华通移动 App 银联支付方案

目录

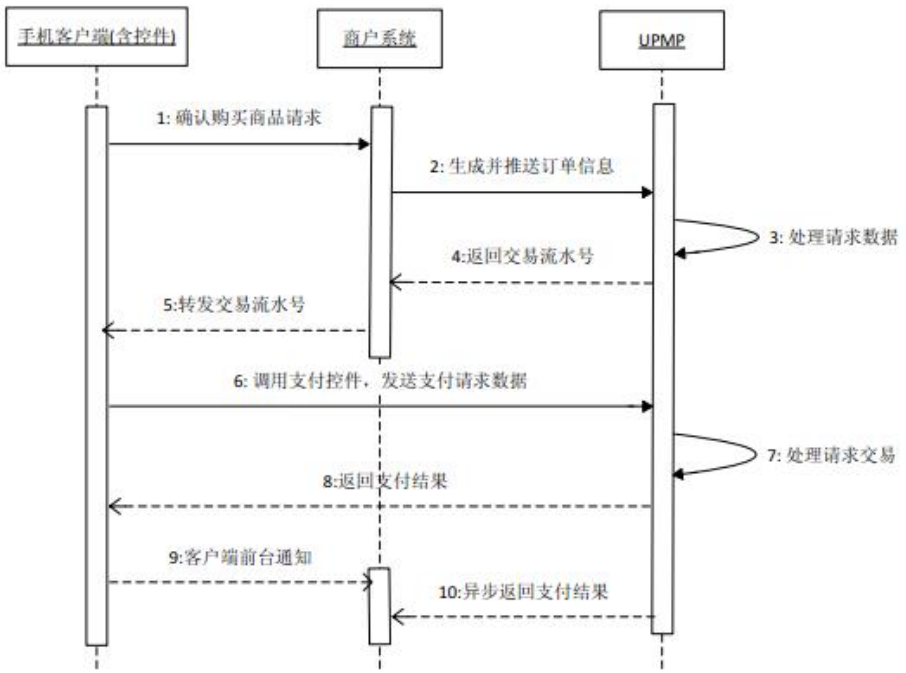
前言

本方案适用于开发银联支付交易服务端 Api，不适用于桌面浏览器或者手机浏览器网页版银联在线支付。
在前期，会进行商户银联支付接入申请，在拿到商户专有测试 id 和 key 以后，可开始相关 api 的开发和调试。
下文的“银联支付”均指 native app 与银联相关支付服务

开发流程

1、用公用的测试账号（商户号、合作密钥）参考 Demo 跑通支付流程。
2、待商户的专用测试账号下发后，用商户专用的测试账号（商户号、合作密钥）参考测试报告模板，出测试报告发给银联技术，后面放出测试模板示例。
3、我们拿测试报告去申请商用账号，商用账号下发后，替换、测试、上线。

支付流程



以上为支付流程图，详细参考《银联支付接入指南》

说明一点，因为银联商务的商户号和密钥均是敏感信息，故**不允许由移动端直接向银联接口发起支付请求**。商户号和密钥只能放在客户服务端，**相关的交易请求由服务端向银联**

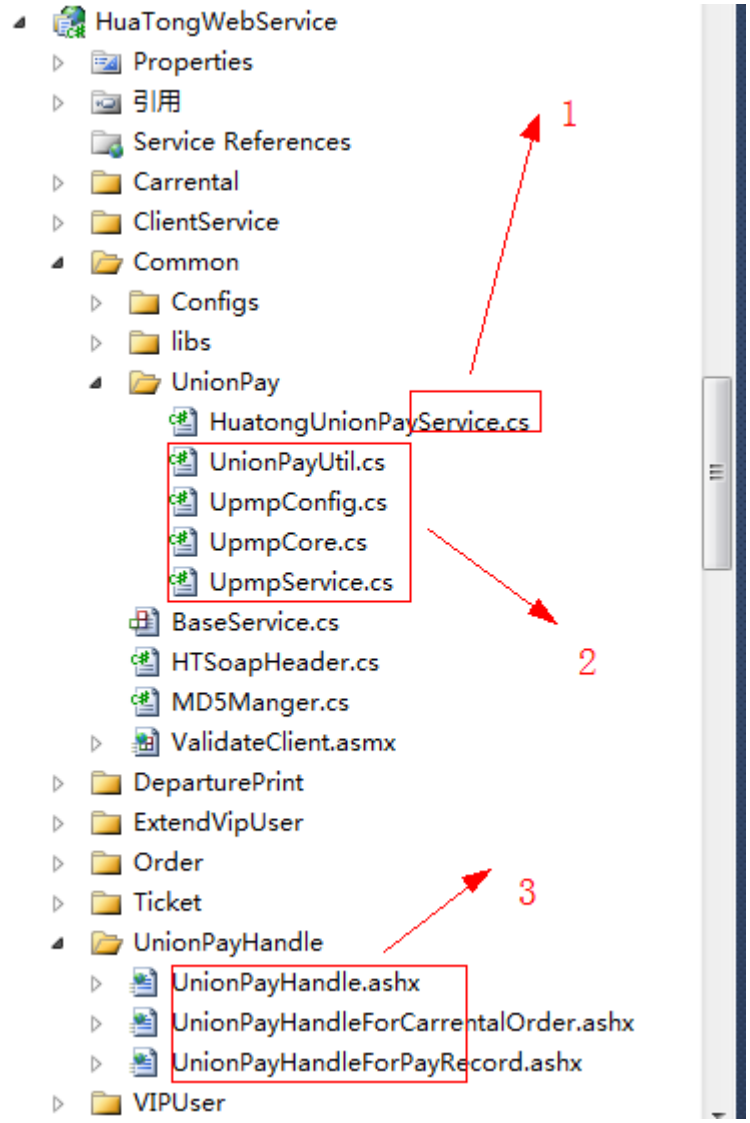
接口发起。

支付服务主要由以下 2 个流程，订单推送和订单查询。具体如下

1. 客户端下单，向服务端相关 api 请求，由服务端向银联交易接口发起请求，此动作叫订单推送。银联交易接口响应成功以后，会返回响应报文，内容包含但不限于响应码 code 和交易流水号 tn。具体报文内容后述。
2. 服务端在收到银联的响应报文以后，需要做判断，如果银联交易接口响应成功（code=00，具体参考银联交易帮助文档），将交易流水号 tn 返回给客户端。客户端凭借此流水号，启动移动端银联支付组件，输入银行卡号密码等信息，完成支付。
3. 移动端支付结束后，如果设备和网络正常，支付插件会收到银联的交易提示。此时，客户端需要再次向服务端交易查询 api 接口发请求，由服务端再次向银联的交易查询接口发起查询，在收到交易查询报文后，做相关判断和处理(比如改订单的支付标志等操作)。

注意，为防止客户端环境突变，网络环境还有银联服务端消息推送的延时和失效。**发起交易查询这一步是必须的。银联服务方也要求交易查询这个动作必须有。**

文件组织和说明

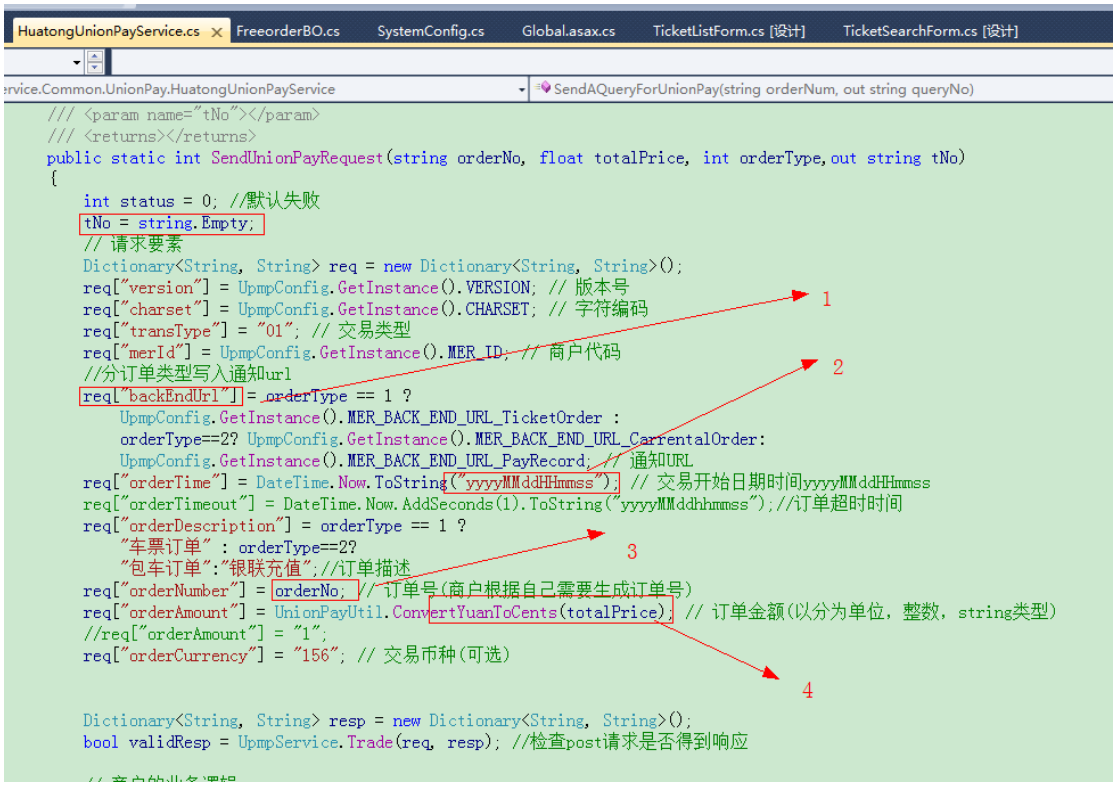


1. 包含两个服务，一个银联支付交易请求方法，一个银联交易查询方法，开发方编写，业务相关
2. 2 处文件和代码由银联提供
3. 在交易成功后，用来处理银联接口推送的交易信息的文件，开发方编写。

Config 配置文件需要我们改的有 1，2，3 处，显然，这些敏感信息以 class 的形式被编写，而不是 xml。

1 处，商户 id 和密钥
2 处，接收交易信息的 url，注意一定要发布到外网，内网无法接收到银联推送的消息。
3 处，银联交易请求接口和查询接口，注意测试接口和生产环境的接口不同，测试接口经常不稳定，出现问题请联系银联服务方对接人。

订单推送



上图为交易请求报文部分，注意，凡是我注释掉的代码的一律忽略

说明几个要注意的地方

1 处，交易成功后接收银联推送消息的 url，外网地址，不多说。

2 处，本处时间格式一定要跟示例保持一致，否则报报文错误。

3 处，订单号，有长度限制，对特殊字符敏感。如果选用 guid 作为订单号，那么请使用 toString(“N”)格式化 GUID，否则带有”-”字符会导致报文错误。

4 处，订单金额，单位是分，类型为 string，不支持浮点数。切换到生产环境时，请仔细检查本处，防止大笔金额付款。

非必须的报文字段别加进去了，否则报错后找错误困难。

```
/// <summary>
/// 交易接口处理
/// </summary>
/// <param name="req">请求要素</param>
/// <param name="resp">应答要素</param>
/// <returns>是否成功</returns>
public static bool Trade(Dictionary<String, String> req, Dictionary<String, String> resp)
{
    String nvp = BuildReq(req, resp);
    LogHelper.WriteLog("交易请求报文:"+nvp);
    String respString = UpmpCore.Post(UpmpConfig.GetInstance().TRADE_URL, nvp);
    LogHelper.WriteLog("交易响应报文:"+respString);
    return VerifyResponse(respString, resp);

}
```

注意把报文打入日志，便于开发阶段对比分析，上线后去掉即可。

看下交易日志中写入的交易请求报文

```
日志时间：2014-04-12 00:56:22,573 [178]
日志级别：INFO
异 常 类：loginfo [(null)]
交易请求报文:version=1.0.0&charset=UTF-8&transType=01&merId=802310048992610&backEndUrl=http%3a%2f
%2f121.199.13.50%3a8088%2fUnionPayHandle%2fUnionPayHandle.ashx&orderTime=20140412005622&orderDescription=%e8%bd%a6%e7%a5%a8%e8%ae
%a2%e5%8d
%95&orderNumber=78eca10390118703&orderAmount=15000&orderCurrency=156&signature=a4c362982b05dca585bba55547bbd15c&signMethod=MD5

日志时间：2014-04-12 00:56:22,902 [178]
日志级别：INFO
异 常 类：loginfo [(null)]
交易响应报文:respCode=00&tn=20140412005622006760 2&signMethod=MD5&transType=01&charset=UTF-
8&signature=d70541a5f7f124d6aecad3ba7fe81a3d&version=1.0.0
```

以上是交易请求的请求报文和 对应的响应报文。注意 respCode，00 是正常响应，其他异常。注意交易响应报文返回了流水号 tn，tn 返回给客户端，并且这个交易流水号需要保存到数据库，方便出现异常后找银联查询。

交易查询

本步骤是必须的，保证可靠性。

```
/// <returns></returns>
public static int SendAQueryForUnionPay(string orderNum, out string queryNo)
{
    int status = 0; //默认失败
    queryNo = string.Empty;
    // 请求要素
    Dictionary<String, String> req = new Dictionary<String, String>();
    req["version"] = UpmpConfig.GetInstance().VERSION; // 版本号
    req["charset"] = UpmpConfig.GetInstance().CHARSET; // 字符编码
    req["transType"] = "01"; // 交易类型
    req["merId"] = UpmpConfig.GetInstance().MER_ID; // 商户代码
    req["orderTime"] = DateTime.Now.ToString("yyyyMMdd"); // 交易开始日期时间yyyyMMddHHmmss或yyyyMMdd
    req["orderNumber"] = orderNum; // 订单号

    Dictionary<String, String> resp = new Dictionary<String, String>();
    bool validResp = UpmpService.Query(req, resp);

    // 商户的业务逻辑
    if (validResp) //验证签名
    {
        string qn = string.Empty; //查询交易流水号
        string responseCode = string.Empty; //查询交易，银联应答码
        string responseMsg = string.Empty; //银联的应答消息
        string transStatus = string.Empty; //交易状态
        resp.TryGetValue("qn", out qn);
        resp.TryGetValue(UpmpConfig.GetInstance().RESPONSE_CODE, out responseCode);
        resp.TryGetValue(UpmpConfig.GetInstance().RESPONSE_MSG, out responseMsg);
        resp.TryGetValue(UpmpConfig.GetInstance().TRANS_STATUS, out transStatus);
        // 服务器应答签名验证成功
         if (responseCode.Equals("00") && transStatus.Equals("00") && (!string.IsNullOrEmpty(qn)))
        {
            //应答成功，交易成功,交易查询码非空
            status = 1;
```

和交易请求类似，交易查询的处理，报文字段明显要少，关键的报文字段是 orderNum，要求同上述。也要注意时间格式。

把交易查询的请求报文和响应报文也写入日志。
看下交易交易查询的请求报文和响应报文

```
日志时间：2014-04-14 09:59:07,506 [22]
日志级别：INFO
异 常 类：loginfo [(null)]
交易查询请求报文：version=1.0.0&charset=UTF-
8&transType=01&merId=802310048992610&orderTime=20140414&orderNumber=bb73814546e34e639935687110c89c38&signature=1702e9b446bc6c7d2ed
3179310ba7013&signMethod=MD5

日志时间：2014-04-14 09:59:07,896 [22]
日志级别：INFO
异 常 类：loginfo [(null)]
交易查询请求响应报文：qn=20140414095703094723l&respCode=00&exchangeRate=0&charset=UTF-
8&merId=802310048992610&settleDate=0414&orderTime=20140414095703&transStatus=00&sysReserved=%7BacqCode%3D88022900%26traceNumber
%3D094723%26traceTime
%3D0414095703%7D&version=1.0.0&settleCurrency=156&signMethod=MD5&transType=01&settleAmount=10&orderNumber=bb73814546e34e6399356871
10c89c38&signature=37c58c3332f009f85486f27618162f70
```

注意响应码 respCode,正常未 00，其他异常。
交易查询返回的是查询流水号 qn，有必要的话，需要保存到数据库。

为什么交易查询是必须的呢。一般说来，我们在报文中指定了接收支付消息的 url，银联接口在支付成功后会向我们指定的 url 推送交易信息。但是，这种推送是异步的，而且也不保证消息 100%可靠推送。所以，在移动端收到支付成功以后，需要主动发起一次交易查询，确认交易是否成功，如果成功，则需要订单的一些标志，最明显的是处理支付标志。

而在服务端接收到银联推送的消息后，也会做类似的处理，相当于有两个动作来保证支付完毕后，订单的支付状态能及时可靠的被修改。

推送消息处理

在交易成功后，银联会向交易请求报文中指定的 url 推送交易信息，此推送是异步推送。在接口繁忙时，会稍有延迟。

看代码，分两张图

```
    /// UnionPayHandle 的摘要说明
    /// add by ys on 2013-11-15
    /// </summary>
    public class UnionPayHandle : IHttpHandler
    {
        private OrderFacade orderFacade { get; set; }

        public UnionPayHandle()
        {
            orderFacade = (OrderFacade)SpringFactory.GetObject("orderFacade");
        }
        public void ProcessRequest(HttpContext context)
        {
            #region
            if (para.Count > 0)
            {
                StringBuilder builder = new StringBuilder();
                foreach (KeyValuePair<string, string> keyValuePair in para)
                {
                    builder.AppendLine(keyValuePair.Key + ":" + keyValuePair.Value);
                }
                LogHelper.WriteLog("Server端接收消息时间:"+DateTime.Now.ToString("yyyy-MM-dd hh:mm:ss")+" 消息内
容:"+builder.ToString());//写入log

                if (UpmpService.VerifySignature(para))
                {   // 服务器签名验证成功
                    //请在这里加上商户的业务逻辑程序代码
                    //获取通知返回参数，可参考接口文档中通知参数列表(以下仅供参考)
                    String transStatus = para["transStatus"];// 交易状态
                    string orderNumber = string.Empty;
                    string qn = string.Empty;
                    string rspMsg = string.Empty;
```

```
                    string rspMsg = string.Empty;
                    string unionPayNum = string.Empty;
                    if ("" != transStatus && "00" == transStatus)
                    {
                        para.TryGetValue("orderNumber",out orderNumber);
                        para.TryGetValue("qn", out qn);
                        // 交易处理成功,更改订单支付状态
                        if((!string.Empty.Equals(orderNumber)))
                        {
                            #region
                            bool updateOrderStatus=orderFacade.GetTicketLineOrderSer().UpdateOrderStatus(orderNumber,qn);
                            if(!updateOrderStatus)
                            {
                                LogHelper.WriteLog(string.Format("车票订单{0}银行扣款成功，更改订单支付标志失败,交易查询流水号
{1}",orderNumber,qn));
                            }
                            else
                            {
                                LogHelper.WriteLog(string.Format("车票订单{0}银行扣款成功，更改订单支付标志成功,交易查询流水号
{1}", orderNumber, qn));
                            }
                        }
                    }
                    else
                    {
                        para.TryGetValue("respMsg",out rspMsg);
                        LogHelper.WriteLog(string.Format("车票订单{0}银行扣款失败,{1}", orderNumber, rspMsg));
                    }
                    context.Response.StatusCode=200;//写回200响应码给银行接口
                }
                else// 服务器签名验证失败
                {
```

在华通的 api 中我采用的是一般处理程序 ashx 来 处理推送消息。原因无疑是 ashx 封装的更薄，比页面文件 aspx 要好。Aspx 也能达到要求，视个人喜好而定。

在本处理文件中，接收到银联的推送消息后进行判断处理，如果消息显示交易成功，那么再本处要对订单做必要的处理，比如更改订单的支付标志或者生效标志等。

前面客户端主动发起交易查询，与本处消息处理文件目的相同。再此不在赘述。

对本文档有疑问的欢迎私下找我。