先导入插件 SDK 所依赖库，插件依赖库具体请参考“3.2.3 备注”。

libUMSFastPay.h 文件中通过类方法 `payOrder: callback:` 来调起支付插件，示例代码如下：

```
[libUMSFastPay payOrder:orderString callback:^(NSString *errCode, NSString *errInfo)
{
    NSLog(@"errCode = %@, errInfo = %@", errCode, errInfo); //返回的支付结果
}];
```

插件会在完成支付结果解析后在上述 `TransactionResultBlock` 中统一回调。

注：`orderString` 是由商户传入插件时所需的支付参数按 `key="value"` 形式以 `&` 拼接而成的字符串，如：

```
NSString *orderString= [NSString
stringWithFormat:@"%amount=%@"&merchantId=%@"&merchantUserId=%@"&merOrderId=%@"&m
obile=%@"&mode=%@"&notifyUrl=%@"&sign=%@"", @"1", @"123451234512345", @"12345123451
", @"622014040126637643", @"15600005555", @"1", @"http://notify.msp.hk/notify.htm", @"0aDoW+XT/yP
GYx"];
```

对 `orderString` 按 `key="value"` 形式以 `&` 拼接时，顺序可以不固定，上面的 `key` 值及 `value` 值仅供参考，具体支付参数详细说明请参阅“3.2.2.1 启动参数”。`callback` 回调参数详细说明请参考“3.2.2.2 回调参数”。

3.2.2 参数说明

3.2.2.1 启动参数

参数名	数据项	类型	长度	输入/选择	备注
agentMerchantId	代理商户号	N		O	由银商分配
amount	交易金额	N		M	支付金额，单位为分
merOrderId	商户订单号	An		M	
merchantId	商户号	N		M	由银商分配
merchantUserId	商户唯一id	An		M	商户自己系统用户号，字母和数字构成
mobile	用户手机号	N	11	M	11 位数字
mode	启动模式	N	1	M	插件模式值（2 远程快捷模式）
notifyUrl	支付成功通知接口	Ans		M	支付成功的通知接口（详见 4.商户通知 ）
sign	签名	C		M	RSA，用第三方商户 privateKey 对参数字符串拼接后进行签名（详见 3.2.4 签名机制）
signType	签名方式	C		M	签名方式值（值为 RSA；表示参数以字典顺序排列后签名）（ 本参数不做验签 ）

3.2.2.2 回调参数

参数名	数据项	类型	长度	输出/选择	备注
errCode	错误码	C		C	标识接口请求是否成功，若返回“0000”则标识成功
errInfo	错误信息	C		C	接口返回的状态描述

若根据第三方商户传入的信息进行联合登录失败，商户可在 TransactionResultBlock 回调中根据插件返回的 errCode 和 errInfo 信息进行提示，主要 errCode 和 errInfo 对应的场景如下：

a、errCode="0000",errInfo="支付成功";

- b、errCode="1000",errInfo="用户取消支付";
- c、errCode="1001",errInfo="手机号不能为空";
- d、errCode="1002",errInfo="手机号格式不正确";
- e、errCode="1003",errInfo="商户号不能为空";
- f、errCode="1004",errInfo="商户用户号不能为空";
- g、errCode="1005",errInfo="签名信息不能为空";
- h、errCode="1006",errInfo="网络连接超时";
- i、errCode="8002",errInfo="验签失败";
- j、errCode="8003",errInfo="无此商户";

其他异常:

- k、errCode="2000",errInfo 为其他异常信息;

注意：订单支付状态请以后台通知为准，前端返回状态仅作页面展示所用。

3.2.3 备注

集成全民付 IOS 插件静态库时，如遇到集成冲突问题，可参考如下解决方法：

Targets——>Build Settings——>Linking 下的 Other Linker Flags 改为-ObjC

3.2.4 签名机制

商户应用通过调用银商支付插件接口来完成支付功能，为保证交易安全性，支付接口采用 RSA 验签机制，银商通过商户提供的公钥验证消息来源。

3.2.4.1 待签名字符串

商户需要额外提供签名串 sign，签名串由商户自己的私钥对支付参数进行签名。支付参数详见列表 4.5.4.2（区分大小写），key 与 value 值以“=”符号连接，不同的参数对以“&”符号拼接。最终将所有的参数（sign 及 signType 除外）按 key 的字典排序，组成待签名字符串。示例：

```
agentMerchantId=898330572300000&amount=100&merOrderId=1442803121615&merchantId=898330572300006&merchantUserId=00000001&mobile=18601770573&mode=2&notifyUrl=http://172.16.26.178:8080/connectDemo/NotifyOperServlet
```

3.2.4.2 支付参数

参数	参数名称	类型	必填	描述	范例
----	------	----	----	----	----

参数	参数名称	类型	必填	描述	范例
agentMerchantId	代理商户号	String(15)	否	银商分配商户号	898330572300000
amount	交易金额	Long	是	支付金额, 单位为分	100
merOrderId	商户订单号	String(64)	是	商户系统唯一订单号	1442803121615
merchantId	商户号	String(15)	是	银商分配商户号	898330572300006
merchantUserId	商户唯一 id	String	是	商户系统自己的用户号, 字母和数字构成	00000001
mobile	用户手机号	String(11)	是	11 位手机号	18601770573
mode	启动模式	String(1)	是	模式值 (1: 近场模式 2: 远程快捷模式)	2
notifyUrl	通知地址	String(256)	否	支付成功的通知服务地址	http://172.16.26.178:8080/connectDemo/NotifyOperServlet
sign	签名	String(172)	是	商户请求参数的签名串	详见实例 3.2.4.3
signType	签名格式	String	是	商户请求参数的签名串格式	RSA

3.2.4.3 签名示例

使用 OpenSSL 的加密算法对待签名字符串进行签名并进行 **Base64** 编码，以下为 Java 代码的签名方法代码片段，供参考：

```
/**
 * 用私钥对信息生成数字签名
 * @param data 待签名数据
 * @param privateKey 私钥
 * @return
 * @throws Exception
 */
public static String sign(byte[] data, String privateKey) throws Exception {
    byte[] keyBytes = Base64Utils.decode(privateKey);
    PKCS8EncodedKeySpec pkcs8KeySpec = new PKCS8EncodedKeySpec
(keyBytes);
    //加密算法：RSA
    KeyFactory keyFactory = KeyFactory.getInstance("RSA");
    PrivateKey privateK = keyFactory.generatePrivate(pkcs8KeySpec);
    //用 SHA 算法签名，用 RSA 算法加密
    Signature signature = Signature.getInstance("SHA1withRSA");
    signature.initSign(privateK);
    signature.update(data);
    //对签名进行 Base64 编码，变为可读的字符串
    return new String(Base64Utils.encode(signature.sign()));
}
```

提示：私钥由商户生成，如何生成私钥，请参考 [6.2](#) 内容。

4、 Apple Pay 调用接口

4.1 iOS 客户端调用 Apple Pay 接口

4.1.1 Apple Pay 支付接口&预授权接口

4.1.1.1 集成插件时，请先导入插件 **SDK** 所依赖库，插件依赖库具体请参考 [“4.1.3 Apple Pay 备注”](#)

libUMSFastPay.h 文件中通过类方法 payOrder: callBack:来调起支付插件，示

例代码如下：

```
[libUMSFastPay payOrder:orderString callBack:^(NSString *errCode, NSString *errInfo)
{
    NSLog(@"errCode = %@, errInfo = %@", errCode, errInfo); //返回的支付结果
}];
```

插件会在完成支付结果解析后在上述 TransactionResultBlock中统一回调。

注：orderString是由商户传入插件时所需的支付参数按key="value"形式以&拼接而成的字符串，如：

```
amount="1"&appleMerchantId="merchant.com.chinaums.appleTest"&merOrderId="000000000001"&merchantId="898310053994009"&merchantUserId="0000000"&mode="3"&notifyUrl="https://www.chinaums.com/"&productName="商品 1"&specifiedPaymentMedium="0000,2,0"&sign="ZxoEvT1begI5RjV09vFNrG2ReShEcJ2kftfUdMeT3SL1hylH1REa5S0t1CcMUUH/suhlAS4pGYA0E0Qqrxr5Nk0d3kIBxMU5BiRY5LFGIlulMVroYcot8Rpx5T2/0bVnv3vMeyrY8SEkRRg15o11wPS/CBYy8oqvu0QVMjE484o="
```

对orderString按key="value"形式以&拼接时，按照key的字母升序排序，上面的key值及value值仅供参考，具体支付参数及回调参数详细说明请参考“4.1.1.2 回调参数”。

4.1.1.2 参数说明

■ 启动参数

参数名	数据项	类型	长度	输入/选择	备注
merchantId	商户号	N		M	由银商分配
merchantUserId	商户唯一 id	An		M	商户系统用户唯一标示，字母和数字构成
merOrderId	商户订单号	Ans		M	商户订单号
appleMerchantId	Apple 商户号	Ans		M	商户在 Apple 配置的商户号，格式：merchant.com.xxx.xx

amount	交易金额	N		M	支付金额，单位为分
productName	商品名称	C		M	商品名称，在 PaymentSheet 中展示
mode	启动模式	N	1	M	插件模式值(3-Apple Pay)
transactionType	交易类型	N	2	M	01-支付，11-预授权
specifiedPaymentMedium	指定扣款介质	Ns		M	指定支持的扣款介质或默认使用扣款介质（详见 5.1.3.3 扣款介质配置说明）
notifyUrl	交易成功通知接口	Ans		M	交易成功的通知接口
sign	签名	C		M	RSA，用第三方商户 privateKey 对参数字符串拼接后进行签名（详见 3.1.8 签名机制）

■ 回调参数

参数名	数据项	类型	长度	输出/选择	备注
errCode	错误码	C		C	标识接口请求是否成功，若返回“0000”则标识成功
errInfo	错误信息	C		C	接口返回的状态描述

若根据第三方商户传入的信息进行联合登录失败，商户可在 TransactionResultBlock 回调中根据插件返回的 errCode 和 errInfo 信息进行提示，主要 errCode 和 errInfo 对应的场景如下：

errCode	errInfo
0000	支付成功/预授权成功
1000	用户取消支付/用户取消预授权
1006	网络连接超时
1100	传入参数有误(内容为具体错误信息)
8002	验签失败
8003	无此商户
2000	其他异常

注意：订单支付状态请以后台通知为准，前端返回状态仅作页面展示所用。

4.1.1.3 扣款介质配置说明

specifiedPaymentMedium 在 Apple Pay 中指允许使用的扣款介质，如：指定只能使用贷记卡进行扣款；

specifiedPaymentMedium 格式： 机构号，介质类型，使用模式；各参数间以英文半角逗号“,”号分隔

参数名	类型	长度	输入/选择	备注
机构号	An		M	Apple Pay 默认为 0000
介质类型	Ans		M	0 - 借记卡 1 - 贷记卡 2 - 借记卡&贷记卡
使用模式	An		M	Apple Pay 默认为 0

例如：

指定仅可使用贷记卡 **specifiedPaymentMedium=0000,1,0**

4.1.2 Apple Pay 商户通知

见 [5、商户通知](#)

4.1.3 Apple Pay 备注

集成全民付 IOS 插件静态库时，如遇到集成冲突问题，可参考如下解决方法：

Targets——>Build Settings——>Linking 下的 Other Linker Flags 改为-ObjC

4.2 服务端调用 Apple Pay 接口

4.2.1 总则

- 所有调用接口须以accessToken为接口调用凭据
- accessToken有效期为2小时
- 使用appId和appKey来获得accessToken

4.2.2 获取 accessToken 流程

4.2.2.1 内容要素

- appId: 产品ID, 由银联商务方提供
- appKey: 产品密钥, 由银联商务方提供
- accessToken: 服务授权令牌

4.2.2.2 报文协议

HTTPS + JSON

4.2.2.3 接口地址

测试环境: <http://116.228.21.162:29015/v1/token/access>

生产环境: <https://mop.chinaums.com/api-portal-web/v1/token/access>

4.2.2.4 报文格式

- 请求

URL 参数: 无

POST 参数:

格式: JSON				
参数名称	参数说明	参数类型	是否必须	备注
appId	产品 ID	字符串	是	
timestamp	时间戳	字符串	是	yyyyMMddHHmmss
nonce	随机数	字符串	是	
signature	签名	字符串	是	SHA1 (appId + timestamp + nonce + appKey)

■ 响应

格式: JSON				
参数名称	参数说明	参数类型	是否必须	备注
errCode	错误代码	字符串	是	0000 为成功
errInfo	错误说明	字符串	是	
accessToken	授权令牌	字符串	是	
expiresIn	失效时间	整型	是	单位为秒

4.2.2.5 建议

为了保密 appKey, 建议需要一个 accessToken 获取和刷新的中控服务器, 而其他业务逻辑所使用的 accessToken 均来自于该中控服务器。

4.2.3 业务接口

4.2.3.1 Apple Pay 交易查询

进行交易查询

■ 报文协议

HTTPS + JSON

■ 接口地址

测试环境: <http://116.228.21.162:29015/v1/applepay/transaction/query>

生产环境: <https://mop.chinaums.com/api-portal-web/v1/applepay/transaction/query>

■ 报文格式

● 请求

HTTP 报文头:

参数名称	参数说明	参数类型	是否必须	备注
Authorization	授权令牌	字符串	是	accessToken

URL 参数：无

POST 参数：

格式：JSON				
参数名称	参数类型	是否必填	说明	备注
merchantNo	String	是	商户号	
originalMerchantOrderId	String	否	原交易商户订单号	
orderId	String	否	原交易银商订单号	

● 响应

格式：JSON				
参数名称	参数类型	是否必填	说明	备注
errCode	String	是	响应码	
errInfo	String	是	响应内容	
transactionDate	String	否	交易日期：格式 MMDD	
orderId	String	否	订单号	

4.2.3.2 Apple Pay 退货

进行退货操作

■ 报文协议

HTTPS + JSON

■ 接口地址

测试环境：<http://116.228.21.162:29015/v1/applepay/transaction/refund>

生产环境：<https://mop.chinaums.com/api-portal-web/v1/applepay/transaction/refund>

■ 报文格式

●请求

HTTP 报文头:

参数名称	参数说明	参数类型	是否必须	备注
Authorization	授权令牌	字符串	是	accessToken

URL 参数: 无

POST 参数:

格式: JSON				
参数名称	参数类型	是否必填	说明	备注
transactionAmount	String	是	交易金额	
orderId	String	否	原交易银商订单号	
originalMerchantOrderId	String	否	原交易商户订单号	
newMerchantOrderId	String	是	新商户订单号	

●响应

格式: JSON				
参数名称	参数类型	是否必填	说明	备注
errCode	String	是	响应码	
errInfo	String	是	响应内容	
transactionDate	String	否	交易日期: 格式 MMDD	
orderId	String	否	订单号	

4.2.3.3 Apple Pay 预授权撤销

进行预授权撤销操作

■ 报文协议

HTTPS + JSON

■ 接口地址

测试环境: <http://116.228.21.162:29015/v1/applepay/transaction/preauthorization/voidauthorization>

生产环境：<https://mop.chinaums.com/api-portal-web/v1/applepay/transaction/preauthorization/voidauthorization>

■ 报文格式

● 请求

HTTP 报文头：

参数名称	参数说明	参数类型	是否必须	备注
Authorization	授权令牌	字符串	是	accessToken

URL 参数：无

POST 参数：

格式：JSON				
参数名称	参数类型	是否必填	说明	备注
orderId	String	否	预授权银商订单号	
merchantNo	String	是	商户号	
originalMerchantOrderId	String	否	预授权商户订单号	

● 响应

格式：JSON				
参数名称	参数类型	是否必填	说明	备注
errCode	String	是	返回码	
errInfo	String	是	返回信息	
transactionDate	String	否	交易日期：格式 MMDD	
orderId	String	是	订单号	

4.2.3.4 Apple Pay 预授权完成

进行预授权完成操作

■ 报文协议

HTTPS + JSON

■ 接口地址

测试环境：<http://116.228.21.162:29015/v1/applepay/transaction/preauthorization/complete>

生产环境：<https://mop.chinaums.com/api-portal-web/v1/applepay/transaction/preauthorization/complete>

■ 报文格式

● 请求

HTTP 报文头：

参数名称	参数说明	参数类型	是否必须	备注
Authorization	授权令牌	字符串	是	accessToken

URL 参数：无

POST 参数：

格式：JSON				
参数名称	参数类型	是否必填	说明	备注
newMerchantOrderId	String	是	新商户订单号	
merchantNo	String	是	商户号	
orderId	String	否	预授权银商订单号	
originalMerchantOrderId	String	否	预授权商户订单号	
transactionAmount	String	是	交易金额	

● 响应

格式：JSON				
参数名称	参数类型	是否必填	说明	备注
errCode	String	是	返回码	
errInfo	String	是	返回信息	
transactionDate	String	否	交易日期：格式 MMDD	
orderId	String	是	订单号	

4.2.3.5 Apple Pay 预授权完成撤销

进行预授权完成撤销操作

■ 报文协议

HTTPS + JSON

■ 接口地址

测试环境：<http://116.228.21.162:29015/v1/applepay/transaction/preauthorization/voidcompletion>

生产环境：<https://mop.chinaums.com/api-portal-web/v1/applepay/transaction/preauthorization/voidcompletion>

■ 报文格式

● 请求

HTTP 报文头：

参数名称	参数说明	参数类型	是否必须	备注
Authorization	授权令牌	字符串	是	accessToken

URL 参数：无

POST 参数：

格式：JSON				
参数名称	参数类型	是否必填	说明	备注
orderId	String	否	预授权完成银商订单号	
merchantNo	String	是	商户号	
originalMerchantOrderId	String	否	预授权完成商户订单号	

● 响应

格式：JSON				
参数名称	参数类型	是否必填	说明	备注
errCode	String	是	返回码	
errInfo	String	是	返回信息	

transactionDate	String	否	交易日期：格式 MMDD	
orderId	String	否	订单号	

5、 商户通知

5.1 接口介绍

商户在调起银商插件支付时需传入回调 URL 通知地址，银商系统在支付成功后将支付结果信息实时的推送回商户指定的回调 URL 地址上，所有参数采用 POST 方式推送。

银商后台会通过 3DES 对传送的支付信息进行加密，并对加密后的字段进行 RSA 签名，商户网关在接收消息时需要对接收到的参数进行解密和验签操作。

如果商户调起插件支付时无法提供通知地址，则表示商户不关心支付结果实时通知，则可以跳过本章节。

5.2 使用方法

银商系统会将支付结果作为参数 POST 到商户提供的 URL 服务中。

商户需要自行实现接受支付通知信息的 URL 服务，并对接收到的参数进行解密及验签操作。

5.2.1 发送及接收方式

1. 目前 URL 支持(HTTP/HTTPS)两种格式，以下参数会以 POST 方式发送到商户指定的 URL 服务中。
2. 发送字段中 params 字段采用 3DES 加密方式进行发送，密钥信息目前由银商系统为每个不同商户分配生成，并且以邮件形式通知商户。
3. 发送时会通过加密机做签名，增加签名信息，公钥信息见 [4.2.5](#)。

5.2.2 API 参数

参数名	参数说明
params	向商户传送的支付结果信息，该字段为 json 格式并经 3DES 加密，详见章节 4.2.6.1

参数名	参数说明
signature	银商私钥签名信息，采用加密机得到的签名，银商公钥见 4.2.5

params 解密后为 json 格式，示例如下：

```
{
  "account": "621444*****0038",
  "amount": "1",
  "currencyCode": "156",
  "dealDate": "2015-03-12",
  "dealStatus": "1",
  "dealTime": "10:08:51",
  "liqDate": "0202",
  "merchantId": "898000156911002",
  "merchantName": "大众点评****",
  "merchantOrderId": "0000000000000001",
  "orderId": "642015020326960682",
  "refId": "000105063751",
  "subInst": "101600",
  "agentMerchantId": "898000156911000"
}
```

JSON 字段说明：

字段名称	字段说明
------	------

字段名称	字段说明
------	------

5.2.3 回调通知 URL

商户系统如在每次跳转支付插件时传入通知地址，即支付完成后可实时接收该商户相关的支付信息。

参数	参数名称	类型	必填	描述	范例
notifyUrl	通知 URL	String(256)	否	银商服务器主动通知商户服务器里指定的页面 http 路径。	http://172.16.26.178:8080/connectDemo/NotifyOperServlet

5.2.4 3DES 密钥

3DES 密钥会通过邮件方式给各个商户分配，具体使用参见 [4.2.6.1](#)。

5.2.5 RSA 公钥（2048 位）

目前生产环境配置的银商公钥为 2048 位，信息如下：

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAs1MhXQrcDSUgx23qnXZKP
0WSY746WEQxk+WIElmW/0/mbkCLqcSh5s4d6SATRrrdjcj5fB9eGe5+IPx9An9cKN
rECCLuUFY3Wx+AV4rVPPJAAX/6AdxPxFuI+3+C6J7vNuuve8xk0gBjzXEoqzBdA6c
2/LEaa2f09X64/YEGD3C7DVP4GDV6etpSQHHfSolnHem/y13JO/NO6mJZRVm/7Wj
8uISgpnPJwK81JJO+i2MpEK7iT0tJejQM97C1FFVqHoqnfeSnBMOCIOnlXHaIrDBV
pK/wXu9oquMV0xJw/tmsfygYwqX5S6UsHAIJ6eKZOv8HvOoXSNzehgpiUSyUwIDAQ
AB
-----END PUBLIC KEY-----
```


5.2.6 商户接收设置

1. 商户需要提供通知地址 URL，并且该 URL 服务通过公网能被访问。
2. 商户对接收到的参数进行解密，解密后格式详见 [4.2.6.1](#)。

5.2.6.1 参数 params 接收处理

商户根据收到的邮件获取各自对应的 DES 密钥，可参考以下示例中的解密代码进行操作，内容如下：

```
/
**
 * 3DES 解密示例
 */
public static void main(String[] args) throws Exception {
    final String desKey = "a135ca0734fac82c924f99d78ad41493";//DES 密钥
    String encryptdStr = "31f524d8751b723c9c071cb72fbd9f876eba3c245fa55c6795f38
ba1d60666b01a8fb5dacf83ac64d8af60c2827d8f8d52fd694456ee69e53813ba60fc9375b1eb22b1e
2c04558e41337cd16ab4f2446ee81996fc6afe6c4c4b323fb3622dcef674b36e32d58172c45f2a980cd
3501c1941c422e8518719b8b74e1ea5afc85e7874c012dcfd7a4e015fa0569ac9a987c2867b0f2b4ea
e60df6c9e836d2f6cd513aadcb300528ae59d35d706c742f4b183ac2ddf8ba6207ed10b5c1529e33d
c8bda5fb772e8da85e2dd8891deffda80c42cff65b05a82a69a958e62c6efae4322554d975473f829c6
c6ccde6bc333a280e7cb1679e516d71e90a53f8e09f7ae2e7fe1dfebeab854bd394fde0e674230c3085
c62aef6200fb3863fd4ef96b07960dd36cebe4063733047731eb8633587208985195c5d83066d4fbc
3881d5a8a9e3568566e19ef457d9a5e987b69273c012f0eb8eb0a0706c9b3f5685ced4cbe0666d169c
75f6e742fbe23062dde3434f8d70a7d36d8dd238782c5d4136bb2ceba3eb136ec61dd5e852ac20b21f
6cef590a4502fd51aadf3a03eb136ec61dd5e8516764b307e122aa4a80452eba63fbd9b3a09a7be15b
6d24590835e0d5989f1cf";//加密字符串

    byte[] desByte = desEdeEcbPkcs5Decrypt(ByteArrayUtil.hexStringToByteArray(enc
ryptdStr), ByteArrayUtil.hexStringToByteArray(desKey));
    System.out.println("3DES 解密: " + new String( desByte ,"UTF-8"));
}

public static byte[] desEdeEcbPkcs5Decrypt(byte[] data, byte[] keyData)
    throws Exception {
    if (keyData.length < 24)
        keyData = make3DesKey(keyData);
    Key localKey = makeDesKey(keyData);
    Cipher localCipher = Cipher.getInstance("DESede/ECB/PKCS5Padding");
    localCipher.init(Cipher.DECRYPT_MODE, localKey);
    return localCipher.doFinal(data);
}
```

控制台输出内容如下：

```
3DES 解密: {"account":"6214 44** **** 0103","acqBankCode":"48023310","amount":"1",
"authNo":"","batchNo":"000001","billNo":"000000000000001","currencyCode":"156","dealDate":
null,"dealStatus":"","dealTime":"","expireDate":"1912","id":"40288b134af1ada1014af1ae55e900
00","issNo":"90310000","merchantId":"898000156911002","merchantName":"绫致服装****
*","operator":"000001","orderId":"","refId":"000105012799","termId":"00019130","voucherNo":
"316044"}
```

5.2.6.2 参数 signature 接收处理

根据 4.2.5 提供的 RSA 银商公钥，将接收到的 signature 与 params 值进行验签，如果一致，则认为正确，不一致则舍弃。可参考以下示例中的验签代码进行操作，内容如下：

```
/**
 * RSA 验签示例
 */
public static void main(String[] args) throws Exception {
    final String publicKey = "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA
vx8WfUE6n3dVCtJbLgEFgJaV70WZXq2ouWmHCwNXLjSr4UZ9IibGHX7KTSj/PozyO
hdTv0sncIrar1tSIQAWLq5qpP8TVPjMwu9uV/WNg16tnqY/eYowSFex+MlrqAAj1E8Rb3dnRtN
rqCPzZr0bZIsxFAdshp+7OgMVqHQ9fIoR3EkYDCJFwx9Ng8u5k4qbuNLAshcuZRgoMtPDC
B14hvLP+euI9W+JUIKv+BTG2v9uMh42Zp3NPas1vdTKtDDX7GAsiu6fex5KT7zYxXNOrg8N
YGzAILyFPO8bDosFcd+l/SD+M8WcTmsOA98YjWrOcepBwKrWoWoe7/zQDTv0QIDAQAB";
    //银商公钥
    String params = "2C467F9EA92253A55D8E02FC872A61CCAF3A8D958517C6864
11AA4B16A4D42F867D0746005E3EA4FFCF82A934D4670AC28106B7A54E62B43F1168A06
65C2B00935CA52DB3F42F3F0D6AD9CCFAA27865BC814DA7AEAC2644C83CAC418611A
C34F94311CDA1BCF8EACA4FF03ADC8D7722E7D2387793E54ACEE6A81256009B0170F98
92955076836767AA1031ED84BEB4AB4EFDA50AFC5D19BFB59E9F656052C21324828DED
DFDE7866CAED6D532340DCEBA8E708531A5B97F5AB6B7B8D3875F17FD4288C9223A39
19AC3A29B1D8579B7EFD3506A2D83FAF8B004F8EB412C4C2DC936EFAC480FF80460B3F
F7B814AD8C39D85F24DF84BB97FA9A6DCA1DAF246706D4AD700F8C77FD21400F35E40
65917A2C65C360136321CAEA3980B85A43DC82F7D05C94A139CD6B2F37DE482E9699710
63BD6BF0D839E5664F720FA2CD101A9A655F7C405065972E4F6E1D42C9B43707C053FE2
BE2BB4B84576AACAD6C6CDA04111B71FFE8A60E33807C5E74FC10382CEA81346117387
```

```
2CAD52B33497FAF83F257E74A742A90AEA46906CAEAE35EB2C81E3CE883BE5F8B76D7
941EBAAE67B64E686DE8A32C0F600B862CECFC311075441";

    String signature = "8BFD8565379ACEC4DA11A0AF423E1870C153E7D959DB8E
0658FB6AF0CA18E48F60DEBFC7993BB1E907E263CAC4064A4B0B9EC71ED6910D136AC
4CCF8111AE2CA3621963EB628D650CAB0A791AD73E56B66414C702A0E4D8711BDB1974
6821DE11B666F4D4801ABB02D118409658B7AB2BE33A3FB3E660BD20B2C6BE35201025F
B1E149A58A008B20642263149E736C0798E18DD1B2B6F4ACD0858842BFDCB25C0956D36
C6C5B1A3F1A1B91B1D588FB88B452038FD11C6088F1E00CBEA6E4E8A7887DCD42A038
C48C0EA91C3244B6D7D7704FCBCE14E7C0C2F85F3EA617D0ECA36712DC95C789B6BE0
086F6DC514A97D355488255E8808D10FE9AD4610A619EC2";

    System.out.println("验签结果: "+verify(params.getBytes(), publicKey, signature));
}

public static boolean verify(byte[] data, String publicKey, String sign)
    throws Exception {
    byte[] keyBytes = Base64Utils.decode(publicKey);
    X509EncodedKeySpec keySpec = new X509EncodedKeySpec(keyBytes);
    KeyFactory keyFactory = KeyFactory.getInstance("RSA");
    PublicKey publicK = keyFactory.generatePublic(keySpec);
    Signature signature = Signature.getInstance("SHA1withRSA");
    signature.initVerify(publicK);
    signature.update(data);
    return signature.verify(ByteArrayUtil.hexStringToByteArray(sign));
}
```

提示：加密算法为 RSA，签名算法为 SHA1withRSA，其中 signature 为 16 进制编码。

控制台输出内容如下：

```
验签结果: true
```

5.2.7 商户返回

商户响应银商 json 格式如下：

参数名	参数说明	是否必传
merchantRecMsgProcessState	商户处理结果状态(0 失败， 1 成功)	是
merchantMsgProcessTime	商户接收到的时间	否

示例：

```
{
  "merchantRecMsgProcessState":1
  "merchantMsgProcessTime":"2015-03-20 00:05:57"
}
```

6、 单笔订单查询

6.1 规则介绍

通过该网关接口，商户可以根据银商系统订单号或商户唯一订单号查询交易详细信息（包括交易结果，支付信息，商户信息等）。

商户接入时，调用接口须遵循以下规则：

1. 传输方式 为保证交易安全性，采用 HTTPS 传输
2. 提交方式 采用 HTTP POST 方法提交
3. 数据格式 返回参数均为 JSON 格式
4. 字符编码 支持 UTF-8 字符编码
5. 签名算法 商户生成签名字符串，现支持的签名算法类型为 RSA
6. 签名要求 请求数据需要校验签名

6.2 界面原型

说明：请求时必须提供商户编号，商户唯一订单号与银商订单号二选一去查询，代理模式下需提供代理商户号。

商户号: 898330572300006

银商订单号: 392015090728416656

商户唯一订单号: 1442803121615

代理商户号: 898330572300000

单笔订单查询

提示：该界面仅供参考，请依据商户自身的业务逻辑和平台管理情况，把接口嵌入到商户系统中。

6.3 请求、响应参数说明

请求参数：商户在与银商系统进行数据交互时，提供给银商系统的请求数据，以便银商系统根据这些数据进一步处理。

参数	参数名称	类型	必填	描述	范例
orderId	银商订单号	String(18)	否	银商订单号与商户订单号 选其一	3920150907284 16656
merOrderId	商户订单号	String	否	银商订单号与商户订单号 选其一	144280312161 5
billsMID	商户号	String(15)	是		8983305723000 06
agentMID	代理商户号	String(15)	否	代理模式下提供	8983305723000 00
sign	签名	String(172)	是	商户请求参数的签名串	详见实例 5.5

响应参数：银商系统对商户提供的请求数据进行处理后，返回给商户结果数据，以便商户根据这些数据进一步处理。

参数	参数名称	类型	描述	范例
orderId	银商订单号	String		392015090728416656
merOrderId	商户订单号	String		1442803121615
billsMID	商户号	String		898330572300006
billsMercName	商户名称	String		大众点评
payDate	支付时间	String	格式: yyyy/MM/dd HH:mm:ss	2015/03/13 10:50:28
orderState	订单状态	String	详见附录 7.1	1
payState	支付状态	String	详见附录 7.2	1
pAccount	卡号	String	前 6 后四	621738*****3592
amount	交易金额	String	单位: 分	8500
refId	检索参考号	String		000067838248
currencyCode	货币代码	String	详见附录 7.3 (暂只支持人民币)	156
memo		String	保留	
errCode	错误码	String	成功: 0000 失败: 非 0000	0002
errInfo	错误信息	String	成功: SUCCESS 失败: 相应错误信息	验签失败

6.4 POST 消息体

开发者网关可以向银商网关发送 POST 请求，以下为开发者调用订单查询时发送的 POST 消息，服务地址：<https://mpos.quanminfu.com/qs/queryOrder/>

```
REQUEST URL: https://mpos.quanminfu.com/qs/queryOrder/
REQUEST METHOD: POST
Form Data:
    billsMID=2014072300007148
    orderId=392015090728416656
    merOrderId=1442803121615
    sign=e9zEAe4TTQ4LPLQvETPoLGXTiURcxiAKfMVQ6Hrrsx2hmyIEGvSfAQzbLxHrhy
    Z48wOJXTsD4FPnt+YGdK57+fP1BCbf9rIVycfjhYCqlFhbTu9pFnZgT55W+xbAFb9y7vL0My
    AxwXUXvZtQVqEwW7pURtKilbcBTEW7TAxzgro=
```

提示：Form 表单数据请参照 [5.3](#) 请求。

6.5 消息签名

在 POST 消息体中的 sign 参数是由商户使用自己的私钥对参数进行签名产生的，签名的方法：

1. 将所有 POST 表单参数（sign 除外）进行字典排序，中间以”&”拼接组成待签名字符串，如：

```
billsMID=898330572300006&merOrderId=1442803121615&orderId=642015081428406867
```

2. 使用 OpenSSL 的加密算法对以上字符串进行签名并进行 Base64 编码，以下为 Java 代码的签名方法代码片段，供参考：

```
/**
 * 用私钥对信息生成数字签名
 * @param data 待签名数据
 * @param privateKey 私钥
 * @return
 * @throws Exception
 */
public static String sign(byte[] data, String privateKey) throws Exception {
    byte[] keyBytes = Base64Utils.decode(privateKey);
```

```
PKCS8EncodedKeySpec pkcs8KeySpec = new PKCS8EncodedKeySpec(keyBytes);
//加密算法: RSA
KeyFactory keyFactory = KeyFactory.getInstance("RSA");
PrivateKey privateK = keyFactory.generatePrivate(pkcs8KeySpec);
//用 SHA 算法签名, 用 RSA 算法加密
Signature signature = Signature.getInstance("SHA1withRSA");
signature.initSign(privateK);
signature.update(data);
//对签名进行 Base64 编码, 变为可读的字符串
return new String(Base64Utils.encode(signature.sign()));
}
```

提示: 如何获取私钥, 请参考 [6.2](#) 内容

7、 RSA 公私钥

银商系统采用了 RSA 安全签名机制, 开发者可以通过提供给银商的公钥验证消息来源, 同时可使用自己的私钥对信息进行加密。RSA 算法及数字签名机制是银商系统与开发者网关安全通信的基础, 若开发者不熟悉 RSA 及数字签名, 请先查阅相关资料。

7.1 OpenSSL 工具安装

1. Linux 用户 (以 Ubuntu 为例) `sudo apt-get install openssl`
2. Windows 用户 开发者可以在 OpenSSL 官方网站下载 Windows 的 OpenSSL 安装包进行安

7.2 RSA 私钥及公钥生成

1. Linux 用户 (以 Ubuntu 为例) `$ openssl` 进入 OpenSSL 程序
`OpenSSL> genrsa -out rsa_private_key.pem 1024` //生成私钥
`OpenSSL> pkcs8 -topk8 -inform PEM -in rsa_private_key.pem -outform PEM -nocrypt` //Java 开发者需要将私钥转换成 PKCS8 格式
`OpenSSL> rsa -in rsa_private_key.pem -pubout -out rsa_public_key.pem`
//生成公钥
`OpenSSL> exit`
2. Windows 用户在 cmd 窗口中进行以下操作:
`C:\Users\futao>cd C:\OpenSSL-Win32\bin` //进入 OpenSSL 安装目录
`C:\OpenSSL-Win32\bin>openssl.exe` //进入 OpenSSL 程序
`OpenSSL> genrsa -out rsa_private_key.pem 1024` //生成私钥

OpenSSL> pkcs8 -topk8 -inform PEM -in rsa_private_key.pem -outform PEM -nocrypt //Java 开发者需要将私钥转换成 PKCS8 格式

OpenSSL> rsa -in rsa_private_key.pem -pubout -out rsa_public_key.pem //生成公钥

OpenSSL> exit

注意：对于使用 Java 的开发者，将 pkcs8 在 console 中输出的私钥去除头尾、换行和空格，作为开发者私钥，对于.NET 和 PHP 的开发者来说，无需进行 pkcs8 命令行操作。

经过以上步骤，开发者可以在当前文件夹中（Windows 用户在 C:\OpenSSL-Win32\bin）看到 **rsa_private_key.pem** 和 **rsa_public_key.pem** 两个文件，前者为私钥，后者为公钥。开发者将私钥保留，将公钥提交给银商系统，用于信息加密及解密。以下为使用 OpenSSL 生成的私钥文件和公钥文件示例。

1. 标准的私钥文件示例（PHP、.NET 使用）

```
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQC+L0rfjLl3neHleNMOsYTW8r0QXZ5RVb2p/vvY3fJNNugvJ71o4
+fdBz+LN4mDxTz4MTOhi5e2yeAqx+v3nKpNmPzC5LmDjhHZURhwbqFtIpZD51mOfn
o2c3MDwlrsvi6mTypbNu4uaQzw/TopwufSLWF7k6p2pLoVmmqJzQiD0QIDAQABAoG
AakB1risquv9D4zX7hCv9MTFwGyKSfpJOYhkIjwKAik7wrNeeqFEbisqv35FpjGq3
Q1oJpGkem4pxaLVEyZOHONefZ9MGVChT/MNH5b0FJYWl392RZy8KCdq376Vt4gKVl
ABvaV1DkapL+nLh7LMo/bENudARsxD55IGObMU19lkCQQDwHmzWPMHfc3kdY6AqiL
rOss+MVIAhQqZOHHDe0aW2gZtwiWeYK1wB/fRxJ5esk1sScOWgzvCN/oGJLhU3kip
HAkEAysNoSdG2oWADx1It4W9kUiiiqNgimHGMHPwp4JMxupHMTm7D9XtGUIiDijZx
unHv3kvktNfwj3Yji0661zHVJwJBAM8Tdf077F4NsVc9AXVs8N0sq3xzqwQD/HPFz
fq6hdR8tVY5yRMB4X7+SX4EDPORKKsgnYcur5lk8MUi7r072iUCQQC8xQvUne+fcd
pRyrR4StJlQvucogwjTKMbYRBDygXkIlTJOIorgudFlrKP/HwJDoY4uQNl8gQJb/1
LdrKwIe7FAkBl0TNtfodGrDXBHwBgtN/t3pyi+sz7OpJdUklKE7zMSBuLd1E3O4JM
zvWP9wEE7JDb+brjgK4/cxxUHUTkk592
-----END RSA PRIVATE KEY-----
```

2. PKCS8 处理后的私钥文件示例（Java 使用）

```
-----BEGIN PRIVATE KEY-----
MIICeAIBADANBgkqhkiG9w0BAQEFAASCAmIwggJcAgEAAoGBAN0yqPkLXlnhM+2H/
57aHsYHaHXazr9pFQun907TMvmbR04wHChVsKVgGUF1hC0FN9hfeYT5v2SXg1WJSg
2tSgk7F29SpsF0I36oSLCIszxd7C107c22mxEVuCuYpJdq6XweAZzv4Is661jX
P4PdrCTHRdVTU5zR9xUByiLSVAgMBAAECgYEAhznORRonHyIm9oKaygEsqQGkYdXB
bnsOS6busLi6xA+ioveUdbAVIrTCG9t854z2HAGaISoRUKyztJoOtJfIlwJaQU+XL
+U3JIh4jmNx/k5UzJijfvfpT7Cv3ueMtqyAGBJrkLvXjis7O5ylaCGuB0Qz711bWG
kRrVoosPM3N6ECQQD8hVQUgnHEVHZYtvFqfcoq2g/onPbSqyjdrRu35a7PvgDAZx6
```

```
9Mr/XggGNTgT3jJn7+2XmiGkHM1fd1Ob/3uAdAkEA4D7aE3ZgXG/PQqlm3VbE/+4M
vNl8xhjQOkByBOY2ZFfWKhlRziLEPSSAh16xEJ79WgY9iti+guLRAMravGrs2QJBA
OmKWYeaWKNNxiIoF7/4VDgrcpkcSf3uRB44UjFSn8kLnWBUPo6WV+x1FQBdjQRviZ
4NFGIP+KqrJnFHZNjgJhVUCQFzCAukMDV4PLfeQJSmna8PFz2UKva8fvTutTryyEYu
+PauaX5laDjyQbc4RIEMU0Q29CRX3BA8WDYg7YPGRdTkcQQCG+pjU2FB17ZLuKRlK
EdtXNV6zQFTmFc1TKhlsDTtCkWs/xwkoCfZKstuV3Uc5J4BNJDkQOGm38pDRPcUDU
h2/
-----END PRIVATE KEY-----
```

3. 公钥文件示例

```
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDQWiDVZ7XYxa4CQsZoB3n7bfxLD
keGKjyQPt2FUtM4TWX9OYrd523iw6UUqnQ+Evfw88JgRnhyXadp+vnPKP7unormYQ
AfsM/CxzrfMoVdtwSiGtIJB4pfyRXjA+KL8nIa2hdQy5nLfgPVGZN4WidfUY/Qpkd
dCVXnZ4bAUaQjXQIDAQAB
-----END PUBLIC KEY-----
```

8、 附录

8.1 订单状态表

代码	描述
0	新订单
1	订单完成
2	订单失败
3	订单处理中

8.2 支付状态表

代码	描述
0	未支付
1	支付成功
2	支付失败
3	支付中
14	撤销中
15	撤销成功
16	撤销失败

8.3 货币代码表

代码	描述
156	人民币

8.4 Android 客户端 APK 签名表

308202bf308201a7a003020102020420741661300d06092a864886f70d01010b050030
0f310d300b060355040313045169616e3020170d3134313130363035353533305a180f
32313134313031333035353533305a300f310d300b060355040313045169616e308201
22300d06092a864886f70d01010105000382010f003082010a0282010100b39c49501c
25231967f4ef41cbbbed3659a3a7499a103826f62e943b8f334978dd1fb59a22b8ae801

5c84f9d56efdd0995bc7be663529f0403e3438e0074d46ccb06cba640156a2a947a26e
70a9faadda63c24d267021f8ad0816b8506afd71843183b98fdb8f7e875efecbb9c0d
1bb50ac85f80e010bbcf5a52956880d0b433bf83977c7ebeaf307a14919a2b209557787
2ce24fa173d59aa362248d96617d5aec6491098cafc2cb65eaafdbf8806371ba855f0f
f1506e0a089b07cb4e44c697d089f7c66ee9a77f3bc0fcf1069348b542ae7881cb4433
65c16e8de77541f409135f1adac4357471aed95b8858db0e2b767ae43526c48deba73f
4f181c4924850203010001a321301f301d0603551d0e04160414cade6cc4a4ed6647e3
ddc1d5d03edb33c09460ad300d06092a864886f70d01010b050003820101008ee28c91
ed39be71db6d50b82558b1ccc3b00b4e8ec3147bbf3c5dcab5bda1bd7a87ffad636389
87e4c96c764309e4a25eb0fd23ecf27944cbc3cd80e426847698fbe0db710149e967fa
22f9abbefa5e38b710bd32510019587a7da7884cf3f43c1b1b5cada1879a34c2818921
221ac3e8749dd0b445504bf83fd91fc3314c8b282060d6163ac6c0b9a46c0c891badd3
318c5b360bb85c9439b4c47a6f362bf367b632c0de939331aab38ae7f0e3dd313ef375
a403dd495aa61e043fe8f3fb9ca1be21f9db3c4cc557fcc6b7f22142096667c62e052a
5a2e6f8221704f8dab1615346e380c9f24fc210b960b7ec064d27dc62d721c6d3a7ff7
2509f6c3a68103