通联商户对账接口规范

V2.2

通联支付网络服务股份有限公司

2016-9-22

版本控制信息

文档名称：	通联互联网支付网关商户对账文件下载技术规范		
当前版本号：	2.2	文档编号：	
撰写者：		日期：	
复核者：		日期：	
审批者：		日期：	

版本历史：

版本	日期	拟稿和修改	说明
1.0	2011-3-18	苗洪梅	初稿
1.1	2014-10-17	徐仁彬	修改
1.2	2016-01-15	孙淑静	修改
2.1	2016-08-30	孙淑静	修改（基于二代系统）
2.2	2016-09-22	孙淑静	修改 settleDate 参数

1. 目的

文档为使用通联互联网支付网关的商户开发者编写
本文档为通联互联网支付网关的开发者和相关人员提供有效的指引和帮助
本文档的目标读者为技术人员

2. 对账文件下载

商户对账文件下载目前有两种方式：

- 登录通联商户服务平台下载对账文件（https://merchant.allinpay.com/sso/login）。
- HTTP 方式下载对账文件地址（https://merchant.allinpay.com/ms/onlinebill/download）

3. 对账文件格式

第一行为汇总信息：

结算日期|批次号|交易笔数|成功笔数|交易金额|退款笔数|退款金额|手续费|清算金额

第二行开始为交易明细：

交易类型|结算日期|商户号|交易时间|商户订单号|通联流水号|交易金额|手续费|清算金额|币种|商户原始订单金额(分)

备注：币种采用国标形式（3 位数字，比如人民币 156 美元 840 等）

文件最后为签名信息，与交易明细间有一空行。

请参考对账文件范例如下：

汇总和交易明细明文：

```
20141016|SN20141016111036182232|31|31|139303.06|0|0.00|696.50|139303.06
ZF|2014-10-16|20140901|2014-10-15 09:33:24|20141015093102005|201410150933039977|134.90|0.67|
134.90|156|13490|
ZF|2014-10-16|20140901|2014-10-15 09:34:31|20141015093154006|201410150933549978|486.09|2.43|
486.09|156|48609|
ZF|2014-10-16|20140901|2014-10-15 10:50:45|20141015104838002|201410151050370080|288.03|1.44|
288.03|156|28803|
ZF|2014-10-16|20140901|2014-10-15 10:51:53|20141015104945003|201410151051440082|363.23|1.82|
363.23|156|36323|
ZF|2014-10-16|20140901|2014-10-15 10:58:26|20141015105613007|201410151058120091|484.89|2.42|
484.89|156|48489|
ZF|2014-10-16|20140901|2014-10-15 11:13:48|20141015111129006|201410151113270125|11.04|0.06|
11.04|156|1104|
ZF|2014-10-16|20140901|2014-10-15 11:30:13|20141015112613000|201410151124570142|2.58|0.01|2.58|
156|258|
ZF|2014-10-16|20140901|2014-10-15 11:51:04|20141015114851021|201410151150540177|0.01|0.00|0.01|
156|1|
ZF|2014-10-16|20140901|2014-10-15 11:57:05|20141015115450023|201410151156540181|0.01|0.00|0.01|
156|1|
ZF|2014-10-16|20140901|2014-10-15 13:55:05|20141015135538000|201410151354180249|1234.56|6.17|
1234.56|156|123456|
ZF|2014-10-16|20140901|2014-10-15 13:58:30|20141015135618000|201410151357480254|1234.56|6.17|
```

签名信息：

```
YXVowwkTq6/yRHJpvT1A1vPEM1CWiUgR9Cgm0DfnP7M3RS13wD99vfFd2wvvbONKHlH8wBcHkk9VTLK
GOGt7C66esYeR78cEixcxi9UDMcVQFgUrUfMjArJxSoMq8arf5bGD3dluZ14DWr3JVcNdFvVrHeAtwT
/mlGfg3OQsgIM=
```

支付类型说明：
ZF：支付成功
TH:退款，通联系统扎差成功，退款请求发送到银行
CX:冲销成功，未能成功退款，资金已经退回给商户

4. HTTP 方式下载对账文件接口

4.1 下载请求参数说明

注意：测试环境提交地址支持 http，不支持 https；生产环境请使用 https 提交

提交地址	测试环境：http://116.236.252.102:8019/ms/onlinebill/download 生产环境： https://merchant.allinpay.com/ms/onlinebill/download			
参数名称	参数含义	长度	可否为空	参数说明
报文参数				
mchtCd	商户号	15	不可空	由通联统一分配
settleDate	清算日期	10	不可空	格式为 yyyy-MM-dd， 是指通联清算的自然日期（通联 T 日清算）
signMsg	签名字符串		不可空	摘源串为商户号、结算日期及 md5key 字符串连接而成，再通过 MD5 摘要产生。 例如： 用于 MD5 摘要的字符串： 1000201101019002011-03-01pwd123 生成的 MD5 摘要即 signMsg 为： 667C0C64535346764D1A0DF25B320937

4.2 错误代码描述

* ERRORCODE:001 ERRORDES:系统繁忙.请稍候.....
* ERRORCODE:002 ERRORDES:请传入有效的商户号,结算日期,signMsg
* ERRORCODE:003 ERRORDES:结算日期格式错误(格式为 yyyy-MM-dd)
* ERRORCODE:004 ERRORDES:商户号不存在或者 MD5key 没有设置
* ERRORCODE:005 ERRORDES:摘要信息验证有误
* ERRORCODE:006 ERRORDES:没有相应的对账信息

4.3 对账文件验证签名

- 为保障文件来源的安全性，商户需要对下载的文件先进行验证签名，再解析对账文件；
- 开发包安装方法及验签方法请参考《通联互联网支付网关商户接口技术规范》；
- 明文和密文在文件中以空行分开，空行及前面的行为明文，空行后的一行为密文；(请参考 2.1.2 的对账文件图示)
- 其中明文需要先进行 MD5 摘要，再调用验证签名方法。

4.4 JAVA 示例代码

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.security.KeyManagementException;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;
import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSession;
import javax.net.ssl.SSLSocketFactory;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;
import org.apache.commons.httpclient.HttpException;
import org.bouncycastle.util.encoders.Base64;
import com.allinpay.ets.client.SecurityUtil;

public class CheckfileDownloadTest {
    public static void main(String[] args) {
        String serverUrl = " http(s)://通联支付网关地址
/member/checkfiledown/CheckFileDownLoad/checkfileDownLoad.do";
        String mchtCd = "100020110202002"; // 商户号
        String settleDate = "2011-02-16"; // 格式为 yyyy-MM-dd
        String md5key = "1234567890"; // md5key
```

```
String fileAsString = ""; // 签名信息前的对账文件内容
String fileSignMsg = ""; // 文件签名信息
boolean isVerified = false; // 验证签名结果
try {
    // 得到摘要(MD5Encode 函数的传入参数为商户号+结算日期+md5key)
    String signMsg = SecurityUtil.MD5Encode(mchtCd + settleDate
            + md5key);

    // 建立连接
    URL url = new URL(serverUrl + "?mchtCd=" + mchtCd
            + "&settleDate=" + settleDate + "&signMsg=" + signMsg);
    CheckfileDownloadTest test = new CheckfileDownloadTest();
    HttpURLConnection httpConn = test.getHttpsURLConnection(url);
    httpConn.connect();

    // 读取交易结果
    BufferedReader fileReader = new BufferedReader(
            new InputStreamReader(httpConn.getInputStream()));
    StringBuffer fileBuf = new StringBuffer(); // 签名信息前的字符串
    String lines;
    while ((lines = fileReader.readLine()) != null) {
        if (lines.length() > 0) {
            // 按行读，每行都有换行符
            fileBuf.append(lines + "\r\n");
        } else {
            // 文件中读到空行，则读取下一行为签名信息
            fileSignMsg = fileReader.readLine();
        }
    }
    fileReader.close();

    fileAsString = fileBuf.toString();
    System.out.println("File: \n" + fileAsString);
    System.out.println("Sign: \n" + fileSignMsg);

    // 验证签名：先对文件内容计算 MD5 摘要，再将 MD5 摘要作为明文进行
验证签名
    String fileMd5 = SecurityUtil.MD5Encode(fileAsString);
    isVerified = SecurityUtil.verifyByRSA("c:/TLCert.cer", fileMd5
            .getBytes(), Base64.decode(fileSignMsg));

    if (isVerified) {
```

```
                    // 验证签名通过，解析交易明细，开始对账
                    System.out.println("验证签名通过，解析交易明细，开始对
账...");

                } else {
                    // 验证签名不通过，丢弃对账文件
                    System.out.println("验证签名不通过，丢弃对账文件。");

                }
            } catch (HttpException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            } catch (SecurityException e) {
                e.printStackTrace();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        /*
         * 得到HttpURLConnection对象 @author: Robin @param : URL
         * @return:HttpURLConnection @date : 2011-03-10
         */
        public HttpURLConnection getHttpsURLConnection(URL url) {

            try {
                HttpURLConnection httpConnection = (HttpURLConnection) url
                        .openConnection();

                if ("https".equals(url.getProtocol())) // 如果是https协议
                {
                    HttpsURLConnection httpsConn = (HttpsURLConnection)
httpConnection;
                    TrustManager[] managers = { new MyX509TrustManager() };// 证
书过滤
                    SSLContext sslContext;
                    sslContext = SSLContext.getInstance("TLS");
                    sslContext.init(null, managers, new SecureRandom());
                    SSLSocketFactory ssf = sslContext.getSocketFactory();
                    httpsConn.setSSLSocketFactory(ssf);
                    httpsConn.setHostnameVerifier(new MyHostnameVerifier());//
```

主机名过滤

```
                    return httpsConn;

                }
                return httpConnection;

            } catch (NoSuchAlgorithmException e) {
                e.printStackTrace();
            } catch (KeyManagementException e) {
                e.printStackTrace();
            } catch (IOException e1) {
                e1.printStackTrace();
            }
            return null;

        }

        /*
         * 编写证书过滤器 @author: Robin @date : 2011-03-10
         */
        public class MyX509TrustManager implements X509TrustManager {

            /**
             * 该方法体为空时信任所有客户端证书
             *
             * @param chain
             * @param authType
             * @throws CertificateException
             */
            public void checkClientTrusted(X509Certificate[] chain, String
authType)
                    throws CertificateException {
            }

            /**
             * 该方法体为空时信任所有服务器证书
             *
             * @param chain
             * @param authType
             * @throws CertificateException
             */
```

```
            public void checkServerTrusted(X509Certificate[] chain, String
authType)
                    throws CertificateException {
            }

            /**
             * 返回信任的证书
             *
             * @return
             */
            public X509Certificate[] getAcceptedIssuers() {
                return null;
            }
        }

        /*
         * 编写主机名过滤器 @author: Robin @date : 2011-03-10
         */
        private class MyHostnameVerifier implements HostnameVerifier {

            /**
             * 返回 true 时为通过认证 当方法体为空时，信任所有的主机名
             *
             * @param hostname
             * @param session
             * @return
             */
            public boolean verify(String hostname, SSLSession session) {
                return true;
            }
        }
    }
```

执行输出：

```
File:
20110216||1|1|200.00|0|0.00|0.50|200.00
ZF|2011-02-16|100020110202002|2011-02-15
18:18:27|NO20110215182000|20110215181827001|200.00|0.50|200.00|0|2000
0.00
```

Sign:
T4eB4OMvGUX2oYphcgLAAi7JGNPhipB9AcHcPFaQbx1J3AVYp18ppzH1roW/bMUlMm5ee
k9vMl4zlvvJ+5Fcy1bm2TSZjMXBzrL09SlpyGnIpDrJdAJQKXE7otcr/WoJ6FWqs1/KU9
wWaY4NhJbUEu+gLMH2mC7L1fmw8sIRDDA=

验证签名通过，解析交易明细，开始对账...