慧用工开放接口文档

v1.3.0

目录

一、 文档简介

1.0 版本变更信息 时间倒叙

版本号	日期	变更信息	姓名
V1.3.0	2020-08-18	重大改版:签约一律升级为电子签约 需要新增两个接口 新增 2.17 自由职业者签约发送验证码接口 新增 2.18 生成自由职业者生成电子签接口 新增签约相关状态码 200406～200504 （这些状态码目前只是接口 2.17 和 2.18 会返回这些状态码） 新增签约状态 1 待生成电子签 （意味着证件照已经识别成功可以获取短信验证码进行电子签约了） 接口 2.2 改为上传证件照接口 识别成功后 签约状态不再返回 2，而是返回 1 代表:待生成电子签	TwT
V1.2.9	2020-07-20	下发（打款）接口 （2.4） 手机号参数改为必填字段。	TwT
V1.28	2020-07-17	新增 200405 返回码	Jacob
V1.2.7	2020-06-23	新增 2.16 商户完税证明下载接口	TwT
V1.2.6	2020-06-11	2.8 对账接口返回值增加 requestNo 和 distributeId 字段	Jacob
V1.2.5	2020-06-05	2.5 下发(打款)查询接口返回值增加 createTime 创建时间字段	Jacob
V1.2.4	2020-06-04	新增 2.13 充值记录列表及开票类目列表接口 新增 2.14 申请开票接口 新增 2.15 开票结果回调借口 回调地址从商户端 接口参数那配置(白名单,公钥那配置)	
V1.2.3	2020-05-29	新增 2.12 （支付宝账号和姓名一致性校验） 接口 新增 200403 和 200404 状态码	TwT

V1.2.2	2020-05-13	自由职业者信息提交（2.1） 自由职业者网签（2.2） 下发接口（2.3） 以上三个接口的证件类型参数 由原来的 1、2、3、4 改为 1、2、3、4、5 （即将港澳台通行证拆分为港澳居民来往内地通行证 和 台湾居民来往大陆通行证） 港澳台身份证签约回调修改为港澳居民来往内地和台湾居民来往大陆通行证签约回调	TwT，Jacob
V1.1.2	2020-03-18	下发(打款)接口增加对备注的校验,并附上备注所允许的内容列表 3.3	Jacob
V1.1.1	2020-02-05	a. 签约支持港澳台身份证，（接口 2.2 新增回调地址字段） b. 签约状态查询增加状态描述 agreeDesc（接口 2.3） c. 下发支持港澳台身份证（接口 2.6 证件类型新增） d. 增加港澳台身份证签约回调接口（接口 2.11）	Jacob TwT
V1.1.0	2020-01-09	新增充值动帐通知接口(2.10)	Fred
V1.0.8	2020-01-08	a.下发(打款)回调接口增加下发金额字段 b.下发(打款)回调和下发(打款)查询接口的下发状态增加待验证状态	Jacob
V1.0.7	2019-12-16	a.下发(打款)回调和下发(打款)查询接口增加银行返回打款备注字段 b.下发（打款）接口回调地址改成非必填。	Jacob
V1.0.6	2019-12-02	a.下发(打款)接口 receiptChannel 收款渠道字段增加 51-中金支付电子账户 b. 职业者添加接口幂等	Jacob Wyt
V1.0.5	2019-11-27	a. 下发（打款）接口增加新的状态码 200000、200101、200201、200301、200401、200402 b. 下发(打款)查询接口慧用工订单号改为不必填	Jacob
V1.0.4	2019-11-15	对账接口	Jon
V1.0.3	2019-11-07	新增下发状态 75-已退款	Jacob

V1.0.2	2019-11-01	a.支持支付宝下发 b.新增（2.8）测试环境下发回调主动触发接口 c.新增状态码 400301 异常请求。	Jacob
V1.0.1	2019-10-17	下发接口添加收款渠道字段	Jacob

1.1 特别声明

1.2 文档说明

此文档专为慧用工的合作商户提供。要求商户的技术人员必须按此文档要求进行接入。此文档适用于自由职业者注册签约、工资下发及余额查询。

1.3 对接流程

1. 合作商户联系慧用工业务人员，创建测试环境账号，并打开开发者模式；
2. 合作商户从慧用工业务人员获取登录账号和初始密码；
3. 合作商户登录慧用工智能结算系统，进入管理中心—>接口对接 页面。
4. 合作商户填写商户验签公钥（由慧用工提供的工具类生成，私钥自己留用，用于签名）、对接服务器 ip 白名单，并保存。
5. 合作商户在管理中心—>接口对接 页面可以通过短信验证码方式获取加密密钥，用于请求参数和响应数据的加密和解密。**该密钥不要透漏给他人。
6. 合作商户在测试环境联调通过之后联系慧用工业务人员开通生产环境账号，然后配置生产环境参数并联调通过后完成上线。

1.4 调用说明

1) 调用方式:接口通过 HTTPS 协议，使用 POST 方式进行接口的调用。
2) 报文格式：application/json;charset=UTF-8.
3) domain：
   i. 测试环境：https://boapi.hvyogo.com
   ii. 生产环境：https://oapi.hvyogo.com

4) 慧用工智能结算系统:
   i. 测试环境：http://bweb.hvyogo.com
   ii. 生产环境：https://web.hvyogo.com

5) 接口请求报文：为 JSON 格式的键值对，包含商户 id 和 businessBody 两部分，商户 id 为慧用工创建并告知合作商户，businessBody 为当前接口的核心业务参数。
6) 安全策略：
   i. 将核心业务参数通过 SHA1withRSA 的方式签名然后使用 AES 密钥加密之后提交，保证了参会不会被泄露和篡改。
   ii. 响应参数中的核心部分也通过 AES 密钥加密，防止泄露。
   iii. 每个商户配置独立的 ip 白名单。
   iv. HTTPS 协议，保证链接安全。

1.5 请求报文

参数名称	参数类型	参数含义	必填	备注
cooperatorId	String	商户 id	Y	由慧用工提供
businessBody	String	将业务参数通过 AES 加密后生成	Y	加密方式为 AES，密钥 key 由慧用工提供（1.3.5 中的密钥），业务参数应该是 json 格式的字符串，以下的每个方法里的参数都为业务参数。

1.6 参数封装举例

1) 按照 ASCII 排序（以下面 2.1 添加职业者接口为例，参数排序后应该为）：
cooperatorId=C631542165494833152&idCard=130527197709082289&timestamp=1571119177664&workerMobile=15754978891&workerName=张三&workerType=1

2) 用 1.3.4 中的私钥签名（签名后生成 sign 为）：
ghMu51+KjpVZJPW2Z7L6BTCebQALnokMDSM/603xbsEJQ5RRMoCR1zAakUEP0ULeAOUb7KRcoZJymbYqziD4t5N/4uHk2kfTur32pl9CMO+vr6v8pfhZK+45R2tBYgFoCAwpS6hDsfwpXKda6wA/4Z37KXgkCCww/GG4Fxeu6TI=

3) sign 追加到业务参数中（json 格式的字符串为）：
{"workerType":1,"cooperatorId":"C631542165494833152","idCard":"130527197709082289","workerMobile":"15754978891","sign":"ZhtxZUETkKkTbbYvGQ/RjuSfo0eryzweE0+tvAcRbpFNNiYcC/1tR3EOW2B2JhU1vOtWaJipdKjxFfRh95JIQuyDUzAoTU7aA18pIXF41BIcE8Gt2ehkL6xr/mff/PlFePTyOob2RpQ9xveAPvc1PXq4sddRIfnFQR/T1bRpjWM=","workerName":"张三","timestamp":"1571119393552"}

4) 使用 1.3.5 中的 AES 密钥加密（上一步字符串加密后为）：
BECBAE4FD2F130F69B840C4D3C118778BAB374263D0AB570D11956E3FD8B1558B4CDAD900B6B011E6450A1E175E06A29C5481318BA175E9DA683D28AE49583EFB5753DBD128327FB49A17C45DD1B62EE7A797775347E06D124023111927B06F9D45AE82218E8F63153D643EB8288FA71FCE26441F358874313DB4D7987BEFF8C618F940542426B5B51C92F5D880976BD656617AA12825C7E6331354D67CA87A235F03CE39F9F526D81FA722F0F747CF8C6103C25242E0264AA22DE26F6970762FA03F3BFB7CF9DC038AAD7B478BAB307E0443F1D1A5FDAAC0B8B3C3E17B9BA81029CBB648E4CB18C626C1CB6F9F20B66D50F386CDEC13745C1072B11737129551A429146BFFE5A9BA422ACA04C35DC305BF09F71D90CF3177F1FBDD4B869B9B1301181EA7CF4BD34CD4DF7BDFBE90864B0F7028E95FC5B7A3434C1AE210B875C718B430CE617C8E55DE9F6CBB3DDDCBD8C4D2D56E679B1430644B06D01A0717F 这个就是 businessBody

5) 请求参数封装（businessBody 是上一步加密串，最终 json 串为）：

{"cooperatorId":"C631542165494833152","businessBody":"BECBAE4FD2F130F69B840C4D3C118778BAB374263D0AB570D11956E3FD8B1558B4CDAD900B6B011E6450A1E175E06A29C5481318BA175E9DA683D28AE49583EFB5753DBD128327FB49A17C45DD1B62EE7A797775347E06D124023111927B06F9D45AE82218E8F63153D643EB8288FA71B87C0DADDB2EF6197CB7F2A08F8521093FABFE69234B180935EA8D23B179D680C7C6E7B4FAC298134BFC23DF1DD5816386E4CABA5E7420E59F15DDBC5576638B59F6C6A2F3F24BDACFD18A48E8A84B367B4F62E1F5229C40030BEDE6ED7FD40B8B25F6BB8840F25F2BFA7C43A539359C62CD9357D5DC893FE9702309B34D2C8E96A2536832AFF283AA0A181888639F54147CADCEF17E66F26B601E34B6A7A812D3D352FE602A094D97B6EFC89CE55F77301181EA7CF4BD34CD4DF7BDFBE90864B0F7028E95FC5B7A3434C1AE210B875C718B430CE617C8E55DE9F6CBB3DDDCBD9093C2C0DAB40B32AEE93385BE8A8974"}

6) 发送数据（用上面参数，请求添加职业者接口）：

二、 接口列表

2.1 自由职业者信息提交接口

2.1.1 接口说明

添加一个职业者

2.1.2 接口地址

https://domain/api/add/worker

2.1.3 请求参数

参数名称	参数类型	参数含义	必填	备注
cooperatorId	String	商户 id	Y	商户 id
timestamp	String	时间戳	Y	Unix 时间戳，（毫秒）,超过 10 分种后该链接无效，以服务器时间为准
workerType	Integer	职业者证件类型	Y	1 身份证;2 港澳居民来往内地通行证 ;3 护照 ; 4 军官证证; 5 台湾居民来往大陆通行证 （目前只支持 1、2 、5）
idCard	String	职业者证件号码	Y	
workerName	String	职业者真实姓名	Y	
workerMobile	String	职业者手机号	Y	
sign	String	签名	Y	将非空参数按照参数名名称按照 ASCII 升序排列(a=1&b=2&c=3)然后使用签名工具签名生成,(该字段不参与签名)

2.1.4 返回参数

参数名称	参数类型	参数含义	备注
statusCode	String	状态码	
statusText	String	状态描述	
data	String	业务描述	AES 加密后的字符串,密钥 key 由慧用工提供，解密后详细见下表

返回业务参数明细

参数名称	参数类型	参数含义	备注
workerId	String	职业者 id	添加成功后返回,用户后续查询

2.2 自由职业者网签接口

2.2.1 接口说明

职业者签约

2.2.2 接口地址

https://domain/api/worker/agreement

2.2.3 请求参数

参数名称	参数类型	参数含义	必填	备注
cooperatorId	String	商户 id	Y	商户 id
timestamp	String	时间戳	Y	Unix 时间戳，（毫秒）,超过 10 分种后该链接无效，以服务器时间为准
workerType	Interge	职业者证件类型	Y	1 身份证;2 港澳居民来往内地通行证 ;3 护照 ; 4 军官证证; 5 台湾居民来往大陆通行证 （目前只支持 1、2 、5）
idCard	String	职业者证件号码	Y	
workerName	String	职业者真实姓名	Y	
idCardFront	String	证件照人像面	Y	Base64 字符串,源文件必须为 JPG（JPEG）格式，照片宽和高必须大于 8px,小于 4000px，照片的大小不超过 3M；图片压缩：同比例缩放一般能满足照片大小要求，若还是过大可对照片质量进行压缩，对与 jpg 文件，建议使用 0.5 ~ 0.7 左右的质量参数压缩
idCardBack	String	证件照国徽面	Y	Base64 字符串,源文件必须为 JPG（JPEG）格式，照片宽和高必须大

				于 8px,小于 4000px，照片的大小不超过 3M；图片压缩：同比例缩放一般能满足照片大小要求，若还是过大可对照片质量进行压缩，对与 jpg 文件，建议使用 0.5 ~ 0.7 左右的质量参数压缩
callbackUrl	String	回调地址	N	用于港澳台居民来往内地、大陆通行证签约回调，因为港澳台居民来往内地、大陆通行证需要手动审核
sign	String	签名	Y	将非空参数按照参数名名称按照 ASCII 升序排列(a=1&b=2&c=3)然后使用签名工具签名生成,(该字段不参与签名)

2.2.4 返回参数

参数名称	参数类型	参数含义	备注
statusCode	String	状态码	
statusText	String	状态描述	
data	String	业务描述	AES 加密后的字符串,密钥 key 由慧用工提供，解密后详细见下表

返回业务参数明细

参数名称	参数类型	参数含义	备注
workerId	String	职业者 id	添加成功后返回,用户后续查询
agreeState	Interge	签约状态	-1 签约失败; 0 待签约； 1 待生成电子签； 2 签约成功;3 待人工审核（港澳台身份证返回）
agreeDesc	String	状态描述	失败的原因

2.3 自由职业者网签查询

2.3.1 接口说明

查询职业者签约状态

2.3.2 接口地址

https://domain/api/worker/agree/state

2.3.3 请求参数

参数名称	参数类型	参数含义	必填	备注
cooperatorId	String	商户 id	Y	商户 id
timestamp	String	时间戳	Y	Unix 时间戳，（毫秒）,超过 10 分种后该链接无效，以服务器时间为准
workerId	String	职业者 id	Y	添加成功后返回
sign	String	签名	Y	将非空参数按照参数名名称按照 ASCII 升序排列(a=1&b=2&c=3)然后使用签名工具签名生成,(该字段不参与签名)

2.3.4 返回参数

参数名称	参数类型	参数含义	备注
statusCode	String	状态码	
statusText	String	状态描述	
data	String	业务描述	AES 加密后的字符串,密钥 key 由慧用工提供，解密后详细见下表

返回业务参数明细

参数名称	参数类型	参数含义	备注
workerId	String	职业者 id	
agreeState	Interge	签约状态	-1 签约失败; 0 待签约; 1 待生成电子签；2 签约成功；3 待人工审核（港澳台身份证返回）
agreeDesc	String	签约状态描述	成功或失败的原因

2.4 下发（打款）接口

2.4.1 接口说明

单个自由职业者下发

2.4.2 接口地址

https://domain/api/distribute/singleDistribute

2.4.3 请求参数

参数名称	参数类型	参数含义	必填	备注
cooperatorId	String	商户 id	Y	商户 id
timestamp	String	时间戳	Y	Unix 时间戳，（毫秒）,超过 10 分种后该链接无效，以服务器时间为准
workerName	String	收款人姓名	Y	长度不超过 60
receiptChannel	Integer	收款渠道	Y	10 银行卡;20 支付宝;30 微信;51 中金支付电子账户(暂时不支持 30 微信)
workerAccount	String	收款人账号	Y	银行卡长度 16-60 位; 支付宝手机号码或者邮箱
workerType	Integer	职业者证件类型	Y	1 身份证;2 港澳居民来往内地通行证 ;3 护照 ; 4 军官证证; 5 台湾居民来往大陆通行证 （目前只支持 1、2 、5）
idNumber	String	职业者身份证号码	Y	长度 18 位
workerMobile	String	职业者手机号码	Y	长度 11 位
distributeAmount	String	打款金额	Y	单位:分,大于 0 的整数
requestNo	String	下发请求单号	Y	商户平台唯一标识,用于查询
callbackUrl	String	下发结果回调地址	N	这里不传的话,会以系统里配置的为准，若系统里也没有配置,则不会回调
remark	String	备注	N	**参照 3.3**

sign	String	签名	Y	将非空参数按照参数名名称按照 ASCII 升序排列(a=1&b=2&c=3)然后使用签名工具签名生成,(该字段不参与签名)

2.4.4 返回参数

参数名称	参数类型	参数含义	备注
statusCode	String	返回状态码	000000 成功;其他失败　请注意,下发结果以回调为准
statusText	String	返回状态描述	
data	String	返回信息	AES 加密后的字符串,密钥 key 由慧用工提供，解密后详细见下表

明细参数说明

参数名称	参数类型	参数含义	备注
distributeAmount	String	下发金额	与上送金额相同
distributeId	String	慧用工平台处理单号	

2.5 下发（打款）查询接口

2.5.1 接口说明

商户单笔下发状态

2.5.2 接口地址

https://domain/api/query/singleQuery

2.5.3 请求参数

参数名称	参数类型	参数含义	必填	备注
cooperatorId	String	商户 id	Y	商户 id
timestamp	String	时间戳	Y	Unix 时间戳，（毫秒）,超过 10 分种后该链接无效，以服务器时间为准
distributeId	String	慧用工平台处理单号	N	
requestNo	String	商户下发请求单号	Y	
sign	String	签名	Y	将非空参数按照参数名名称按照 ASCII 升序排列(a=1&b=2&c=3)然后使用签名工具签名生成,(该字段不参与签名)

2.5.4 返回参数

参数名称	参数类型	参数含义	备注
statusCode	String	返回状态码	000000 成功;其他失败　请注意,下发结果以回调为准
statusText	String	返回状态描述	
data	String	返回信息	AES 加密后的字符串,密钥 key 由慧用工提供，解密后详细见下表

明细参数说明

参数名称	参数类型	参数含义	备注
requestNo	String	商户下发请求单号	
distributeId	String	慧用工平台处理单号	
distributeStatus	Integer	下发状态	9-待验证，10-验证成功，20-验证失败，30-已取消,40-待打款，50-打款中，60-打款成功，70-打款失败,75-已退款,80-退票中,90-已退票
distributeAmount	String	下发金额(单位:分)	与上传的相同
serviceCharge	String	该笔下发的服务	

		费(单位:分)	
remark	String	银行返回打款备注	此字段用来展示打款失败或者已退款的原因,打款成功可忽略此字段
createTime	String	创建时间	格式 yyyy-MM-dd HH:mm:ss

2.6 下发（打款）回调

2.6.1 接口说明

单笔下发结果回调

2.6.2 接口地址

无

2.6.3 请求参数

参数名称	参数类型	参数含义	备注
businessBody	String	将业务参数通过 AES 加密后生成	加密方式为 AES，密钥 key 由慧用工提供，业务参数应该是 json 格式的字符串，以下的每个方法里的参数都为业务参数。
cooperatorId	String	商户 id	

明细参数说明

参数名称	参数类型	参数含义	备注
requestNo	String	商户下发请求单号	
distributeId	String	慧用工平台处理单号	
distributeStatus	Integer	下发状态	9-待验证，10-验证成功，20-验证失败，30-已取消,40-待打款，50-

			打款中，60-打款成功，70-打款失败,75-已退款,80-退票中,90-已退票
serviceCharge	String	该笔下发的服务费(单位:分)	
distributeAmount	String	下发金额(单位:分)	与上传的金额相同
timestamp	String	下单时间	
remark	String	银行返回打款备注	此字段用来展示打款失败或者已退款的原因,打款成功可忽略此字段

2.6.4 返回结果

参数名称	参数类型	参数含义	备注
statusCode	String	返回状态码	000000 成功;其他失败,不需要加密　返回失败或者无返回的情况下会每隔 5 分钟回调一次,一共 10 次
statusText	String	返回状态描述	

2.7 余额查询查询接口

2.7.1 接口说明

查询一个商户的余额

2.7.2 接口地址

https://domain/api/cooperator/balance

2.7.3 请求参数

参数名称	参数类型	参数含义	必填	备注
cooperatorId	String	商户 ID	Y	在慧用工系统注册时产生
timestamp	String	时间戳	Y	Unix 时间戳，（毫秒）,超过 10 分

				种后该链接无效，以服务器时间为准
sign	String	签名	Y	将非空参数按照参数名名称按照 ASCII 升序排列(a=1&b=2&c=3)然后使用签名工具签名生成,(该字段不参与签名)

Response Param：

参数名称	参数类型	参数含义	必返回	备注
statusCode	String	响应 code 码	Y	
statusText	String	Code 码说明	Y	
data	String	接口返回数据体	Y	AES 加密后的字符串,密钥 key 由慧用工提供，解密后详细见下表

Response Param data 明细返回参数说明：

参数名称	参数类型	参数含义	必返回	备注
cooperatorId	String	商户 ID	Y	
cooperatorName	String	商户名称	Y	
disCompany	String	下发公司名称	Y	执行资金下发的公司（慧用工）
balance	String	余额	Y	账户余额(单位：分)

2.8 对账接口

2.8.1 接口说明

查询商户交易记录（对账）

2.8.2 接口地址

https://domain/api/cooperator/account/record

2.8.3 请求参数

参数名称	参数类型	参数含义	必填	备注
cooperatorId	String	商户 ID	Y	在慧用工系统注册时产生
beginDate	String	开始时间	Y	查询开始时间（查询时间范围为 1 周，超出一周返回：查询天数不能大于 7 天）格式：yyyy-MM-dd
endDate	String	结束时间	Y	查询结束时间（查询时间范围为 1 周，超出一周返回：查询天数不能大于 7 天）格式：yyyy-MM-dd
timestamp	String	时间戳	Y	Unix 时间戳，（毫秒）,超过 10 分种后该链接无效，以服务器时间为准
sign	String	签名	Y	将非空参数按照参数名名称按照 ASCII 升序排列(a=1&b=2&c=3)然后使用签名工具签名生成,(该字段不参与签名)

Response Param：

参数名称	参数类型	参数含义	必返回	备注

statusCode	String	响应 code 码	Y	
statusText	String	Code 码说明	Y	
data	String	接口返回数据体	Y	AES 加密后的字符串,密钥 key 由慧用工提供，解密后详细见下表

Response Param data 明细返回参数说明：

参数名称	参数类型	参数含义	必返回	备注
dealAmount	String	交易金额	Y	交易金额(单位：分)
dealId	String	订单号	Y	
dealCode	Integer	交易状态	Y	20.扣款，30.退款
createDate	Date	交易时间	Y	
requestNo	String	商户下发请求单号	N	
distributeId	String	慧用工平台处理单号	N	

2.9 测试环境主动触发回调

2.9.1 接口说明

测试环境主动触发回调(仅限测试环境)

2.9.2 接口地址

https://domain/api/launchCallback

2.9.3 请求参数

参数名称	参数类型	参数含义	必填	备注
cooperatorId	String	商户 ID	Y	在慧用工系统注册时产生
distributeId	String	慧用工平台单号	Y	
status	integer	需要回调的状态	Y	仅支持 60-打款成功;70-打款失败;调用之前非 50-打款中状态触发不起作用
timestamp	String	时间戳	Y	Unix 时间戳，（毫秒）,超过 10 分种后该链接无效，以服务器时间为准
sign	String	签名	Y	将非空参数按照参数名名称按照 ASCII 升序排列(a=1&b=2&c=3)然后使用签名工具签名生成,(该字段不参与签名)

2.9.4 返回结果

参数名称	参数类型	参数含义	备注
statusCode	String	返回状态码	000000 成功;其他失败
statusText	String	返回状态描述	

2.10 商户充值动帐通知

2.10.1 接口说明

商户充值动帐通知，充值加款完成之后可对指定地址发起通知。

2.10.2 接口地址

无

2.10.3 请求参数

参数名称	参数类型	参数含义	备注
businessBody	String	将业务参数通过 AES 加密后生成	加密方式为 AES，密钥 key 由慧用工提供，业务参数应该是 json 格式的字符串，以下的每个方法里的参数都为业务参数。
cooperatorId	String	商户 id	

明细参数说明

参数名称	参数类型	参数含义	备注
orderNo	String	慧用工系统充值单号	
cooperatorId	String	慧用工平台商户 ID	
cooperatorName	String	慧用工平台商户名称	
payerName	String	支付方名称	
payerAccount	String	支付账户	
payerRemark	String	支付备注	
paymentAmount	String	充值金额	单位分, 0 位小数.
payeeName	String	收款方名称	
payeeAccount	String	收款账户	
rechargeStatus	String	充值状态	10-等待加款, 20-加款成功. (仅 20 状态会发起通知, 可作为判断条件)
completeTime	String	完成时间	加款完成时间

2.10.4 返回结果

参数名称	参数类型	参数含义	备注
statusCode	String	返回状态码	000000 表示成功;其他状态码表示失败,不需要加密;返回失败或者无返回的情况下会每隔 5 分钟回调一次,一共 10 次;
statusText	String	返回状态描述	

2.11 港澳居民来往内地通行证和台湾居民来往大陆通行证签约回调

2.11.1 接口说明

港澳居民来往内地通行证和台湾居民来往大陆通行证签约结果回调，回调地址在签约接口传入。

2.11.2 接口地址

无

2.11.3 请求参数

参数名称	参数类型	参数含义	备注
businessBody	String	将业务参数通过 AES 加密后生成	加密方式为 AES，密钥 key 由慧用工提供，业务参数应该是 json 格式的字符串，以下的每个方法里的参数都为业务参数。
cooperatorId	String	商户 id	

明细参数说明

参数名称	参数类型	参数含义	备注
workerId	String	职业者 id	
agreeState	String	签约状态	-1 签约失败; 1 待生成电子签；2 签约成功
agreeDesc	String	签约状态描述	成功或失败的原因

2.11.4 返回结果

参数名称	参数类型	参数含义	备注
statusCode	String	返回状态码	000000 表示成功;其他状态码表示失败,不需要加密;返回失败或者无返回的情况下会每隔 5 分钟回调一次,一共 10 次;
statusText	String	返回状态描述	

2.12 支付宝账号和姓名一致性校验接口

2.12.1 接口说明

支付宝账号和姓名一致性校验接口,仅仅开放给需要给支付宝账号打款的客户。

2.12.2 接口地址

https://domain/api/worker/aliPay/accountNo/check

2.12.3 请求参数

参数名称	参数类型	参数含义	必填	备注
cooperatorId	String	商户 id	Y	商户 id
timestamp	String	时间戳	Y	Unix 时间戳，（毫秒）,超过 10 分种后该链接无效，以服务器时间为准
workerType	Integer	职业者证件类型	Y	1 身份证;2 港澳居民来往内地通行证 ;3 护照 ; 4 军官证证; 5 台湾居民来往大陆通行证 （该接口目前只支持 1）
idCard	String	职业者证件号码	Y	
workerName	String	职业者真实姓名	Y	
aliPayAccountNo	String	支付宝账号	Y	
sign	String	签名	Y	将非空参数按照参数名名称按照 ASCII 升序排列(a=1&b=2&c=3)然后使用签名工具签名生成,(该字段不参与签名)

2.12.4 返回参数

参数名称	参数类型	参数含义	备注
statusCode	String	状态码	
statusText	String	状态描述	
data	String	业务描述	AES 加密后的字符串,密钥 key 由慧用工提供，解密后详细见下表

返回业务参数明细

参数名称	参数类型	参数含义	备注
aliPayAccountNo	String	支付宝账号	原样返回
workerName	String	职业者真实姓名	原样返回

idCard	String	职业者证件号码	原样返回
validState	Boolean	ture 一致 false 不一致	校验是否一致状态
validStateDesc	String	一致 或者 不一致	校验是否一致状态描述

2.13 充值记录列表及开票类目列表

2.13.1 接口说明

充值记录列表及开票类目列表

2.13.2 接口地址

https://domain/api/cooperator/rechargeListAndInvoiceMsg

2.13.3 请求参数

参数名称	参数类型	参数含义	必填	备注
cooperatorId	String	商户 id	Y	商户 id
pageNum	Integer	页码	N	默认 1 不传默认第一页
pageSize	Integer	每页条数	N	默认 10
startTime	String	开始时间	Y	
endTime	String	结束时间	Y	
sign	String	签名	Y	将非空参数按照参数名名称按照 ASCII 升序排列(a=1&b=2&c=3)然后使用签名工具签名生成,(该字段不参与签名)
timestamp	String	时间戳	Y	Unix 时间戳，（毫秒）,超过 10 分种后该链接无效，以服务器时间为准

2.13.4 返回参数

参数名称	参数类型	参数含义	备注
statusCode	String	状态码	
statusText	String	状态描述	
data	String	业务描述	AES 加密后的字符串,密钥 key 由慧用工提供，解密后详细见下表

返回 data 业务参数明细

参数名称	参数类型	参数含义	备注
pageNum	Interger	页码	原样返回
pageSize	Interger	每页条数	原样返回
totalPageCount	Interger	总页数	
totalCount	Boolean	总条数	
list	List	充值记录列表	具体参数见 list 参数明细
invoiceList	List	发票类目列表	具体参数见 invoiceList 参数明细

list 参数明细

参数名称	参数类型	参数含义	备注
rechargeId	String	充值记录 id	
rechargeStatus	Interger	充值记录状态	充值状态：10、加款中（收到请求）；20、加款成功；30、挂账（待操作）；40、退款成功；
rechargeAmount	BigDecimal	交易金额	单位为分
paymentTime	String	交易时间	
invoiceStatus	Interger	开票状态	开票状态（10.未开票，20.开票处理中，30.已开票，40.驳回）

invoiceList 参数明细

参数名称	参数类型	参数含义	备注
invoiceVariety	String	开票类目	
invoiceContent	String	开票内容	逗号拼接 (开票类目可能对应多个开票内容,开票时切割一下,和对应的开票类目按对应参数传递)

2.14 申请开票

2.14.1 接口说明

申请开票

2.14.2 接口地址

https://domain/api/cooperator/approvalInvoice

2.14.3 请求参数

参数名称	参数类型	参数含义	必填	备注
cooperatorId	String	商户 id	Y	商户 id
rechargReecordIds	list	充值记录 id 列表	Y	充值记录 id
invoiceKind	Integer	发票类型	Y	10 普通发票,20 专用发票
invoiceMoney	BigDecimal	开票金额	Y	
invoiceVariety	String	开票类目	Y	
invoiceContent	String	开票内容	Y	
recipientsName	String	收件人	Y	
phoneNo	String	收件人手机号	Y	
recipientsLocation	String	收件人地址	Y	
timestamp	String	时间戳	Y	Unix 时间戳，（毫秒）,超过 10 分

				种后该链接无效，以服务器时间为准
sign	String	签名	Y	将非空参数按照参数名名称按照 ASCII 升序排列(a=1&b=2&c=3)然后使用签名工具签名生成,(该字段不参与签名)

2.14.4 返回参数

参数名称	参数类型	参数含义	备注
statusCode	String	状态码	
statusText	String	状态描述	
data	String	业务描述	无

2.15 开票结果通知

2.15.1 接口说明

系统审批成功后 会通知开票结果

2.15.2 接口地址

无

2.15.3 请求参数

参数名称	参数类型	参数含义	备注
businessBody	String	将业务参数通过 AES 加密后生成	加密方式为 AES，密钥 key 由慧用工提供，业务参数应该是 json 格式的字符串，以下的每个方法里的参数都为业务参数。
cooperatorId	String	商户 id	

明细参数

参数名称	参数类型	参数含义	必填	备注
cooperatorId	String	商户 id	Y	商户 id
cooperatorName	String	商户名称	Y	
invoiceApprovalId	String	发票审批流水号	Y	
invoiceAmount	BigDecimal	开票金额	Y	单位: 分
rechargeIds	List<String>	充值 id 记录列表	Y	
invoicePhoto	String	发票照片	Y	可能有多个
invoiceApprovalState	Interger	发票审批状态	Y	开票状态（10.未开票，20.开票处理中，30.已开票，40.驳回)

回调参数示例


```
{
        "cooperatorId": "C62107178963567823O8",
        "cooperatorName": "孙先生",
        "invoiceAmount": 2000000,
        "invoiceApprovalId": "IO7178270944778158O8",
        "invoiceApprovalState": 30,
        "invoicePhoto":
"[\"https://static.dingtalk.com/media/lADPD3zUKhMbQ6nNAqPNBQA_1280_675.jpg\"]",
        "rechargeIds": [
                "7076177972612464O4"
        ]
}
```


2.15.4 返回参数

参数名称	参数类型	参数含义	备注
statusCode	String	返回状态码	000000 表示成功;其他状态码表示失败,不需要加密;

			返回失败或者无返回的情况下会每隔 5 分钟回调一次,一共 10 次;
statusText	String	返回状态描述	

2.16 商户完税证明下载

2.16.1 接口说明

提供商户的完税证明下载，下载结果为 zip 格式的压缩包,下载频率每小时 6 次。

2.16.2 接口地址

https://domain/api/cooperator/tax/prove/download

2.16.3 请求参数

参数名称	参数类型	参数含义	必填	备注
cooperatorId	String	商户 id	Y	商户 id
overTaxTime	String	商户名称	Y	完税时间 格式 yyyy-MM
timestamp	String	时间戳	Y	Unix 时间戳，（毫秒）,超过 10 分种后该链接无效，以服务器时间为准
sign	String	签名	Y	将非空参数按照参数名名称按照 ASCII 升序排列(a=1&b=2&c=3)然后使用签名工具签名生成,(该字段不参与签名)

2.16.4 返回参数

该接口是下载文档接口，所以返回的是字节数组,客户端在接收到字节数组后可以直接下载为 zip 格式的文件。

伪代码：

```
// 请求并返回字节格式的数组
public static byte[] postJsonOfByte(String url, String json) throws IOException {
        RequestBody body = RequestBody.create(MediaType.parse("application/json; charset=utf-8"), json);
        Request request = new Request.Builder().url(url).post(body).build();
        Response response = execute(request);
        if (response.isSuccessful()) {
            return response.body().bytes();
        } else {
            throw new IOException("Unexpected code " + response);
        }
    }
    //奖获取的字节数组下载到本地
Files.write(Paths.get("/Users/xxx/Desktop/完税证明.zip"),
                bytes, StandardOpenOption.CREATE);
```

说明：如果下载的 zip 格式文件错误,则是参数传递错误导致的,所以请务必检查擦数的正确性。

2.17 自由职业者签约发送验证码

2.17.1 接口说明

自由职业者在上传完证件照之后需要获取短信验证码来生成电子签。获取的验证码在接口 2.18 使用。

2.17.2 接口地址

https://domain/api/worker/sign/send/smsCode

2.17.3 请求参数

参数名称	参数类型	参数含义	必填	备注
cooperatorId	String	商户 id	Y	商户 id
timestamp	String	时间戳	Y	Unix 时间戳，（毫秒）,超过 10 分种后该链接无效，以服务器时间为准
workerId	String	职业者 ID	Y	调用添加职业者之后返回的 workerId

workerMobile	String	职业者手机号	N	职业者手机号,用于获取短信验证码，如果没有传递，则默认使用添加职业者的时候的手机号。如果传递了，则用该手机号。（说明：必须填写真实手机号，否则通不过三要素校验）
sign	String	签名	Y	将非空参数按照参数名名称按照ASCII 升序排列(a=1&b=2&c=3)然后使用签名工具签名生成,(该字段不参与签名)

2.17.4 返回参数

参数名称	参数类型	参数含义	备注
statusCode	String	状态码	返回 000000 代表发送成功
statusText	String	状态描述	

2.18 自由职业者生成电子签

2.18.1 接口说明

接口 2.17 获取验证码之后在这个接口输入用于生成电子签。该接口调用成功之后才算签成功。

2.18.2 接口地址

https://domain/api/worker/sign/createContract

2.18.3 请求参数

参数名称	参数类型	参数含义	必填	备注
cooperatorId	String	商户 id	Y	商户 id
timestamp	String	时间戳	Y	Unix 时间戳，（毫秒）,超过 10 分种后该链接无效，以服务器时间为准
workerId	String	职业者 ID	Y	调用添加职业者之后返回的 workerId
smsCode	String	短信验证码	Y	调用 2.17 接口获取的验证码
sign	String	签名	Y	将非空参数按照参数名名称按照 ASCII 升序排列(a=1&b=2&c=3)然后使用签名工具签名生成,(该字段不参与签名)

2.18.4 返回参数

参数名称	参数类型	参数含义	备注
statusCode	String	状态码	
statusText	String	状态描述	
data	String	业务描述	AES 加密后的字符串,密钥 key 由慧用工提供，解密后详细见下表

返回业务参数明细

参数名称	参数类型	参数含义	备注
workerId	String	职业者 id	添加成功后返回的,用于后续查询
agreeState	Interge	签约状态	2 签约成功
agreeDesc	String	状态描述	状态描述

附录

3.0 返回码列表

状态码	状态描述	备注
000000	success	代表请求和响应成功 可以进一步解析响应 data 该状态码并不能代表具体业务的成功，具体业务的状态需要看响应中的业务参数
100000	业务操作异常，具体问题具体响应	提供合作商户进一步操作的依据
200000	订单已存在,请勿多次下发	针对同一个 requestNo 重复下单的情况
200101	该人员没有签约	收款人员没有签约时返回
200201	三要素验证失败	收款信息三要素验证不通过 三要素包括:姓名,身份号,收款账号
200301	年龄超限	收款人员年龄超限
200401	月下发金额超限	收款人员单月下发金额超过最大限制
200402	已达到年下发最大金额	收款人员年下发金额超过最大限制
200403	您的商户配置不支持支付宝账号校验,请联系管理人员!	
200404	支付宝账号和人名一致性校验失败,请联系管理人员!	
200405	当前存在未支付批次	Web 端存在未支付批次

200406	职业者不存在	签约获取验证码和生成电子签未查询到职业者返回
200407	当前商户和职业者所属商户不一致	签约获取验证码和生成电子签传递了其他商户的 workerId 返回
200408	该职业者已经签约成功,无需重复签约	签约获取验证码时该职业者已经签约成功则返回该提示
200409	请检查手机号、姓名、证件号是否一致	签约获取验证码传递的手机号不是当前职业者的时候返回该提示
200501	请先上传证件照,并确保识别成功	签约获取验证码和生成电子签约发现该职业者还未上传证件照时返回该提示
200502	手机号不能为空	签约获取验证码时没有传递手机号并且系统内也没有预留手机号时返回该提示
200503	还未获取验证码	签约生成电子签时未调用过获取验证码接口
200504	验证码不正确	签约生成电子签验证码输入错误时返回该提示
400101	无权访问	未开通开发者模式或商户不存在
400201	参数问题	请求参数不正确
400301	异常请求	比如下发公司配置不存在，联系慧用工人员
400501	验签失败	签名错误或参数被篡改
400601	时间戳异常	服务器时间相差较大
400701	请先配置白名单	未配置白名单提示
500000	服务器异常，请稍后重试	联系慧用工人员查找具体原因

3.1 AES 加解密工具类（JAVA）

```java
package com.hvyogo.api.utils;

import org.springframework.util.Base64Utils;

import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.io.IOException;
import java.security.Key;
import java.security.NoSuchAlgorithmException;

public class AESUtils {
    private static String transformation = "AES/ECB/PKCS5Padding";
    private static String algorithm = "AES";
    /**
     * 解密
     * @param content
     * @param key
     * @return
     * @throws Exception
     */
    public static String decryptByHex(String content, String key) throws Exception {
        return decrypt(hex2Byte(content), key);
```

```
    }

    /**
     * 解密过程
     * @param encryptBytes
     * @param key
     * @return
     * @throws Exception
     */
    private static String decrypt(byte[] encryptBytes, String key) throws Exception {
        Key k = toKey(Base64Utils.decodeFromString(key));
        Cipher cipher = Cipher.getInstance(transformation);
        cipher.init(2, k);
        return new String(cipher.doFinal(encryptBytes));
    }




    /**
     * 加密，结果为 16 进制
     *
     * @param content
     * @param key
     * @return
     * @throws Exception
     */
    public static String encrypt2Hex(String content, String key) throws Exception {
        return byte2Hex(encrypt(content, key));
    }

    /**
     * 加密过程
     *
     * @param content
     * @param key
     * @return
     * @throws Exception
     */

    private static byte[] encrypt(String content, String key) throws Exception {

        Key k = toKey(Base64Utils.decodeFromString(key));
        Cipher cipher = Cipher.getInstance(transformation);

        cipher.init(Cipher.ENCRYPT_MODE, k);

        return cipher.doFinal(content.getBytes());

    }

    /**
     * 2 进制转 16 进制
     *
     * @param buff
     * @return
     */
    public static String byte2Hex(byte buff[]) {
        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < buff.length; i++) {
            String hex = Integer.toHexString(buff[i] & 0xFF);
            if (hex.length() == 1) {
                hex = '0' + hex;
            }
            sb.append(hex.toUpperCase());
        }
        return sb.toString();
    }
```

```
    /**
     * 16 进制转 2 进制
     * @param hex
     * @return
     */
    public static byte[] hex2Byte(String hex) {
        if (hex != null && hex.length() >= 1 && hex.length() % 2 == 0) {
            byte[] result = new byte[hex.length() / 2];

            for (int i = 0; i < result.length; ++i) {
                int high = Integer.parseInt(hex.substring(i * 2, i * 2 + 1), 16);
                int low = Integer.parseInt(hex.substring(i * 2 + 1, i * 2 + 2), 16);
                result[i] = (byte) (high << 4 | low);
            }

            return result;
        } else {
            return null;
        }
    }



    public static Key toKey(byte[] key) throws Exception {
        return new SecretKeySpec(key, algorithm);
    }

    /**
     * 密钥生成
     * @return
     * @throws NoSuchAlgorithmException
     * @throws IOException
     */
    public static String getAutoCreateAESKey() throws NoSuchAlgorithmException,
IOException {
        KeyGenerator kg = KeyGenerator.getInstance("AES");
        //要生成多少位，只需要修改这里即可 128, 192 或 256
        kg.init(128);
        SecretKey sk = kg.generateKey();
        byte[] b = sk.getEncoded();
        return Base64Utils.encodeToString(b);
    }

    public static void main(String[] args) throws IOException,
NoSuchAlgorithmException {
        String autoCreateAESKey = getAutoCreateAESKey();
        System.out.println(autoCreateAESKey);
    }

}
```

3.2 RSA 签名工具类（JAVA）

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by Fernflower decompiler)
//

package com.hvyogo.api.utils;

import org.apache.commons.codec.binary.Base64;
```

```java
import org.springframework.util.StringUtils;

import javax.crypto.Cipher;
import java.io.ByteArrayOutputStream;
import java.security.*;
import java.security.interfaces.RSAPrivateKey;
import java.security.interfaces.RSAPublicKey;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import java.util.*;


public class RSAUtils {
    public static final String KEY_ALGORITHM = "RSA";
    public static final String SIGNATURE_ALGORITHM = "SHA1withRSA";
    private static final String PUBLIC_KEY = "RSAPublicKey";
    private static final String PRIVATE_KEY = "RSAPrivateKey";
    private static final int MAX_ENCRYPT_BLOCK = 117;
    private static final int MAX_DECRYPT_BLOCK = 128;

    public RSAUtils() {
    }

    public static Map<String, Key> genKeyPair() throws Exception {
        KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance(KEY_ALGORITHM);
        keyPairGen.initialize(1024);
        KeyPair keyPair = keyPairGen.generateKeyPair();
        RSAPublicKey publicKey = (RSAPublicKey)keyPair.getPublic();
        RSAPrivateKey privateKey = (RSAPrivateKey)keyPair.getPrivate();
        Map<String, Key> keyMap = new HashMap(2);
        keyMap.put(PUBLIC_KEY, publicKey);
        keyMap.put(PRIVATE_KEY, privateKey);
        return keyMap;
    }

    /**
     * 数字签名
     * @param data
     * @param privateKey
     * @return
     * @throws Exception
     */
    public static String sign(byte[] data, String privateKey) throws Exception {
        byte[] keyBytes = Base64.decodeBase64(privateKey);
        PKCS8EncodedKeySpec pkcs8KeySpec = new PKCS8EncodedKeySpec(keyBytes);
        KeyFactory keyFactory = KeyFactory.getInstance(KEY_ALGORITHM);
        PrivateKey privateK = keyFactory.generatePrivate(pkcs8KeySpec);
        Signature signature = Signature.getInstance(SIGNATURE_ALGORITHM);
        signature.initSign(privateK);
        signature.update(data);
        return new String(Base64.encodeBase64(signature.sign()));
    }

    /**
     * 验证签名
     * @param data
     * @param publicKey
     * @param sign
     * @return
     * @throws Exception
     */
    public static boolean verify(byte[] data, String publicKey, String sign) throws
Exception {
        byte[] keyBytes = Base64.decodeBase64(publicKey);
        X509EncodedKeySpec keySpec = new X509EncodedKeySpec(keyBytes);
        KeyFactory keyFactory = KeyFactory.getInstance(KEY_ALGORITHM);
        PublicKey publicK = keyFactory.generatePublic(keySpec);
        Signature signature = Signature.getInstance(SIGNATURE_ALGORITHM);
        signature.initVerify(publicK);
```

```
        signature.update(data);
        return signature.verify(Base64.decodeBase64(sign));
    }

    /**
     * 私钥解密
     * @param encryptedData
     * @param privateKey
     * @return
     * @throws Exception
     */
    public static byte[] decryptByPrivateKey(byte[] encryptedData, String privateKey)
throws Exception {
        byte[] keyBytes = Base64.decodeBase64(privateKey);
        PKCS8EncodedKeySpec pkcs8KeySpec = new PKCS8EncodedKeySpec(keyBytes);
        KeyFactory keyFactory = KeyFactory.getInstance(KEY_ALGORITHM);
        Key privateK = keyFactory.generatePrivate(pkcs8KeySpec);
        Cipher cipher = Cipher.getInstance(keyFactory.getAlgorithm());
        cipher.init(2, privateK);
        int inputLen = encryptedData.length;
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        int offSet = 0;

        for(int i = 0; inputLen - offSet > 0; offSet = i * MAX_DECRYPT_BLOCK) {
            byte[] cache;
            if (inputLen - offSet > MAX_DECRYPT_BLOCK) {
                cache = cipher.doFinal(encryptedData, offSet, MAX_DECRYPT_BLOCK);
            } else {
                cache = cipher.doFinal(encryptedData, offSet, inputLen - offSet);
            }

            out.write(cache, 0, cache.length);
            ++i;
        }

        byte[] decryptedData = out.toByteArray();
        out.close();
        return decryptedData;
    }

    /**
     * 公钥解密
     * @param encryptedData
     * @param publicKey
     * @return
     * @throws Exception
     */
    public static byte[] decryptByPublicKey(byte[] encryptedData, String publicKey)
throws Exception {
        byte[] keyBytes = Base64.decodeBase64(publicKey);
        X509EncodedKeySpec x509KeySpec = new X509EncodedKeySpec(keyBytes);
        KeyFactory keyFactory = KeyFactory.getInstance(KEY_ALGORITHM);
        Key publicK = keyFactory.generatePublic(x509KeySpec);
        Cipher cipher = Cipher.getInstance(keyFactory.getAlgorithm());
        cipher.init(2, publicK);
        int inputLen = encryptedData.length;
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        int offSet = 0;

        for(int i = 0; inputLen - offSet > 0; offSet = i * MAX_DECRYPT_BLOCK) {
            byte[] cache;
            if (inputLen - offSet > MAX_DECRYPT_BLOCK) {
                cache = cipher.doFinal(encryptedData, offSet, MAX_DECRYPT_BLOCK);
            } else {
                cache = cipher.doFinal(encryptedData, offSet, inputLen - offSet);
            }

            out.write(cache, 0, cache.length);
            ++i;
```

```
        }

        byte[] decryptedData = out.toByteArray();
        out.close();
        return decryptedData;
    }

    /**
     * 公钥加密
     * @param data
     * @param publicKey
     * @return
     * @throws Exception
     */
    public static byte[] encryptByPublicKey(byte[] data, String publicKey) throws
Exception {
        byte[] keyBytes = Base64.decodeBase64(publicKey);
        X509EncodedKeySpec x509KeySpec = new X509EncodedKeySpec(keyBytes);
        KeyFactory keyFactory = KeyFactory.getInstance(KEY_ALGORITHM);
        Key publicK = keyFactory.generatePublic(x509KeySpec);
        Cipher cipher = Cipher.getInstance(keyFactory.getAlgorithm());
        cipher.init(1, publicK);
        int inputLen = data.length;
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        int offSet = 0;

        for(int i = 0; inputLen - offSet > 0; offSet = i * MAX_ENCRYPT_BLOCK) {
            byte[] cache;
            if (inputLen - offSet > MAX_ENCRYPT_BLOCK) {
                cache = cipher.doFinal(data, offSet, MAX_ENCRYPT_BLOCK);
            } else {
                cache = cipher.doFinal(data, offSet, inputLen - offSet);
            }

            out.write(cache, 0, cache.length);
            ++i;
        }

        byte[] encryptedData = out.toByteArray();
        out.close();
        return encryptedData;
    }

    /**
     * 私钥加密
     * @param data
     * @param privateKey
     * @return
     * @throws Exception
     */
    public static byte[] encryptByPrivateKey(byte[] data, String privateKey) throws
Exception {
        byte[] keyBytes = Base64.decodeBase64(privateKey);
        PKCS8EncodedKeySpec pkcs8KeySpec = new PKCS8EncodedKeySpec(keyBytes);
        KeyFactory keyFactory = KeyFactory.getInstance(KEY_ALGORITHM);
        Key privateK = keyFactory.generatePrivate(pkcs8KeySpec);
        Cipher cipher = Cipher.getInstance(keyFactory.getAlgorithm());
        cipher.init(1, privateK);
        int inputLen = data.length;
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        int offSet = 0;

        for(int i = 0; inputLen - offSet > 0; offSet = i * MAX_ENCRYPT_BLOCK) {
            byte[] cache;
            if (inputLen - offSet > MAX_ENCRYPT_BLOCK) {
                cache = cipher.doFinal(data, offSet, MAX_ENCRYPT_BLOCK);
            } else {
                cache = cipher.doFinal(data, offSet, inputLen - offSet);
            }
```

```
            out.write(cache, 0, cache.length);
            ++i;
        }

        byte[] encryptedData = out.toByteArray();
        out.close();
        return encryptedData;
    }

    /**
     * 获取私钥
     * @param keyMap
     * @return
     * @throws Exception
     */
    public static String getPrivateKey(Map<String, Key> keyMap) throws Exception {
        Key key = keyMap.get(PRIVATE_KEY);
        return new String(Base64.encodeBase64(key.getEncoded()));
    }

    /**
     * 获取公钥
     * @param keyMap
     * @return
     * @throws Exception
     */
    public static String getPublicKey(Map<String, Key> keyMap) throws Exception {
        Key key = keyMap.get(PUBLIC_KEY);
        return new String(Base64.encodeBase64(key.getEncoded()));
    }


    /**
     * 将参数名称按照 ASCII 排序
     * @param paramMap
     * @return
     */
    public static String sortParam(Map<String, Object> paramMap) {
        List<Map.Entry<String, Object>> list = new ArrayList<Map.Entry<String,
Object>>(paramMap.entrySet());
        Collections.sort(list, new Comparator<Map.Entry<String, Object>>() {
            @Override
            public int compare(Map.Entry<String, Object> o1, Map.Entry<String, Object>
o2) {
                return (o1.getKey().compareTo(o2.getKey()));
            }
        });
        StringBuilder sb = new StringBuilder();
        for (Map.Entry<String, Object> item : list) {
            if (!StringUtils.isEmpty(item.getKey())
&& !StringUtils.isEmpty(item.getValue())) {

sb.append(item.getKey()).append("=").append(item.getValue()).append("&");
            }
        }
        return sb.substring(0, sb.length() - 1);
    }

    public static void main(String[] args) throws Exception {
        Map<String, Key> map = genKeyPair();
        String privateKey = getPrivateKey(map);
        String publicKey = getPublicKey(map);
        System.out.println("pub=" + publicKey);
        System.out.println("pri="+ privateKey);
    }

}
```

3.3 备注字段允许的

服务费
技术开发服务费
合同能源管理服务费
专业技术服务费
城市规划服务费
软件服务费
电路设计服务费
电路测试服务费
相关电路技术支持服务费
信息系统服务费
信息系统增值服务费
业务流程管理服务费
设计优化咨询服务费
设计制作服务费
专业设计服务费
文印晒图服务费
知识产权服务费
会议服务费
展览服务费
物流辅助服务费
会计咨询服务费
技术咨询服务费
其他咨询服务费
其他涉税服务费
一般税务咨询费
企业管理服务费
经纪代理服务费
商务辅助服务费
金融代理服务费
代理报关服务费
代理记账服务费
纳税申报代理服务费
其他税务事项代理服务费
推广服务费
音视频服务费
（市场、网络等）渠道搭建及使用服务费
其他现代服务费
文化服务费

体育服务费
体育比赛服务费
体育表演服务费
体育活动服务费
体育训练服务费
体育指导服务费
体育管理服务费
教育辅助服务费
教育测评服务费
考试服务费
招生服务费
教育信息咨询服务费
市容市政管理服务费
家政服务（非员工制）费
保洁服务费
居民日常服务费
家居产品维保服务费
洗车服务费
外卖配送服务费
植物养护服务费
其他生活服务费