云账户结算系统

实时结算模式技术文档

v1.3.18

云账户™

文档修订控制记录（按时间降序排列）

修订日期	修订内容	版本号	修订人
2018-04-23	增加实时下单、查询订单以及回调的打款备注	v1.3.18	呼文豹
2018-03-31	增加余额查询接口	v1.3.17	杨宜
2018-01-26	修改微信、支付宝限额	v1.3.16	杨宜
2017-11-27	增加日订单接口，查单接口增加时间	v1.3.15	杨宜
2017-11-11	增加日流水接口，查单接口增加时间	v1.3.14	杨宜
2017-10-28	增加注意事项	v1.3.13	杨宜
2017-09-15	增加商户用户身份证上传接口	v1.3.12	杨宜
2017-08-15	增加数据上报接口	v1.3.11	杨宜
2017-08-02	status_detail增加更多失败原因，增加status_detail_message字段，调整通道说明，细化挂单原因	v1.3.10	杨宜、邹永强
2017-07-31	修改微信支付相关说明，重画订单状态图	v1.3.9	杨宜、徐飞
2017-07-31	修复版本管理错误，更新为正确的文件对账内容	v1.3.8	邓智杰
2017-07-27	增加支付宝授权接口	v1.3.7	周军
2017-07-26	消息通知增加商户订单号和交易流水号	v1.3.6	周军
2017-07-25	补充订单详细状态码	v1.3.5	周军
2017-07-24	补充消息通知说明	v1.3.4	邓智杰
2017-07-13	增加身份证实名验证接口	v1.3.3	赵斌
2017-06-26	增加银行四要素验证接口	v1.3.2	赵斌
2017-06-02	统一消息通知内容与“查询一个订单”返回一致	v1.3.1	邓智杰
2017-05-26	增加文件对账功能	v1.3.0	邓智杰
2017-05-14	增加java版本3des代码示例	v1.2.9	杨宜
2017-05-13	增加消息通知接口，目前三种通知情况：打款成功、打款失败、银行退汇	v1.2.8	邓智杰、邹永强
2017-05-12	修改订单接口参数、增加三要素鉴权接口、增加错误码列表、增加订单状态列表	v1.2.7	杨宜、徐飞

修订日期	修订内容	版本号	修订人
2017-05-07	修改更新了目前三个支付渠道最新的限额变化	v1.2.6	徐飞
2017-04-13	增加回一个实时接口的参数bank_name	v1.2.5	徐飞
2017-04-01	新增微信支付实时接口	v1.2.4	徐飞
2017-04-01	实时订单接口新增验证模式等参数	v1.2.3	徐飞
2017-03-29	实时下订接口增加对公司户、海外用户信息、存折用户的打款支持	v1.2.2	邓智杰
2017-03-02	增加支付宝结算文档，删除微信支付部分	v1.2.1	徐飞
2016-12-23	打款只需要四要素；规范3deskey和appkey	v1.2.0	杨宜
2016-12-22	增加3des python实现，修改接口名	v1.1.1	杨宜
2016-12-21	增加四要素鉴权接口，调整样式	v1.1.0	邓智杰
2016-11-23	细化打款到银行卡和到微信的对比，细化订单状态	v1.0.6	邹永强
2016-11-21	微调打款到微信的接口参数	v1.0.5	徐飞
2016-11-18	优先付款到银行卡, 修改实时付款接口参数	v1.0.4	杨宜
2016-11-06	补充打款到微信的两种账户方案比较，对比打款到银行卡方案，完成挂单等逻辑	v1.0.3	邹永强
2016-11-05	补充微信支付实时下单的API和额度限制	v1.0.2	杨宜
2016-09-30	补充整体流程，完善各节内容，调整格式	v1.0.1	邹永强
2016-09-28	完成整体文档	v1.0.0	杨宜

目录

第一章 核心价值

本文档介绍云账户结算系统的实时结算模式的技术对接。

云账户结算系统提供SaaS服务，支持自由职业者平台等作为商户接入结算系统，完成其用户的资金结算。云账户结算系统支持实时结算和批量结算两种模式。实时结算模式中，用户可以随时发起打款且资金通常实时到账，过程自动处理无人干预。批量结算模式中，商户通过云账户结算系统(https://jiesuan.yunzhanghu.com)上传待付款excel表格，此时对表格中全部用户统一发起结算，一批结算订单经过相关经纪公司财务人员(或商户相关财务人员)的流程确认后完成资金结算到账。

批量结算模式可以不用技术对接，而由商户财务人员登陆云账户结算系统完成操作。

本文档仅描述云账户结算系统的实时结算模式，并通常简称为结算系统。结算系统向商户提供以下核心价值：

1 自动和实时结算至银行卡

自动和实时结算至银行卡是结算系统默认的模式，可支持用户任意时间提交打款请求，打款过程无需人工干预，承诺提交打款请求后下一个工作日到账，通常实时到账，无单人和单日限额。

打款至银行卡比打款至支付宝和微信的优势是：由于打款至支付宝或者微信方案对于明星用户的单日和单笔的金额限制，随着商户中明星用户的收入增加，打款至支付宝或者微信的方案应对规模增加的压力会增大。另外，由于支付宝或者微信的零钱打款至银行卡还需额外的成本，且大额打款至银行卡的时间可能超过一个工作日，对于平台中的明星用户体验不佳。

转账给个人银行卡账户，单笔最低0.01元，单笔最高500万元，单日最高2000万元。

因银行卡通道单笔支付有0.5元成本（成本由云账户承担），建议商户单笔银行卡付款大于50元。

2 自动和实时结算至支付宝

支持用户任意时间提交打款请求，打款至支付宝，承诺提交打款请求后1个工作日到账，但通常实时到账，打款过程无需人工干预。

转账给个人支付宝账户，单笔最低0.01元，最高100万元，单日最高2000万元。

3 自动和实时结算至微信支付（红包）

支持用户任意时间提交打款请求，打款至微信支付，承诺提交打款请求后1个工作日到账，但通常实时到账，打款过程无需人工干预。使用该功能需要商户预先申请成为云账户的微信子商户（需提供以下信息：1. 联系人；2. 手机号; 3. 常用邮箱; 4. 互联网文化经营许可证；5. 营业执照；6. 社会信用代码； 7. 对公账户及开户信息；8. 客服电话（能打通））。

打款到微信支付对用户有额度限制，因此明星用户的大额打款必须分拆成多笔小额分在多日完成：

(1) 使用微信红包方式进行付款时，单笔限额200元，可以通过在微信商户后台注明原因申请提高额度，最高为4999元，且用户需要额外进行拆红包动作进行收款确认
(2) 单笔最小金额默认为1元
(3) 每个用户每天最多可付款100次
(4) 给同一个用户付款时间间隔不得低于15秒

4 流水和统计

流水方面，可提供每一笔打款请求的订单状态查询，必要时可按需提供支付通道流水号和打款凭证。 统计方面，可提供每个用户的按日、月结等统计情况。

5 商户管理

提供商户管理平台，可进行商户的基本信息管理，提供充值、查询等功能。

第二章 实时结算准备

为实现实时结算，商户需向结算系统提供以下内容：

1 用户数据准备

1）银行卡打款：提供完整的打款人信息

(1) 银行卡号（必填）
(2) 姓名（必填）
(3) 身份证（必填，用于为该用户报税）
(4) 银行预留手机号（选填）

2）打款至支付宝：提供打款人支付宝账户信息

(1) 支付宝账号（必填）
(2) 姓名（必填，使用支付宝USERID付款并不参与报税可不填）
(3) 身份证（必填，用于为该用户报税）

3）打款至微信零钱：绑定用户的微信账号信息

(1) 用户商户公众号内的 openid（必填）
(2) 姓名（必填，用于为该用户报税）
(3) 身份证（必填，用于为该用户报税）

商户通过技术开发，使得用户在商户的打款页面中，可以授权给结算系统，绑定用户的微信账号以便完成付款。本步骤获取用户在结算系统商户内的 openid，以确定付款的用户身份。

2 开立商户并获取登陆账户和密钥

商户需在结算系统中开立商户，需提供下述信息，包括：

(1) 公司名称
(2) 联系人名称，注意可以有多名联系人，方便多人操作和接收提醒，均需提供电话等信息
(3) 联系人手机号码
(4) 联系人邮箱
(5) 服务器IP列表（以便加入白名单）
(6) 确认合作的经纪公司
(7) 确认合同约定的服务费率、手续费率等
(8) 如果希望接受结算系统的订单通知，则需提供接收通知的URL，可选

商户开立后，会生成商户的登录账号，包含商户号即技术对接时的dealer_id，商户联系人的登录名和初始密码，以便登陆结算系统平台查看商户的流水和统计等信息。
每个商户会获取3deskey和 appkey两个密钥，用于技术对接中认证商户的身份。

确认合作的经纪公司后，可以获得经纪公司的ID，用于技术对接中的broker_id。如果商户与多个经纪公司签约，则需要管理其用户与经纪公司的所属关系。

3 预先充值

商户需要预先充值以支持一个打款周期（比如1天、3天或者1周）的用户打款，用于结算系统完成打款付款，并注意保持余额充足。
当余额剩余低于一定阈值时，结算系统会手机短信或者邮件通知商户联系人，进行预警。此阈值可以登录结算系统自助修改。

4 打款挂单

1）余额不足导致挂单

如果余额不足以打款时，结算系统会进行挂单处理，商户的财务人员可以看到打款订单处于“待打款（暂停处理）”状态，直到充值使得余额充足时，自动或手动放行。

2）打款至微信支付时，超过微信日限额导致挂单

打款至微信支付时，对于单个用户只允许单笔不超过单用户日限额的打款，对于超过的拒绝并提示用户分拆。
而随着平台收入规模加大，容易在收入高峰的单日中突破单日商户限额，从而导致用户的打款无法当日完成。
当累积的打款额度超过单日限额而不超过单日限额的两倍时，可以保障至少第二日付款，因此接受用户请求，但对用户打款请求进行挂单，完成打款请求排队，并在第二日恢复额度时放行打款。
当累积的打款额度超过单日限额的两倍时，拒绝用户的打款请求，并提示用户第二日进行。

注意，打款到银行卡时，不会有对商户的单日限额，因此不会出现本原因导致的打款挂单。

3）通道维护导致挂单

当通道（银行卡、支付宝、微信）维护时，系统会对订单自动进行挂单，维护结束后自动对订单进行放行，常见于银行卡通道。

4）税务管理风险导致挂单

当某笔订单导致商户的月人均打款金额超过合同约定的阈值时，该笔订单会被挂单。某笔订单超过合同约定的单人月最高打款金额时，该笔订单会在当月被持续挂单。

5 退汇

当用户微信红包未领取（微信通道结算），打款后第二日，结算通道会退款给云账户，云账户在收到退款后再退款给商户（将打款金额及手续费退回商户）。

第三章 实时结算的订单流程

实时结算中，单个订单的状态处理流程如下：

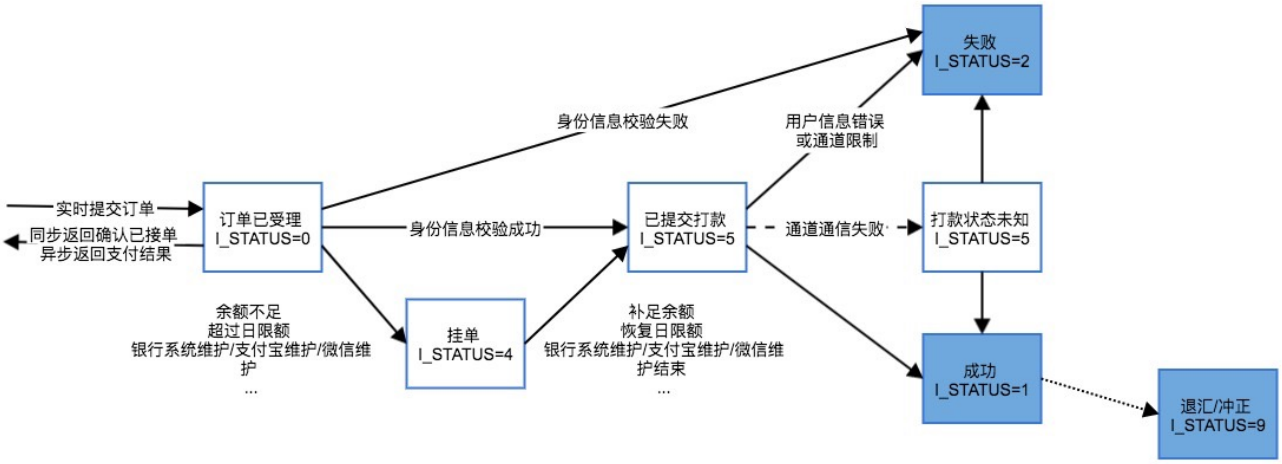
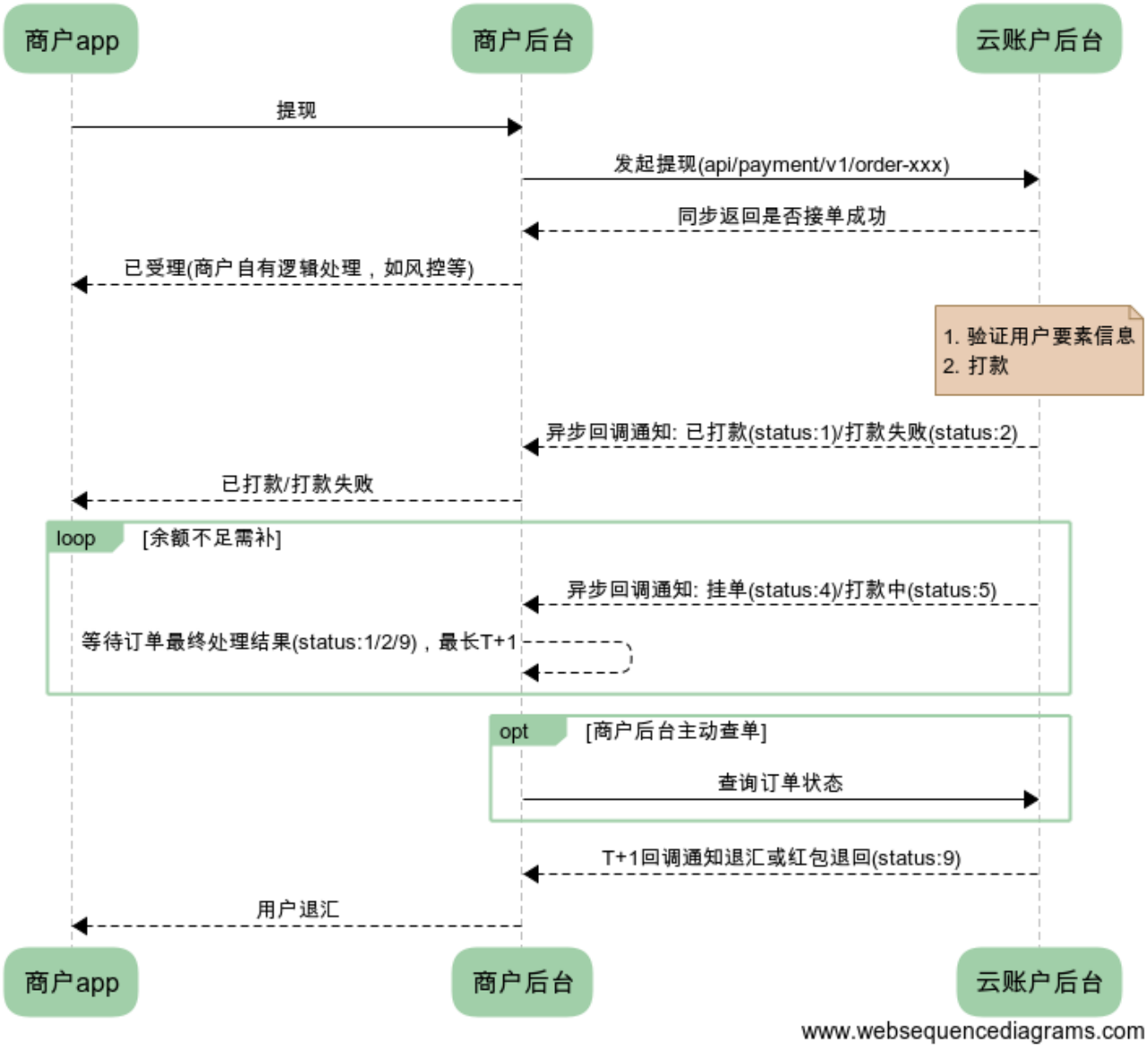


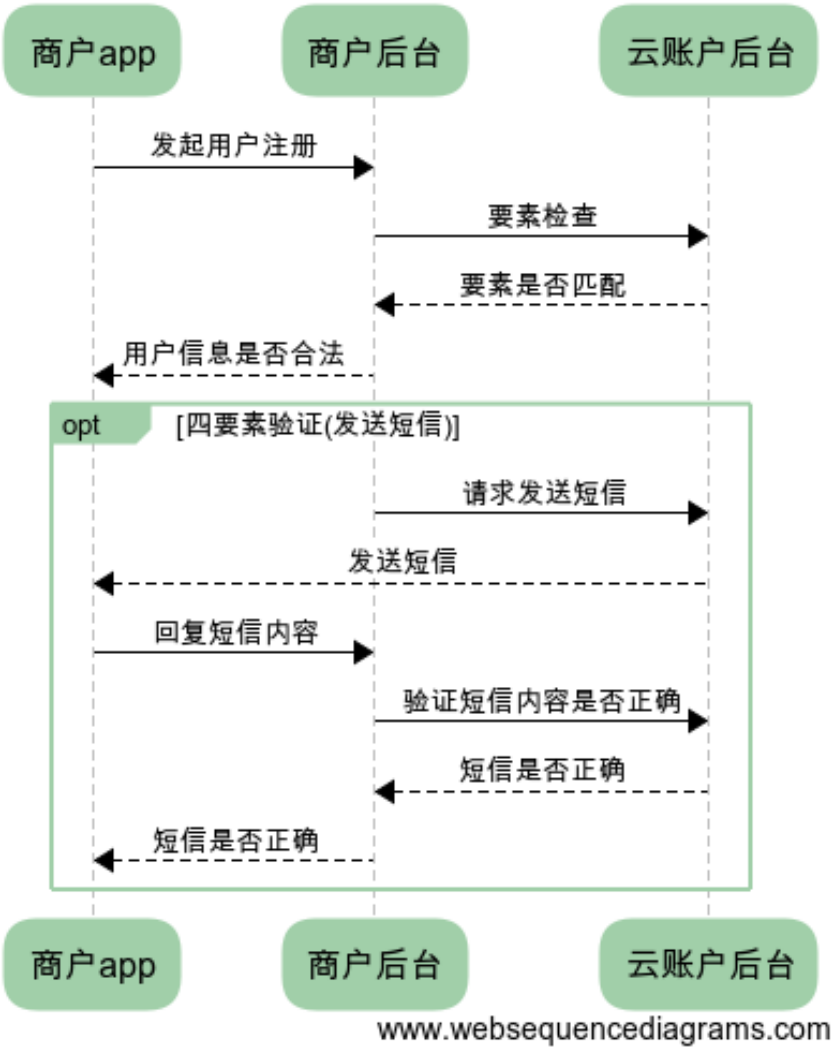
图1：单个订单的处理流程

对于订单状态及对账，商户可以使用第五章中的查单接口（/api/payment/v1/query-realtime-order）进行实时查询。或是通过接收回调通知获取订单状态的改变（建议通过回调通知异获取）。

实时结算中，商户对接泳道图示例如下：



要素验证对接泳道图示例如下（可选，根据商户自身情况自主选择接入；如商户不需要对接可直接跳过）：



第四章 接口规范

1. 接口参数概述

1）所有接口的 header 中传入两个参数

字段	类型	解释
dealer-id	String	填 商户代码（由结算系统分配）
request-id	String	每个 request 的 id，要求每次请求 id 不一样，会在 response 中原样返回

注意：为保证安全，所有接口调用均经过2个步骤
1.1) 3des 加密 —— 需要 3deskey（详见2节）
1.2) hmac(sha256) 签名 —— 需要 appkey（详见3节）

2）所有接口中的参数固定为以下字段（注意 POST 使用 x-www-form-urlencoded, 在 GET 请求时 Query 需要经过 urlencode）

字段	类型	说明
data	String	经过加密后的具体数据
mess	String	随机数，用于签名
timestamp	int	时间戳，精确到秒
sign	String	签名
sign_type	String	签名方式，现在固定填 sha256

1.3 请求实例

data=MhjTl1rWjFxHJ5e&mess=12313&timestamp=123457&sign=b6516baa210161df6f34f1efec2a7c484fd7920fed5640e066e970d8a3f01499&sign_type=sha256

2. 加密算法（3des + base64）

加密采用 3des 算法, 填充采用 PKCS5Padding。

加密后数据再用 base64 进行编码。

1）加密算法的 PHP 实现

```php
<?php

$des3key = '123456788765432112345678';

$desUtils = new DesUtils($des3key);

$data = <<<EOF
{"order_id": "201609010016562012987","batch_id": "201609_-NDJ","issue_id": "201609-NDJ","reserve_id": "56244623 20160901-1","broker_id": "sixcn","anchor_id": "56244623","real_name": "张三","card_no": "6228880199872220","bank_name": "招商银行","bank_branch": "上地分行","bank_prov": "北京","bank_city": "北京","phone_no": "13488795491","id_card": "5326123123123211","yyyy_mm": "201609","pay": "100000.00"}
EOF;

$result = $desUtils->encrypt($data);

echo "result: $result\n";


class DesUtils
{

    private $des3key; // 密钥向量
    private $iv;  // 混淆向量
    private $mode = MCRYPT_MODE_CBC;

    /**
     * 构造，传递二个已经进行base64_encode的KEY与IV
     *
     * @param string $des3key
     * @param string $iv
     */
    function __construct($des3key, $iv = null)
    {
        $this->des3key = $des3key;
        $this->iv = $iv;
    }

    /**
     * 加密
     * @param <type> $value
     * @return <type>
     */
    public function encrypt($value)
    {
        $td = mcrypt_module_open(MCRYPT_3DES, '', $this->mode, '');
        $iv = substr($this->des3key, 0, 8);
        $value = $this->PaddingPKCS7($value);
```

```
        @mcrypt_generic_init($td, $this->des3key, $iv);
        $dec = mcrypt_generic($td, $value);
        $ret = base64_encode($dec);
        mcrypt_generic_deinit($td);
        mcrypt_module_close($td);
        return $ret;
    }

    /**
     * 解密
     * @param <type> $value
     * @return <type>
     */
    public function decrypt($value)
    {
        $td = mcrypt_module_open(MCRYPT_3DES, '', $this->mode, '');
        $iv = substr($this->des3key, 0, 8);
        @mcrypt_generic_init($td, $this->des3key, $iv);
        $ret = trim(mdecrypt_generic($td, base64_decode($value)));
        $ret = $this->UnPaddingPKCS7($ret);
        mcrypt_generic_deinit($td);
        mcrypt_module_close($td);
        return $ret;
    }

    private function PaddingPKCS7($data)
    {
        $block_size = mcrypt_get_block_size('tripledes', $this->mode);
        $padding_char = $block_size - (strlen($data) % $block_size);
        $data .= str_repeat(chr($padding_char), $padding_char);
        return $data;
    }

    private function UnPaddingPKCS7($text)
    {
        $pad = ord($text{strlen($text) - 1});
        if ($pad > strlen($text)) {
            return false;
        }
        if (strspn($text, chr($pad), strlen($text) - $pad) != $pad) {
            return false;
        }
        return substr($text, 0, -1 * $pad);
    }
}
```

2）加密算法的 Golang 实现

```
// 3DES加密
```

```
func TripleDesEncrypt(origData, des3key []byte) ([]byte, error) {
    block, err := des.NewTripleDESCipher(des3key)
    if err != nil {
        return nil, err
    }
    origData = PKCS5Padding(origData, block.BlockSize())
    // origData = ZeroPadding(origData, block.BlockSize())
    blockMode := cipher.NewCBCEncrypter(block, des3key[:8])
    crypted := make([]byte, len(origData))
    blockMode.CryptBlocks(crypted, origData)
    return crypted, nil
}

// 3DES解密
func TripleDesDecrypt(crypted, des3key []byte) ([]byte, error) {
    block, err := des.NewTripleDESCipher(des3key)
    if err != nil {
        return nil, err
    }
    blockMode := cipher.NewCBCDecrypter(block, des3key[:8])
    origData := make([]byte, len(crypted))
    // origData := crypted
    blockMode.CryptBlocks(origData, crypted)
    origData = PKCS5UnPadding(origData)
    // origData = ZeroUnPadding(origData)
    return origData, nil
}

func PKCS5Padding(ciphertext []byte, blockSize int) []byte {
    padding := blockSize - len(ciphertext)%blockSize
    padtext := bytes.Repeat([]byte{byte(padding)}, padding)
    return append(ciphertext, padtext...)
}

func PKCS5UnPadding(origData []byte) []byte {
    length := len(origData)
    // 去掉最后一个字节 unpadding 次
    unpadding := int(origData[length-1])
    return origData[:(length - unpadding)]
}
```

3）加密算法的 Python 实现

```
import pyDes
import base64


des3key = "123456788765432112345678"
iv = des3key[0:8]
```

```
data = r'{"order_id": "201609010016562012987","batch_id": "201609_-NDJ","issue_id": "201609-NDJ","reserve_id": "56244623 20160901-1","broker_id": "sixcn","anchor_id": "56244623","real_name": "张三","card_no": "6228880199872220","bank_name": "招商银行","bank_branch": "上地分行","bank_prov": "北京","bank_city": "北京","phone_no": "13488795491","id_card": "5326123123123211","yyyy_mm": "201609","pay": “100000.00"}'

k = pyDes.triple_des(des3key, pyDes.CBC, iv, pad=None, padmode=pyDes.PAD_PKCS5)
d = k.encrypt(data)
print "Encrypted: %r" % d
print "Decrypted: %r" % k.decrypt(d)
assert k.decrypt(d) == data

print base64.b64encode(d)
```

4）加密算法的 Java 实现

```
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import org.apache.commons.codec.binary.Base64;


public class Util {
    public static byte[] TripleDesEncrypt(byte[] content, byte[] key) throws Exception {
        byte[] icv = new byte[8];
        System.arraycopy(key, 0, icv, 0, 8);
        return TripleDesEncrypt(content, key, icv);
    }

    protected static byte[] TripleDesEncrypt(byte[] content, byte[] key, byte[] icv) throws Exception {
        final SecretKey secretKey = new SecretKeySpec(key, "DESede");
        final Cipher cipher = Cipher.getInstance("DESede/CBC/PKCS5Padding");
        final IvParameterSpec iv = new IvParameterSpec(icv);

        cipher.init(Cipher.ENCRYPT_MODE, secretKey, iv);

        return cipher.doFinal(content);
    }

    public static byte[] TripleDesDecrypt(byte[] content, byte[] key) throws Exception {
        byte[] icv = new byte[8];
        System.arraycopy(key, 0, icv, 0, 8);
        return TripleDesDecrypt(content, key, icv);
    }
```

```
    protected static byte[] TripleDesDecrypt(byte[] content, byte[] key, byte[] icv) throws
Exception {
        final SecretKey secretKey = new SecretKeySpec(key, "DESede");
        final Cipher cipher = Cipher.getInstance("DESede/CBC/PKCS5Padding");
        final IvParameterSpec iv = new IvParameterSpec(icv);

        cipher.init(Cipher.DECRYPT_MODE, secretKey, iv);

        return cipher.doFinal(content);
    }

    public static void main(String[] args) throws Exception {
        byte[] content = "{\"order_id\": \"20160901001656201298 7\",\"init_batch_id\":
\"201609_-NDJ\",\"cur_batch_id\": \"201609_-NDJ\",\"issue_id\": \"201609-
NDJ\",\"reserve_id\": \"56244623 20160901-1\",\"dealer_id\": \"sixcn\",\"broker_id\":
\"0\",\"anchor_id\": \"56244623\",\"real_name\": \"张三\",\"card_no\":
\"6228880199872220\",\"bank_name\": \"招商银行\",\"bank_branch\": \"上地分行
\",\"bank_prov\": \"北京\",\"bank_city\": \"北京\",\"phone_no\": \"13488795491\",\"id_card\":
\"123532612312312321\",\"yyyy_mm\": \"201609\",\"pay\": \"100000.00\"}".getBytes("utf-8");

        byte[] key = "123456788765432112345678".getBytes("utf-8");

        byte[] enc = Util.TripleDesEncrypt(content, key);
        byte[] enc64 = Base64.encodeBase64(enc);
        System.out.println("encrypt: " + new String(enc64));

        byte[] dec64 = Base64.decodeBase64(enc64);
        byte[] dec = Util.TripleDesDecrypt(dec64, key);

        System.out.println("decrypt: " + new String(dec));
    }

}
```

5）加密实例

如以下 json 串：

```
{"order_id": "20160901001656201298 7","init_batch_id": "201609_-NDJ","cur_batch_id":
"201609_-NDJ","issue_id": "201609-NDJ","reserve_id": "56244623 20160901-
1","dealer_id": "sixcn","broker_id": "0","anchor_id": "56244623","real_name": "张三
","card_no": "6228880199872220","bank_name": "招商银行","bank_branch": "上地分行
","bank_prov": "北京","bank_city": "北京","phone_no": "13488795491","id_card":
"123532612312312321","yyyy_mm": "201609","pay": "100000.00"}
```

3des 的 key 为： （(3*8=24长度，内部知晓，由结算系统提供）

123456788765432112345678

加密并 base64 后结果为：

0eUw2Nk2isX+IRlttM7eKomZCAfJlocPw2lG4uVrc0qIlbhOusoy0Pp5hl5sWKRxrp4+/YVTJKqFGlS+dB+0/ApQ9/yVvQmmDdIx4kdUw487S4KruUxMy14qkrqSreTHKkGww71pUX8Xe6jBBZHkGTeibyOWEiaxp3UUNJs7Qxo0rWAct4+0ntTdniNMalsQbKQ7AvhuwWJY+c9HSgMtizJ7IlkhzfOC9SyUr7353G/fd1GX7p2Q+j22b7YceAyZyIgsdEyX24z0tPeh6b49pKJpThf792UJxkxCDq3rdbAfAVpmf+erQ8qBHgD+KElyB5ywHBkTNLxZEW1tzMmd7C76ANmrQgY8oOz7e91DTjbuc8z4Qr8QsWub0+mUNrlGvbYeCfc2+emkeo1moVZXujLof8JmrCZUSstrg9qE1kEptj0SvnELk9RYEAYqXe15O0fLWz/ywRgCmerpWdqRViWp+YV4+9Qrl1bCRlWRp+7Xuc8hNCsIojaOxK0Xpn4gbde1+dX1SYOJu9RqIMDRdTzfllUwzsGexLu9sb3NphodW7brNG5l/lG4k5vs23565DJgbr/zvXYTslLwHkr+RkVYSyhjr0xPF1ZQQHzZdzdN4DiliRAlATE27BDHSTvV

3. 签名算法

(hmac('sha256', data=xxx&mess=xxx&timestamp=xxx&key=appkey, appkey))

签名排列顺序固定为：

data=xxx&mess=xxx&timestamp=xxx&key=appkey

签名实例：

data=0eUw2Nk2isX+IRlttM7eKomZCAfJlocPw2lG4uVrc0qIlbhOusoy0Pp5hl5sWKRxrp4+/YVTJKqFGlS+dB+0/ApQ9/yVvQmmDdIx4kdUw487S4KruUxMy14qkrqSreTHKkGww71pUX8Xe6jBBZHkGTeibyOWEiaxp3UUNJs7Qxo0rWAct4+0ntTdniNMalsQbKQ7AvhuwWJY+c9HSgMtizJ7IlkhzfOC9SyUr7353G/fd1GX7p2Q+j22b7YceAyZyIgsdEyX24z0tPeh6b49pKJpThf792UJxkxCDq3rdbAfAVpmf+erQ8qBHgD+KElyB5ywHBkTNLxZEW1tzMmd7C76ANmrQgY8oOz7e91DTjbuc8z4Qr8QsWub0+mUNrlGvbYeCfc2+emkeo1moVZXujLof8JmrCZUSstrg9qE1kEptj0SvnELk9RYEAYqXe15O0fLWz/ywRgCmerpWdqRViWp+YV4+9Qrl1bCRlWRp+7Xuc8hNCsIojaOxK0Xpn4gbde1+dX1SYOJu9RqIMDRdTzfllUwzsGexLu9sb3NphodW7brNG5l/lG4k5vs23565DJgbr/zvXYTslLwHkr+RkVYSyhjr0xPF1ZQQHzZdzdN4DiliRAlATE27BDHSTvV

mess=12313

timestamp=123457

sign_type=sha256

appkey：（内部知晓，由结算系统提供）

appkey=78f9b4fad3481fbce1df0b30eee58577

则待签名串为：

data=0eUw2Nk2isX+IRlttM7eKomZCAfJlocPw2lG4uVrc0qIlbhOusoy0Pp5hl5sWKRxrp4+/YVTJKqFGlS+dB+0/ApQ9/yVvQmmDdIx4kdUw487S4KruUxMy14qkrqSreTHKkGww71pUX8Xe6jBBZHkGTeibyOWEiaxp3UUNJs7Qxo0rWAct4+0ntTdniNMalsQbKQ7AvhuwWJY+c9HSgMtizJ7llkhzfOC9SyUr7353G/fd1GX7p2Q+j22b7YceAyZyIgsdEyX24z0tPeh6b49pKJpThf792UJxkxCDq3rdbAfAVpmf+erQ8qBHgD+KElyB5ywHBkTNLxZEW1tzMmd7C76ANmrQgY8oOz7e91DTjbuc8z4Qr8QsWub0+mUNrlGvbYeCfc2+emkeo1moVZXujLof8JmrCZUSstrg9qE1kEptj0SvnELk9RYEAYqXe15O0fLWz/ywRgCmerpWdqRViWp+YV4+9QrI1bCRlWRp+7Xuc8hNCsIojaOxK0Xpn4gbde1+dX1SYOJu9RqIMDRdTzfllUwzsGexLu9sb3NphodW7brNG5l/lG4k5vs23565DJgbr/zvXYTslLwHkr+RkVYSyhjr0xPF1ZQQHzZdzdN4DiliRAlATE27BDHSTvV&mess=12313&timestamp=123457&key=78f9b4fad3481fbce1df0b30eee58577

签名后的结果为：

a55c3129a230a4d8abdd5f39bb5f1188a46f6c18adb8146a67f1888650da799d

第五章 开始对接

1. 需要对接哪些方面?

1）必选的服务端对接

商户与结算系统的服务端 REST API 对接，完成实时打款业务的流程。
接口地址：https://api-jiesuan.yunzhanghu.com

2）注意事项

- 超时时间建议设置为30s
- 当下单接口超时无返回时，请使用原单号(order_id)进行重试，防止重复打款(同一个order_id只会被支付一次)
- 支付流程为同步接单，异步支付。下单接口返回成功代表云账户已经收到该订单，支付结果依赖于异步通知，或由商户主动通过查单接口发起

2. 服务端接口列表

服务端接口包括：银行卡实时下单、支付宝实时下单、微信支付实时下单、订单查询、消息通知、银行卡四要素请求鉴权、银行卡四要素确认鉴权、银行卡四要素验证、银行卡三要素验证。

2.1 打款接口

1）银行卡实时下单

接口：{post} /api/payment/v1/order-realtime

```
data=base64(3des(
{
 "order_id": "201609010016562012987", # 商户订单号，由商户保持唯一性(必填)
 "dealer_id": "sixcn",                # 商户代码(必填)
 "broker_id":"111",                   # 经纪公司(必填)
```

```
 "real_name": "张三",                    # 银行开户姓名(必填)
 "card_no": "6228880199872220",          # 银行开户卡号(必填)
 "phone_no": "13488795491",              # 用户或联系人手机号
 "id_card": "5326123123123211",          # 银行开户身份证号
 "pay": "100000.00",                     # 打款金额(必填)
 "anchor_id": "56244623",                # 打款人id(选填)
 "notes": "",                            # 备注信息(选填)
 "pay_remark": ""                        # 打款备注(选填，最大20个字符，一个汉字占2个
                                          字符，不允许特殊字符：' " & | @ % * ( ) - : # ￥)
 }
))
```

请注意：
- pay 为 string 类型
-
-

接口返回：

```
{
 "code": "0000",
 "data": {
   "order_id": "201609010016562012988",
   "ref": “176826728298982", # 云经纪流水
   "pay": "100.00"
 },
 "message": "操作成功",
 "request_id": ""
}
```

2）支付宝实时下单

接口：{post} /api/payment/v1/order-alipay

商户向结算系统发起下单请求

(请注意 pay 为 string 类型)

```
data=base64(3des(
{
 "order_id": "201609010016562012987", # 商户订单号，由商户保持唯一性(必填)
 "dealer_id": "sixcn",                # 商户代码(必填)
 "broker_id": "56244623",             # 经纪公司(必填)
```

```
  "real_name": “张三",                    # 姓名(必填)
  "id_card": "120101198812142146",        # 身份证(必填)
  "card_no": “123@sina.com”,              # 收款人支付宝账户(必填)
  "pay": "1.00"                           # 打款金额（单位为元, 必填）
  "notes": "备注信息",                    # 备注信息(选填)
  "pay_remark": ""                        # 打款备注(选填，最大20个字符，一个汉字占2个
                                            字符，不允许特殊字符：' " & | @ % * ( ) - : # ￥)
 }
))
```

接口返回

```
{
  "code": "0000",
  "data": {
       "pay": “1.00”,
       "ref": “176826728298982", # 云经纪流水
       "order_id": "201609010016562012987"
   },
  "message": "操作成功",
  "request_id": "1234"
}
```

3）微信支付实时下单

接口：{post} /api/payment/v1/order-wxpay

使用结算系统的微信支付进行付款

-步骤1、商户向结算系统接入用户，获取用户的openid
-步骤2、商户向结算系统发起支付请求
(请注意 pay 为 string 类型)

```
data=base64(3des(
{
  "order_id": "201609010016562012987",      # 商户订单号，由商户保持唯一性(必填)
  "dealer_id": “sixcn",                     # 商户代码(必填)
  "broker_id": "56244623",                  # 经纪公司(必填)
  "real_name": “张三",                      # 姓名(必填)
  "id_card": "120101198511142146",          # 身份证(必填)
  "openid": "wx2319u9jk231bhbhbhwad21",     # 商户appid下，某用户的openid(必填)

  "pay": "1.00"                             # 打款金额（单位为元）(必填)
  "notes": "备注",                          #描述信息(选填)
```

```
 "pay_remark": ""                                    # 打款备注(选填，最大20个字符，一个汉字占2个
                                                      字符，不允许特殊字符：' " & l @ % * ( ) - : # ￥)
 }
))
```

接口返回

```
{
 "code": "0000",
 "data": {
      "pay": "1.00",                          # 下单成功
      "ref": “176826728298982",                    # 云经纪流水
      "order_id": "201609010016562012987"
  },
 "message": "操作成功",
 "request_id": "1234"
}
```

TPT模式微信红包打款可能的订单状态：

	红包已发送-未领取	红包已发送-已领取	红包已发送-24小时退回	红包发送失败
订单状态	status 1 status_detail 27	status 1 status_detail 0	status 9 status_detail 213	status 2

PGT模式微信红包打款可能的订单状态：

	系统打款-发送中	系统打款-已领取	系统打款-红包退回
经纪公司打款-发送中	status 1 status_detail 27	status 1 status_detail 27	status 1 status_detail 27
经纪公司打款-已领取	status 1 status_detail 27	status 1 status_detail 0	status 11 status_detail 112
经纪公司打款-红包退回	status 1 status_detail 27	status 11 status_detail 111	status 9 status_detail 213

4）查询一个订单

接口：{get} /api/payment/v1/query-realtime-order?mess=12313&sign_type=sha256&timestamp=1479892123&data=0eUw2Nk2isX%2BIRlttM7eKomZCAfJlocPw2lG4uVrc0qBdIv1NsK1Ug%3D%3D&sign=xxxx

```
data=base64(3des(
{
```

```
"order_id": "201609010016562012999",          // 商户订单号
"channel": "银行卡",          // 银行卡，支付宝，微信(不填时为银行卡订单查询)(选填)
“data_type": "encryption", // 如果为encryption，则对返回的data进行加密(选填)
}
))
```

接口返回

```
{
 "code": "0000",
 "data": {
   "pay": "100.00",
   "anchor_id": "test",
   "broker_amount": "-/-",
   "broker_id": "yunyi",
   "card_no": "6228021232235288058",
   "dealer_id": "sixcn",
   "id_card": "120101198812142146",
   "order_id": "201609010016562012999",
   "phone_no": "18601318616",
   "real_name": "测试",
   "ref": "79822056820047883",
   "notes": "56244623 20160901-1",
   "status": 1,                              // 订单状态码，详见：附录1订单状态码
   "status_detail": 0,                       // 订单详细状态码，详见：附录2订单详细状态码
   "status_message": "已打款",        // 状态码说明，详见：附录1订单状态码
   "status_detail_message": "已打款",   // 状态详细状态码说明，详见：附录2订单详细状态
码
   "sys_amount": “100.00”,
   "pay_remark": "",                                 // 打款备注(选填，最大20个字符，一个汉字占2个
                                                       字符，不允许特殊字符：' " & | @ % * ( ) - : # ￥)
   "tax": "-/-",
   "tax_level": "",
   "created_at": "2017-10-16 20:58:29”,     // 订单接收时间
   "finished_time": "2017-10-16 20:58:30”, // 订单处理时间
   "encry_data": "0eUwilttM7eKrg==",  // 当且仅当data_type为encryption时，返回且仅返
回该加密数据字段
 },
 "message": "操作成功",
 "request_id": "1234"
}
```

请注意：响应中 amount 为 string 类型

5）消息通知

接口：{post} 通知地址在创建商户时进行配置

商户接收到通知后需要返回 success。如果商户反馈给云账户的字符不是 success 这7个字符，云账户服务器会不断重发通知，直到超过24小时22分钟。 一般情况下，25小时以内完成8次通知（通知的间隔频率一般是：4m,10m,10m,1h,2h,6h,15h）

通知内容：

```
data=base64(3des(
{
    "notify_id": "14732279660721952",
    "notify_time": "2017-05-25 11:49:34",
    "data": {
        "pay": "100.00",
        "anchor_id": "test",
        "broker_amount": "-/-",
        "broker_id": "yunyi",
        "card_no": "62280212322352880 58",
        "dealer_id": "sixcn",
        "id_card": "120101198812142146",
        "order_id": "201609010016562012999",
        "phone_no": "18601318616",
        "real_name": "测试",
        "ref": "79822056820047883",
        "notes": "56244623 20160901-1",
        "status": 1,                                    // 订单状态码，详见：附录1订单状态码
        "status_detail": 0,                             // 订单详细状态码，详见：附录2订单详细状态码
        "status_message": "已打款",                      // 状态码说明，详见：附录1订单状态码
        "status_detail_message": "已打款",    // 状态详细状态码说明，详见：附录2订单详细状态码
        "sys_amount": “100.00",
        "pay_remark": "",                               // 打款备注(选填，最大20个字符，一个汉字占2个字符，不允许特殊字符：' " & l @ % * ( ) - : # ￥)
        "tax": "-/-",
        "tax_level": "",
        "sys_wallet_ref": "",                           // 系统打款商户订单号
        "sys_bank_bill": "",                            // 系统打款交易流水号
        "broker_wallet_ref": "",                        // 经纪公司打款商户订单号
        "broker_bank_bill": ""                          // 经纪公司打款交易流水号
    },
}
))
```

6）查询商户余额

接口：{get} /api/payment/v1/query-accounts/?mess=12313&sign_type=sha256&timestamp=1479892123&data=0eUw2Nk2isX%2BIRlttM7eKomZCAfJlocPw2lG4uVrc0qBdIv1NsK1Ug%3D%3D&sign=xxxx

```
data=base64(3des(
{
 "dealer_id": "0933945",     // 商户ID
}
))
```

接口返回

```
{
  "code": "0000",
  "data": {
    "dealer_infos": [
        {
          "broker_id": “yiyun73",                  // 经纪公司ID
          "bank_card_balance": "997.84",       // 银行卡余额
          "is_bank_card": true                     // 是否开通银行卡通道
          "alipay_balance": "0",                   // 支付宝余额
          "is_alipay": false                       // 是否开通支付宝通道
          "wxpay_balance": "997.84",           // 微信余额
          "is_wxpay": false,                       // 是否开通微信通道
          “rebate_fee_balance”: “0.00”,        // 服务费返点余额
        },
        {
          "broker_id": “aoge",                     // 经纪公司ID
          "bank_card_balance": "997.84",       // 银行卡余额
          "is_bank_card": true                     // 是否开通银行卡通道
          "alipay_balance": "0",                   // 支付宝余额
          "is_alipay": false                       // 是否开通支付宝通道
          "wxpay_balance": "997.84",           // 微信余额
          "is_wxpay": false,                       // 是否开通微信通道
          “rebate_fee_balance”: “0.00”,        // 服务费返点余额
        },
    ]
  },
  "message": "操作成功",
  "request_id": "1234"
}
```

2.2 数据接口

1）查询日订单文件

接口：{get} /api/dataservice/v1/order/downloadurl?mess=12313&sign_type=sha256&timestamp=1479892123&data=0eUw2Nk2isX%2BIRlttM7eKomZCAfJlocPw2lG4uVrc0qBdIv1NsK1Ug%3D%3D&sign=xxxx

```
data=base64(3des(
{
 “order _date": "2017-11-11",        // 流水时间
}
))
```

接口返回

```
{
  "code": "0000",
  "data": {
    "order_download_url": "http://xxxxx",
  },
  "message": "操作成功",
  "request_id": "1234"
}
```

下载文件为gz压缩的csv格式文件，格式如下：

商户订单号	订单流水号	商户批次号	姓名	经纪公司ID	订单状态	收款账号	支付渠道	金额	服务费	渠道流水号	短周期授信账单号	服务费账单号	余额账单号	创建时间	完成时间
2017112701	1053817962754221		张**	yiyun**	成功	6228**********52416	银行卡	195	13.65	201711204400000	104475780210006015	-/-	-/-	2017-11-20 00:00:22	2017-11-20 00:00:24
2017112702	1053817962754222		陈**	yiyun**	失败	oQWg**********iusdfzvMZN1HEsdf	微信	200	7.56	100005170120171120300012310620 1	104475780210006015	-/-	-/-	2017-11-20 00:01:39	2017-11-20 00:01:40
2017112703	1053817962754223		李**	yiyun**	成功	6222**********13121	银行卡	10	0.7	201711204400000	104475780210006015	-/-	-/-	2017-11-20 00:01:42	2017-11-20 00:01:43
2017112704	1053817962754224		牟**	yiyun**	成功	134*****431	支付宝	18	1.26	201711204400000	104475780210006015	-/-	-/-	2017-11-20 00:01:42	2017-11-20 00:01:44
2017112705	1053817962754225		陈**	yiyun**	成功	6217**********59820	银行卡	1000	70	201711204499990033	104475780210006015	-/-	-/-	2017-11-20 00:02:01	2017-11-20 00:02:03
2017112705	1053817962754226		张**	yiyun**	成功	6222**********11361	银行卡	110	7.7	201711204499990137	104475780210006015	-/-	-/-	2017-11-20 00:03:03	2017-11-20 00:03:04
2017112706	1053817962754227		聂*	yiyun**	成功	6222**********54786	银行卡	1000	70	201711204499990041	104475780210006015	-/-	-/-	2017-11-20 00:03:47	2017-11-20 00:03:48

注意：

1. 订单状态有以下状态：成功、成功(待领取) 【微信情况】 、失败、退汇、删除、挂单、打款中、-/-。如果出现挂单或打款中或-/-请与云账户工作人员核对。
2. 每日03:45、11:45、16:45会定时生成头一日的订单文件(11:45覆盖03:45，16:45覆盖11:45)
3. order_date的格式为2017-01-01
4. 日账单在服务器上保存90天，超过90天后，请求返回order_download_url为””
5. 当订单中出现“未知”的订单时，可以找云账户工作人员核对，或晚些时间再次下载流水

2）查询日流水文件

接口：{get} /api/dataservice/v1/bill/downloadurl?mess=12313&sign_type=sha256&timestamp=1479892123&data=0eUw2Nk2isX%2BIRlttM7eKomZCAfJlocPw2lG4uVrc0qBdIv1NsK1Ug%3D%3D&sign=xxxx

```
data=base64(3des(
{
 "bill_date": "2017-11-11",  // 流水时间
}
))
```

接口返回

```
{
  "code": "0000",
  "data": {
    "bill_download_url": "http://xxxxx",
  },
  "message": "操作成功",
  "request_id": "1234"
}
```

下载文件为gz压缩的csv格式文件，格式如下：

商户订单号	订单流水号	流水号	流水类型	流水状态	支付渠道	姓名	经纪公司ID	金额	渠道流水号	创建时间	完成时间
2017111101	105381754236469631	10538175482157504 1	经纪公司打款	成功	微信	彭**	yiyun**	6	1000041701201711 2030000016201	2017-11-20 00:00:02	2017-11-20 00:02:38
2017111101	105381754236469631	10538175483200800	经纪公司打款服务费	成功	-/-	彭**	yiyun**	0.42		2017-11-20 00:00:02	2017-11-20 00:02:38
2017111102	105381757468180863	10538175783085313	经纪公司打款	成功	银行卡	刘**	yiyun**	4	20171102020220220	2017-11-20 00:00:04	2017-11-20 00:02:38
2017111102	105381757468180863	1053817578435710	经纪公司打款服务费	成功	-/-	刘**	yiyun**	0.28		2017-11-20 00:00:04	2017-11-20 00:02:38
2017111103	105381778219008382	1053817784072064	经纪公司打款	成功	微信	彭**	yiyun**	3	1000041721201711 203000012324111	2017-11-20 00:00:14	2017-11-20 00:02:38
2017111103	105381778219008382	1053817784146368	经纪公司打款服务费	成功	-/-	彭**	yiyun**	0.21		2017-11-20 00:00:14	2017-11-20 00:02:38
2017111102	105381780522172 81	10538178085984272	经纪公司退汇	成功	微信	刘**	yiyun**	4		2017-11-20 00:00:15	2017-11-20 00:02:38
2017111102	105381780522172820	10538178087030003	经纪公司退汇手续费	成功	-/-	刘**	yiyun**	0.28		2017-11-20 00:00:15	2017-11-20 00:02:38

注意：

1. 当退汇发生时，原打款流水不会改变，会新增对应订单的退汇流水，如图中商户订单号为2017111102的订单
2. bill_date的格式为2017-01-01
3. 日账单在服务器上保存90天，超过90天后，请求返回bill_download_url为""
4. 每日03:30、11:30、16:30会定时生成头一日的账单(11:30覆盖03:30，16:30覆盖11:30)。流水中只有“状态未知”，或“成功(待领取)”(微信情况：会从“成功(待领取)”变为”成功”)的订单状态会改变，其他流水状态不会改变。

5. 当流水中出现“状态未知”的订单时，可以找云账户工作人员核对，或晚些时间再次下载流水。

2.3 用户信息验证接口

1.1）银行卡四要素请求鉴权（下发短信验证码）

接口：{post} /authentication/verify-request

```
data=base64(3des(
{
  "card_no": "6214833103928337",       # 银行卡号
  "id_card": "360232199009115318",     # 身份证号
  "real_name": “张三",                  # 开户人姓名
  "mobile": "18612341234"               # 开户预留手机号
}
))
```

接口返回：

（正确情况）

```
{
  "code": "0000",
  "data": {
       "ref": “rx0g4iiLWoWA=="          # 交易凭证
   },
  "request_id": "1234"
}
```

（出错情况）

```
{
  "code": “2002”,                      # 非 0000 均表示错误,
  “message”: “身份证号有误”,
   "request_id": "1234"
}
```

1.2）银行卡四要素确认鉴权（上传短信验证码）

接口：{post} /authentication/verify-confirm

```
data=base64(3des(
{
  "card_no": "6214833103928337",      # 银行卡号
  "id_card": "360232199009115318",    # 身份证号
  "real_name": “张三",                 # 开户人姓名
  "mobile": "18612341234"             # 开户预留手机号
  "captcha": "861134"                 # 短信验证码
  "ref": "rx0g4iiLWoWA=="             # 交易凭证
 }
))
```

接口返回：

```
{
  "code": "0000",
  "message": "操作成功",
  "request_id": "1234"
}
```

注意：

1. 验证码有效期为1分钟
2. 有效期内重复调用接口，会返回成功，但不会重发短信

2）银行卡四要素验证

接口：{post} /authentication/verify-bankcard-four-factor

```
data=base64(3des(
{
  "card_no": "6214833103928337",      # 银行卡号
  "id_card": "360232199009115318",    # 身份证号
  "real_name": “张三",                 # 开户人姓名
  "mobile": "18612341234"             # 开户预留手机号
}
))
```

接口返回：

```
（正确情况）
```

```
{
```

```
  "code": “0000",
  "message": "操作成功",
  "request_id": "1234"
}
```

（出错情况）

```
{
  "code": “2002”,                              # 非 0000 均表示错误,
  “message”: “身份证号有误”,
   "request_id": "1234"
}
```

3）银行卡三要素验证

接口：{post} /authentication/verify-bankcard-three-factor

```
data=base64(3des(
{
  "card_no": "6214833103928337",       # 银行卡号
  "id_card": "360232199009115318",     # 身份证号
  "real_name": “张三",                          # 开户人姓名
 }
))
```

接口返回：

```
{
  "code": "0000",
  "message": "操作成功",
  "request_id": "1234"
}
```

4）身份证实名验证

接口：{post} /authentication/verify-id

```
data=base64(3des(
{
  "id_card": "360232199009115318",     # 身份证号
  "real_name": “张三",                          # 姓名
 }
))
```

接口返回：

```
{
  "code": "0000",
  "message": "操作成功",
  "request_id": "1234"
}
```

5）支付宝授权

接口：{get} /api/payment/v1/auth-alipay

```
data=base64(3des(
{
  "broker_id": "111"     # 经纪公司(必填)
 }
))
```

接口返回：

```
{
  "code": "0000",
  "data": {
    "info":
"scope=kuaijie\u0026target_id=29399119769856\u0026version=1.0\u0026sign_type=RSA
\u0026format=JSON\u0026auth_type=AUTHACCOUNT\u0026charset=utf-
8\u0026product_id=APP_FAST_LOGIN\u0026biz_type=openservice\u0026timestamp=20
17-07-
27+19%3A06%3A51\u0026method=alipay.open.auth.sdk.code.get\u0026sign=JgxydYEku
T2ihZ8h%2Bk8JMySSRCylk55vfNN1IVQArIhwLgjyfex8ZUPeabF9X%2B47nsbZIpb8B7%2
Fa%2BoHJKYTw29K%2B5iDgL1PkUGVK9v7l7j2SnmsSP6CWHaVM8J%2FcTxnQgbI8li2
vkMo1%2BeMe9r8ucN6reduARO6WNqLHrNaIBBnUi5R5n48zH05bCQapzBe8xIiw6dDwR
CIKcHn5ecKMrT6vEltXeYzIAEHn7hkkbPGE1WC5NlmRFppYx1%2B60dbuUO2oFk2jtDv
%2Fzw43FquEJZXIaTJJc%2BGexUAGOZI2bgqszFWHZhIlaUf7g4%2B8mMkcxkfJSuuua
Nx4X89fz3Y4WA%3D%3D\u0026apiname=com.alipay.account.auth\u0026app_name=mc
\u0026app_id=2017060207405745\u0026pid=2088721068123262"
  },
  "message": "操作成功",
  "request_id": "1234"
}
```

2.4 身份信息上传接口

1）身份证信息上传

接口：{post} /api/payment/v1/sign/idcard/image

```
data=base64(3des(
{
```

```
 "dealer_id": "sixcn",                                  # 商户代码(必填)
 "broker_id":"111",                                     # 经纪公司(必填)

 "real_name": "张三",                                   # 用户姓名(必填)
 "id_card": "5326123123123211",                         # 用户身份证号(必填)
}
))

id_card_image=@id_card.jpg
id_card_image_background=@id_card_background.jpg
```

接口返回：

```
{
 "code": "0000",
 "data": {},
 "message": "操作成功",
 "request_id": ""
}
```

2.5 打款数据上报接口

1）银行卡打款数据上报

接口：{post} /api/notify/order-bankcard

```
data=base64(3des(
{
 "order_id": "201609010016562012987",  # 商户订单号，由商户保持唯一性(必填)
 "dealer_id": "sixcn",                 # 商户代码(必填)
 "broker_id":"111",                    # 经纪公司(必填)

 "real_name": "张三",                  # 银行开户姓名(必填)
 "card_no": "6228880199872220",        # 银行开户卡号(必填)
 "phone_no": "13488795491",            # 用户或联系人手机号(选填)
 "id_card": "5326123123123211",        # 银行开户身份证号(必填)
 "pay": "100000.00",                   # 打款金额(必填)
 "anchor_id": "56244623",              # 打款人id(选填)
 "created_at": "2017-08-15 17:41:48",    # 打款时间
 "result": "1"                           #  打款结果("1"成功 I "2"失败)
 "notes": "",                          #   备注信息(选填，例如打款失败原因)
 }
))
```

接口返回：

```
{
  "code": "0000",
  "data": {
    "order_id": "201609010016562012988",
    "ref": “176826728298982", # 云经纪流水
    "pay": "100.00"
  },
  "message": "操作成功",
  "request_id": ""
}
```

2）支付宝打款数据上报

接口：{post} /api/notify/order-alipay

```
data=base64(3des(
{
  "order_id": "201609010016562012987",  # 商户订单号，由商户保持唯一性(必填)
  "dealer_id": "sixcn",                  # 商户代码(必填)
  "broker_id": "56244623",               # 经纪公司(必填)
  "real_name": “张三",                   # 姓名(必填)
  "id_card": "120101198812142146",       # 身份证(必填)
  "card_no": “123@sina.com”,             # 收款人支付宝账户(支持loginID和userID, 必填)
  "pay": "1.00"                          # 打款金额（单位为元, 必填）
  "created_at": "2017-08-15 17:41:48",     # 打款时间
  "result": "1"                            #  打款结果("1"成功 I "2"失败)
  "notes": "",                           #   备注信息(选填，例如打款失败原因)
 }
))
```

接口返回

```
{
  "code": "0000",
  "data": {
      "pay": “1.00”,
      "ref": “176826728298982", # 云经纪流水
      "order_id": "201609010016562012987"
   },
  "message": "操作成功",
  "request_id": "1234"
```

```
}
```

3）微信打款数据上报

接口：{post} /api/notify/order-wxpay

```
data=base64(3des(
{
 "order_id": "201609010016562012987",        # 商户订单号，由商户保持唯一性(必填)
 "dealer_id": “sixcn",                       # 商户代码(必填)
 "broker_id": "56244623",                    # 经纪公司(必填)
 "real_name": “张三",                        # 姓名(必填)
 "id_card": "120101198511142146",            # 身份证(必填)
 "openid": "wx2319u9jk231bhbhbhwad21",       # 商户appid下，某用户的openid(必填)

 "pay": "1.00"                               # 打款金额（单位为元）(必填)
 "created_at": "2017-08-15 17:41:48",        # 打款时间
 "result": "1"                               # 打款结果("1"成功 | "2"失败)
 "notes": "",                                #  备注信息(选填，例如打款失败原因)
}
))
```

接口返回

```
{
 "code": "0000",
 "data": {
     "pay": "1.00",                  # 下单成功
     "ref": “176826728298982",       # 云经纪流水
     "order_id": "201609010016562012987"
  },
 "message": "操作成功",
 "request_id": "1234"
}
```

第六章 附录

1. 订单状态码（status）

状态码	状态说明	详细解释
-1	删除	被标记为删除的订单
0	已受理	支付订单接收成功，尚未处理

1	已打款	订单提交到支付网关成功
2	打款失败	主要表示订单数据校验不通过
4	待打款(暂停处理)	暂停处理，一般是账户余额不足，充值后可以继续打款
5	打款中(状态未知)	调用支付网关超时等状态异常情况导致，处于等待交易查证的中间状态
8	待打款	订单税务筹划完毕，等待执行打款的状态
9	打款失败（已退款，退汇或者冲正)	支付被退回
11	部分成功	系统打款或经纪公司打款部分成功
15	取消支付	订单被取消，无需支付
18	无需付款	订单支付金额为0的情况会出现此状态

2. 订单详细状态码（status_detail)

状态码	状态说明	详细解释
0	成功	成功
27	微信红包待领取	微信红包已发放，待领取
111	系统打款成功	系统打款成功
112	经纪公司打款成功	经纪公司打款成功
211	系统打款失败	系统打款失败
212	经纪公司打款失败	经纪公司打款失败
213	系统打款和经纪公司打款都失败	系统打款和经纪公司打款都失败
261	支付宝账号错误	支付宝账号错误
262	支付宝账号与姓名不匹配	支付宝账号与姓名不匹配
265	身份证号或收款户名不可以为空	身份证号或收款户名不可以为空
266	支付宝账号受限	用户可能未完成支付宝实名认证
271	微信账号错误	微信账号错误
273	此请求可能存在风险，已被微信拦截	换一个活跃的微信号
274	该用户今日领取红包个数超过限额	该用户今日领取红包个数超过限额
275	微信红包发送红包金额不在限额范围内	发送红包金额不在限额范围内
277	OpenID和AppID不匹配	OpenID和AppID不匹配
278	微信红包超过频率限制，请稍后重试	微信红包超过频率限制，请稍后重试

3. 接口错误码

错误码	错误提示语
0000	成功
1001	签名已过期
1002	请求参数格式不正确
1003	签名错误
1004	加密错误
1005	商户未设置3deskey或没有设置appkey
2001	上传数据有误
2002	已上传过该笔流水
2003	实名认证失败
2006	银行卡号错误
2011	订单不存在
2018	该笔订单不存在
2016	错误的打款金额
2024	订单金额小于0
2027	账户余额不足
2038	该商户不属于该经纪公司
3000	银行系统连接失败，请联系我们修复
5000	不存在的账单