# 中路支付平台

# 接口报文规范 V4.0

| 版本 | 日期 | 作者 | 备注 |
| --- | --- | --- | --- |
| V1.0 | 2017-08-04 | 李志远 | 创建 |
| V1.1 | 2017-08-17 | 李志远 | 删除接口中的投保单号及保单号字段 |
| V1.2 | 2017-09-09 | 李志远 | 统一下单接口添加新的交易方式：微信号内支付<br>去除设备类型等字段的必传 |
| V1.3 | 2018-09-11 | 邢美燕 | 增加调用支付平台例文 |
| V1.4 | 2018-11-05 | 邢美燕 | 更新预下单接口地址和查询接口地址 |
| V2.0.0 | 2019-05-24 | 李志远 | 添加由支付平台生成保单功能 |
| V2.0.1 | 2019-06-25 | 张鑫鹏 | 删除接口中的交易方式、openid，修改订单状态 |
| V2.2 | 2019-06-28 | 张鑫鹏 | 修改支付链接字段 id |
| V3.0 | 2019-07-01 | 张鑫鹏 | 增加环境准备和错误编码章节 |
| V3.1 | 2019-07-01 | 张鑫鹏 | 修改接入地址、附录中参考章节 |
| V3.2 | 2019-07-03 | 张鑫鹏 | 统一下单接口支持微信/支付宝 pc 扫码支付 |
| V3.3 | 2019-07-08 | 张鑫鹏 | 3.1、3.3 增加内网地址，3.2 交易结果通知增加返回支付 |

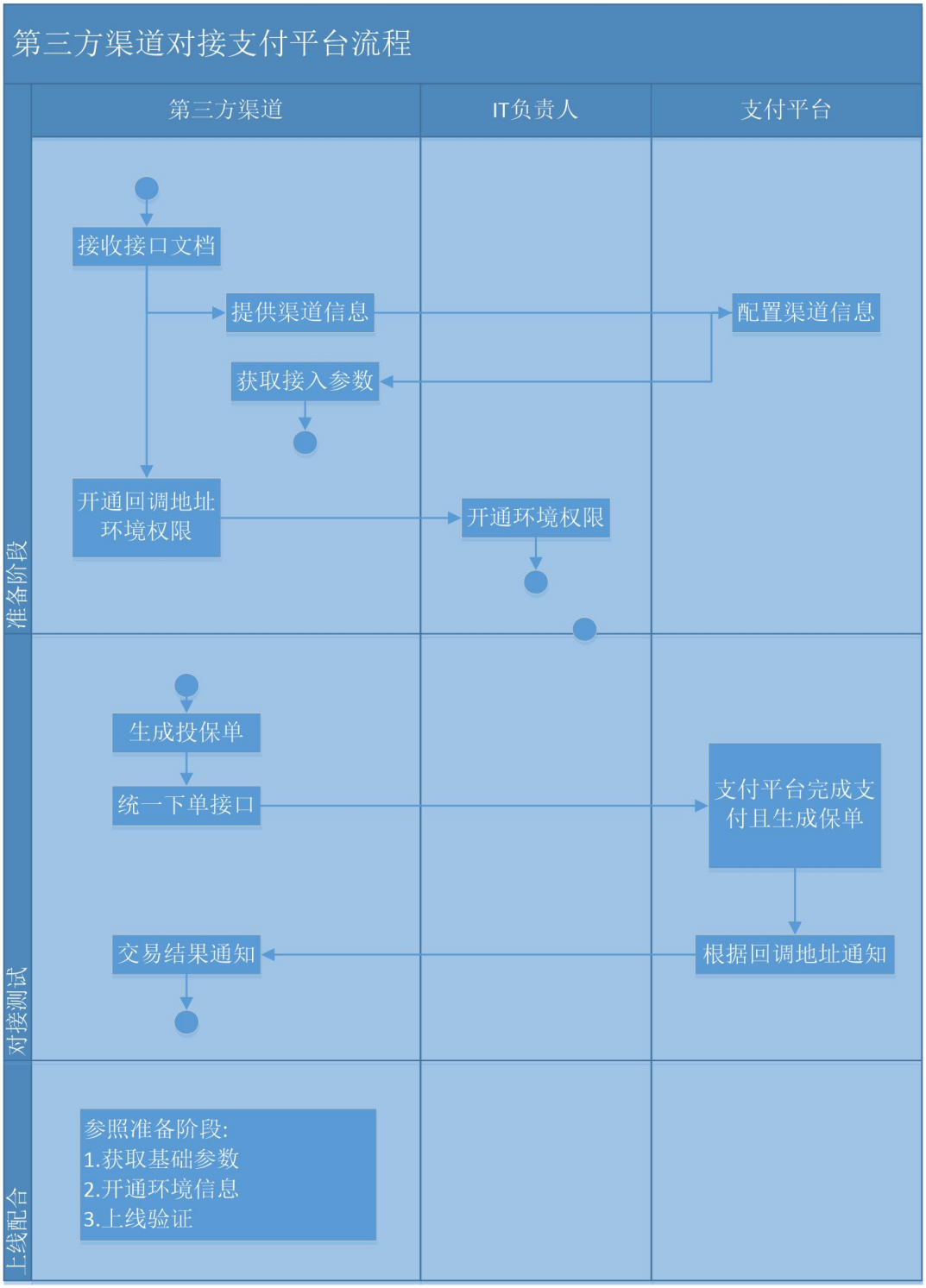
| | | | |
|---|---|---|---|
| | | | 方式 |
| V4.0 | 2019-07-08 | 王慎俊 | 3.4 增加缴费结果通知接口 |

# 目录

1 接入流程（重点）.....4
1.1 环境权限开通（IT 配合).....5
1.2 渠道接入基础信息配置.....5
2 接口介绍.....6
2.1 功能介绍.....6
2.2 接入方式.....6
2.3 接入地址.....7
2.4 支付方式介绍.....7
2.5 Sign 签名机制介绍.....7
3 接口定义.....8
3.1 统一下单接口.....8
3.2 交易结果通知.....11
3.3 交易查询.....12
3.4 缴费结果通知（内部系统关注）.....14
4 错误编码.....16
5 附录.....17

# 1 接入流程（重点）

渠道接入时，务必认真阅读此章节内容，此章节提到的内容，后续支持阶段不再赘述。



## 1.1 环境权限开通（IT 配合)

1．交易结果通知地址需要开通访问权限,反馈回调地址信息至联调支持群,模板如下：

回调地址：XXXXXXXX（渠道提供）

2．相关人员开通权限：

1） 测试环境：

a. 请开通 10.0.20.47 服务器访问交易结果通知地址权限（中路 IT 处理）

b. hosts 文件请配置交易结果通知地址域名映射（支付平台项目组配合）

2） 生产环境（上线时准备）

a. 请开通 10.0.10.112 服务器访问交易结果通知地址权限，并且 hosts 文件请配置交易结果通知地址域名映射（中路 IT 处理）

b. 上述信息升级清单中会说明（支付平台项目组配合）

## 1.2 渠道接入基础信息配置

1．对接渠道提供基础信息，支付平台才能提供对应的对接参数，对接渠道需要按照下述模板将信息反馈至链条支持群。

2． **（必须提供）**渠道方与业务配合提供以下基础信息：（渠道提供）

基础信息，*为必提供信息

| *业务来源 | *业务来源名称 | *渠道中级 | *渠道中级名称 | 业务来源子类(可配置多个) | 业务来源子类名称 | *是否对接通联 | *支付方式 |
|---|---|---|---|---|---|---|---|
| 07 | 网销渠道 | 0701 | 微信营销 | 070101 | 万事保 | 是 | JSAPI/H5 |

3. **（必须提供）**提供给渠道方的支付参数信息：（支付平台项目组配合）

支付配置列表查询信息与支付平台对接信息对应表

| 支付方式配置列表信息 | 对接支付平台信息 |
| --- | --- |
| APPID | app_id |
| 秘钥 | app_secret |

4. （**非必须环节，仅对接支付的渠道无需关注**）在支付平台进行整体的业务操作（包含出单等），需要进行缴费结果通知的内部系统，渠道方与业务配合提供以下基础信息：（渠道提供）

| ***业务来源** | ***业务来源名称** | ***渠道中级** | ***渠道中级名称** | **缴费结果通知地址** | **缴费规则** |
| --- | --- | --- | --- | --- | --- |
| 88 | 内部系统 | 88xx | 云架构 | xxxxxx | 如 Y193 打头等 |

# 2 接口介绍

## 2.1 功能介绍

支付平台提供渠道通过系统对接的方式发起移动支付交易。

## 2.2 接入方式

渠道申请开通支付权限后，支付平台分配给渠道方接入 appid，app_secret

接口通过 HTTP/HTTPS 方式提供，交易参数通过 POST 方式提交上来，返回结果均为 JSON 格式的数据报文。

## 2.3 接入地址

接入测试参数,请阅读 1.2 渠道接入基础信息配置获取以下实际参数值：

app_id：0000000000

app_secret：8JRrQId4p0FWqzjjOLxglzLXSQEB2m0Q

## 2.4 支付方式介绍

用户在下单后进入结算页面，渠道调取预下单接口获取支付页面地址，并调取手机自带浏览器打开支付页面地址，在支付页面完成支付。在完成支付后支付平台会通过 https 调用方式通知到渠道预设的回调地址。

## 2.5 Sign 签名机制介绍

这里将定义生成sign 字符串的方法。

a、 签名字符串为除了sign之外的其他字段，加上分配的app_secret的键值对.

b、 所有参与签名的字段，按字段名的ASCLL码从小到大排序后，使用URL的键值对的格式(即key1=value1&key2=value2)拼接成字符串string.(value为空字段不参与签名)

c、 如果存在List<String> a,则使用a0=1形式进行拼接

d、 如果存在List<Object>,则将List中每个Object以key0=value形式进行拼接

e、 sign=md5(string.getbyte("utf-8")).toUpperCase()

如:

请求报文为


```
{
 "app_id": "201906111545550782661",
 "total_fee": "1",
 "fee_type": "CNY",
 "trade_no": "20190610545454532",
 "nonce_str": "3800",
 "body": "中路保险网销订单:234455555",
```

```
"spbill_create_ip": "60.209.238.22",
"notify_url": "http://baidu.com",
"sign": "91AE622BDC2DF201914C4A62357D1D62",
"sign_type": "MD5",
"proposal_list": [
    {
        "proposal_no": "102370600542019000469",
        "money": "20000",
        "c_order_no": "201905290200192256",
    }
]
```


}则

待签名字符串为

signStr:
app_id=201906111545550782661&app_secret=abEvCKwLuvsnh00LsMxULPv8UfboBBjW&body=

中路保险网销订单:234455555&c_order_no0=201905290200192256& fee_type=CNY&

money0=20000&nonce_str=3800&notify_url=http://baidu.com&
proposal_no0=102370600542019000469&
sign_type=MD5&spbill_create_ip=60.209.238.22&total_fee=1&trade_no=20190610545454532

sign=md5(string.getBytes("utf-8") .toUpperCase()=" C9B88EA190D5006D88511A001AD52C70";

**注意:因为以后可能会增加参数,请获取全量变量,然后遍历获取非空变量,而不能用**

**request.getparam("key")这样用文档中的字段名来一个个获取,到时候增加通知字段,就无法兼容了**

# 3 接口定义

## 3.1 统一下单接口

接口地址

外网地址如下:

测试环境 https://testzfpt.zhlic.com.cn/payment/unifiedorder/v2.0.0
生产环境 https:// zfpt.zhlic.com.cn/payment/unifiedorder/v2.0.0

内网地址如下：

测试环境 http://10.0.101.64:8444/payment/unifiedorder/v2.0.0
生产环境 http://10.0.101.74:8444/payment/unifiedorder/v2.0.0

使用场景：

微信公众号及支付宝服务窗 JSAPI 支付、微信 H5 支付、微信/支付宝 PC 扫码支付。

请求参数：

| 参数 | 参数名称 | 取值 | 可空 | 最大长度 | 备注 |
|---|---|---|---|---|---|
| app_id | 应用 ID | 平台给此应用分配的 APPID | 否 | 10 | |
| total_fee | 交易金额 | 单位为分 | 否 | 15 | 测试环境也传真实金额 |
| fee_type | 货币类型 | 人民币传 CNY | 否 | 16 | 目前仅支持 CNY |
| trade_type | 支付类型 | 微信扫码：W01;支付宝扫码：A01 | 是 | 10 | 仅扫码支付时需要传该值 |
| trade_no | 商户交易单号 | 商户的交易订单号 | 否 | 32 | 保证商户平台唯一 |
| nonce_str | 随机字符串 | 商户自行生成的随机字符串 | 否 | 32 | 推荐使用随机数生成算法生成，不长于 32 位 |
| body | 订单标题 | 订单商品名称 | 否 | 100 | 最大 100 个字节(50 个中文字符) |
| attach | 附加数据 | 附加数据，在查询 API 和支付通知中原样返回，该字段主要用于商户携带订单的自定义数据 | 是 | 127 | 如无需要可不传 |
| remark | 备注 | 备注信息 | 是 | 50 | 最大 50 个字节(25 个中文字符) |
| validtime | 有效时间 | 订单有效时间，以分为单位，不填默认为 15 分钟 | 是 | 2 | 最大 60 分钟 |
| spbill_create_ip | 终端 ip | 发起调用 APP 的真实 IP 地址 | 否 | 16 | 必须传正确的客户端 IP |
| notify_url | 交易结果通知地址 | 接收微信支付异步通知回调地址 | 否 | 256 | 支付成功后会通知到这个接口，详情见 2.2 交易结果通知 |
| redirect_url | 支付成功跳转地址 | 支付成功后会跳转到这个地址 | 是 | 256 | 如果传空,则支付成功后会停留在微信页面，必须靠手动切换回 app。可以使用 Android / iOS 中的中 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 的 scheme 技术来从网页跳转回 APP |
| limit_pay | 支付限制 | no_credit--指定不能使用信用卡支付 | 是 | 32 | 暂时只对微信支付有效,仅支持 no_credit |
| sign | 签名 | 详见 2.5 | 否 | 32 | |
| sign_type | 签名类型 | 默认传 MD5 | 否 | 32 | 目前仅支持 MD5 |
| device_type | 设备类型 | Android 传 Android,iOS 传 iOS,H5 传 Wap | 是 | 10 | 仅当交易方式为 MWEB 时候必传 |
| app_name | 应用名 | Android / iOS 应用传应用名,H5 传网站名 | 是 | 40 | 仅当交易方式为 MWEB 时候必传 |
| app_bundle | 应用 ID | iOS 应用传 bundle_id,Android 传 package_name,H5 传网站 URL | 是 | 180 | 仅当交易方式为 MWEB 时候必传 |
| proposal_list该支付订单的投保单列表list<object> | | | | | |
| proposal_no | 投保单号 | | 否 | 100 | |
| c_order_no | 订单号 | | 否 | | 投保单生成时的订单号 |
| money | 该投保单总保费金额 | | 否 | 32 | 以分为格式<br>金额分项和应和交易金额一致 |

响应参数:

| 字段 ID | 字段名称 | 取值 | 可空 | 最大长度 | 备注 |
|---|---|---|---|---|---|
| return_code | 返回码 | SUCCESS/FAIL | 否 | 8 | SUCCESS/FAIL<br>此字段是通信标识，非交易标识，交易是否成功需要查看result_code来判断 |
| return_msg | 返回码说明 | | 是 | 128 | 返回信息，如非空，为错误原因<br>如验签失败 |
| 以下字段只有当return_code为SUCCESS时有返回 | | | | | |
| app_id | 应用 ID | 平台给此应用分配的 APPID | 否 | 10 | |
| nonce_str | 随机字符串 | 商户自行生成的随机字符串 | 否 | 32 | 和请求报文中的 nonce_str 相同 |
| sign | 签名 | 详见 2.5 | 否 | 32 | 微信返回的 sign,需要对此进行验签 |
| result_code | 业务结果 | SUCCESS/FAIL | 否 | 16 | |
| err_code | 错误代码 | | 是 | 32 | |
| err_code_des | 错误代码描述 | | 是 | 128 | |

| 以下字段在return_code 和result_code都为SUCCESS的时候有返回 | | | | | |
|---|---|---|---|---|---|
| trxid | 交易单号 | 支付平台的内部交易流水号 | 否 | 20 | |
| jump_url | 支付链接 | 支付类型非扫码支付（W01/A01）时返回该值 | 是 | 32 | 跳转此地址进行支付 |
| code_url | 二维码链接 | 支付类型为扫码支付（W01/A01）时返回该值 | 是 | 32 | 用该链接生成二维码进行扫码支付 |

## 3.2 交易结果通知

接口地址：统一下单接口的 notify_url

使用场景：

在支付成功后支付平台会调用此地址进行通知

商户服务端在接收到结果通知并正确处理后，应响应消息。正确处理响应 success，其他响应或者不响应支付平台则通过重发机制进行重发，重发次数为 3 次。

请求参数：

| 字段 ID | 字段名称 | 取值 | 可空 | 最大长度 | 备注 |
|---|---|---|---|---|---|
| app_id | 应用 ID | 平台给此应用分配的 APPID | 否 | 10 | |
| nonce_str | 随机字符串 | 支付平台生成的随机字符串 | 否 | 32 | |
| sign | 签名 | 详见 2.5 | 否 | 32 | |
| result_code | 业务结果 | SUCCESS/FAIL | 否 | 16 | 只会在成功时候才会通知，即此字段默认为 SUCCESS |
| total_fee | 交易金额 | 单位为分 | 否 | 15 | |
| fee_type | 货币类型 | 人民币传 CNY | 否 | 16 | 目前仅支持 CNY |
| open_id | 支付用户标识信息 | 例如用户在微信公众号，支付宝中的唯一标识 id | 是 | 32 | |
| trx_id | 交易单号 | 支付平台的内部交易流水号 | 否 | 20 | |
| chnl_trxid | 渠道平台交易单号 | 例如微信,支付宝平台的交易单号 | 是 | 50 | |
| proxy_trxid | 通联等第三方交易号 | 如走通联等第三方支付渠道,此字段传第三方支付渠道的交易号 | 是 | 50 | |

| | | | | | |
|---|---|---|---|---|---|
| trade_no | 商户交易单号 | 商户的交易订单号 | 否 | 32 | |
| order_status | 订单状态 | 0-未和微信交互<br>1-和微信交互生成预支付单号<br>2-已支付<br>4-已支付并成功生成保单<br>-4-已支付但保单生成失败 | 否 | 10 | |
| trade_type | 支付方式 | 0-微信<br>1-通联微信<br>2-支付宝<br>3-通联支付宝 | 是 | 10 | 本次支付使用的支付方式 |
| proposal_list list<object> | | | | | |
| proposal_no | 投保单号 | | 否 | 100 | |
| policy_no | 保单号 | | 是 | 100 | order_status 为 4 时返回 |
| c_dptacc_code | 做账机构 | | 否 | 200 | |
| 以下字段在order_status为-4时候有返回 | | | | | |
| fail_reason | 保单落地失败原因 | | 否 | 1000 | |

响应参数：

| 字段 ID | 字段名称 | 取值 | 可空 | 最大长度 | 备注 |
|---|---|---|---|---|---|
| return_code | 返回码 | SUCCESS/FAIL | 否 | 8 | SUCCESS/FAIL<br>SUCCESS表示商户接收通知成功并校验成功 |
| return_msg | 返回码说明 | | 是 | 100 | 返回信息，如非空，为错误原因：<br>签名失败<br>参数格式校验错误 |

## 3.3 交易查询

接口地址

外网地址如下：

测试环境 https://testzfpt.zhlic.com.cn/payment/pay/query

生产环境 https://zfpt.zhlic.com.cn/payment/pay/query

内网地址如下：

测试环境 http://10.0.101.64:8444/payment/pay/query
生产环境 http://10.0.101.74:8444/payment/pay/query

使用场景：

支持对支付结果的查询

请求参数：

| 参数 | 参数名称 | 取值 | 可空 | 最大长度 | 备注 |
|---|---|---|---|---|---|
| app_id | 应用 ID | 平台给此应用分配的 APPID | 否 | 10 | |
| trx_id | 交易单号 | 支付平台的内部交易流水号 | 否 | 20 | |
| nonce_str | 随机字符串 | 商户自行生成的随机字符串 | 否 | 32 | 推荐使用随机数生成算法生成，不长于 32 位 |
| sign | 签名 | | 否 | 32 | 详见 2.5 |

响应参数：

| 字段 ID | 字段名称 | 取值 | 可空 | 最大长度 | 备注 |
|---|---|---|---|---|---|
| return_code | 返回码 | SUCCESS/FAIL | 否 | 8 | SUCCESS/FAIL<br>此字段是通信标识，非交易标识，交易是否成功需要查看 result_code来判断 |
| return_msg | 返回码说明 | | 是 | 128 | 返回信息，如非空，为错误原因<br>如验签失败 |
| 以下信息只有当return_code为SUCCESS时有返回 | | | | | |
| app_id | 应用 ID | 平台给此应用分配的 APPID | 否 | 10 | |
| nonce_str | 随机字符串 | 同请求的随机字符串 | 否 | 32 | |
| sign | 签名 | 详见 2.5 | 否 | 32 | |
| result_code | 业务结果 | SUCCESS/FAIL | 否 | 16 | |
| err_code | 错误代码 | | 是 | 32 | |
| err_code_des | 错误代码描述 | | 是 | 128 | |
| 以下内容当业务结果 result_code 位 SUCCESS 时候会返回 | | | | | |
| total_fee | 交易金额 | 单位为分 | 否 | 15 | |
| fee_type | 货币类型 | 人民币传 CNY | 否 | 16 | 目前仅支持 CNY |

| | | | | | |
|---|---|---|---|---|---|
| trx_id | 交易单号 | 支付平台的内部交易流水号 | 否 | 20 | |
| chnl_trxid | 渠道平台交易单号 | 例如微信,支付宝平台的交易单号 | 是 | 50 | |
| trade_no | 商户交易单号 | 商户的交易订单号 | 否 | 32 | |
| order_status | 订单状态 | 0-未和微信交互<br>1-和微信交互生成预支付单号<br>2-已支付<br>4-已支付并成功生成保单<br>-4-已支付但保单生成失败<br>3-已支付成功生成保单且完成支付通知 | 否 | 10 | |
| proposal_list list<object> | | | | | |
| proposal_no | 投保单号 | | 否 | 100 | |
| policy_no | 保单号 | | 是 | 100 | order_status 为 3 或 4 时返回 |
| c_dptacc_code | 做账机构 | | 否 | 200 | |
| 以下字段在order_status为-4时候有返回 | | | | | |
| fail_reason | 保单落地失败原因 | | 否 | 1000 | |

## 3.4 缴费结果通知（内部系统关注）

使用场景：

线上支付平台（含小程序、支付宝、公众号），客户通过缴费号/车牌号等信息，进行投保信息确认同时进行线上支付；支付成功之后，如果对应业务符合后台配置规则，则支付平台会将保单落地信息推送至对应的渠道。

渠道系统在接收到结果通知并正确处理后，应响应消息。正确处理响应 success，，其他响应或者不响应支付平台则通过重发机制进行重发，重发次数为 3 次。

请求参数：

| 字段 ID | 字段名称 | 取值 | 可空 | 最大长度 | 备注 |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| app_id | 应用 ID | 平台给此应用分配的 APPID | 否 | 10 | |
| nonce_str | 随机字符串 | 支付平台生成的随机字符串 | 否 | 32 | |
| sign | 签名 | 详见 2.5 | 否 | 32 | |
| result_code | 业务结果 | SUCCESS/FAIL | 否 | 16 | 只会在成功时候才会通知，即此字段默认为 SUCCESS |
| open_id | 支付用户标识信息 | 例如用户在微信公众号，支付宝中的唯一标识 id | 是 | 32 | |
| total_fee | 交易金额 | 单位为分 | 否 | 15 | |
| fee_type | 货币类型 | 人民币传 CNY | 否 | 16 | 目前仅支持 CNY |
| trx_id | 交易单号 | 支付平台的内部交易流水号 | 否 | 20 | |
| chnl_trxid | 渠道平台交易单号 | 例如微信,支付宝平台的交易单号 | 是 | 50 | |
| proxy_trxid | 通联等第三方交易号 | 如走通联等第三方支付渠道,此字段传第三方支付渠道的交易号 | 是 | 50 | |
| trade_no | 缴费单号 | 缴费单号 | 否 | 32 | |
| trade_type | 支付方式 | 0-微信<br>1-通联微信<br>2-支付宝<br>3-通联支付宝 | 否 | 10 | 本次支付使用的支付方式 |
| order_status | 订单状态 | 4-已支付并成功生成保单<br>-4-已支付但保单生成失败 | 否 | 10 | |
| proposal_list　　出单成功后必传 | | | | | |
| proposal_no | 投保单号 | | 否 | 100 | |
| policy_no | 保单号 | | 否 | 100 | |
| c_dptacc_code | 做账机构 | | 否 | 200 | |
| 出单失败必传 | | | | | |
| fail_reason | 保单落地失败原因 | 出单失败必传 | 否 | 1000 | |

**响应参数:**

| 字段 ID | 字段名称 | 取值 | 可空 | 最大长度 | 备注 |
|---|---|---|---|---|---|
| return_code | 返回码 | SUCCESS/FAIL | 否 | 8 | SUCCESS/FAIL<br>SUCCESS表示商户接收通知成功并校验成功 |
| return_msg | 返回码说明 | | 是 | 100 | 返回信息，如非空，为 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | 错误原因：<br>签名失败<br>参数格式校验错误 |

# 4 错误编码

| 错误编码 | 描述 |
|---|---|
| 20008 | 请求信息传递空节点 |
| 10002 | 请求信息未传递必传信息 |
| 20011 | 签名校验错误 |
| 20018 | 支付类型不支持 |
| 20012 | 货币类型不支持 |
| 20013 | 签名类型不支持 |
| 20015 | 支付限制不支持 |
| 20024 | 投保单保费与预支付金额不一致(请检查总金额和分项金额是否匹配) |
| 20016 | 该订单已支付 |
| 20017 | 支付金额和已生成订单不符 |
| 10000 | 支付平台内部系统异常 |

# 5 附录

调用支付平台例文

```
package com.sinosoft.zhonglu.payment.demo;

import net.sf.json.JSONObject;
import org.apache.commons.codec.digest.DigestUtils;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.protocol.HTTP;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.UUID;

/**
 * 使用到的 jar 包及功能:

 * net.sf.json-生成及解析 json

 * commons-codec-md5 计算

 * org.apache.http-发起 http post 请求

 * 可根据项目中情况自己选用合适的 jar 包完成功能
 */
public class PaymentDemo {

    public static void main(String[] s) throws Exception {
        //请求相关参数

        //渠道相关信息
```

String appId = "201906161034016361262";//支付平台提供的应用 ID

String appSecret = "8JRrQId4p0FWqzjjOLxglzLXSQEB2m0Z ";//支付平台提供的应用密钥

//传固定值

String fee_type = "CNY";//货币类型　人民币传 CNY

String nonce_str = buildRandomStr();//随机字符串 商户自行生成的随机字符串

String validtime = "15";//有效时间 订单有效时间，以分为单位，不填默认为 15 分钟

String sign_type = "MD5";//签名类型 默认传 MD5

String unifiedOrderUrl="https://testzfpt.zhlic.com.cn/payment/unifiedorder/v2.0.0";//预下单接口地址

//根据业务定的传值

String total_fee = "1138";//交易金额 单位为分

String trade_no = buildRandomStr();//商户交易单号 商户的交易订单号

String body = "测试 1 分钱支付";//订单标题 订单商品名称

String attach = "test";//附加数据，在查询 API 和支付通知中原样返回，该字段主要用于商户携带订单的自定义数据

String remark = "测试支付备注";//附加数据，在查询 API 和支付通知中原样返回，该字段主要用于商户携带订单的自定义数据

String spbill_create_ip = "127.0.0.1";//终端 ip 发起调用 APP 的真实 IP 地址,注意如果支付类型为 JSAPI 时,这个传值务必准确

String notify_url = "http://www.baidu.com/payConfirmCallback";//支付成功之后通知

接口,需要按照 3.2 交易结果通知接口进行开发

String redirect_url = "http://www.baidu.com/";//支付成功后前端会跳转此地址

String limit_pay = "no_credit";//如果传值 no_credit 代表不允许使用信用卡支付,如无限制此字段可不传

String device_type = "iOS";//设备类型 Android 传 Android, iOS 传 iOS，H5 传 Wap

String app_name = "测试";//应用名 Android / iOS 应用传应用名，H5 传网站名

String app_bundle = "com.sinosoft.test";//应用 ID iOS 应用传 bundle_id,

String proposal_no = "1201919992999";//投保单号

String c_order_no = "2019010211989";//核心订单号

String money = "1138";//该险别金额

//生成签名

```
JSONArray proposalJsonArray = new JSONArray();
JSONObject proposalJson = new JSONObject();
proposalJson.put("proposal_no", proposal_no);
proposalJson.put("c_order_no", c_order_no);
proposalJson.put("money", money);

JSONObject reqJson = new JSONObject();
reqJson.put(“proposal_list”, proposalJson);
reqJson.put("app_id", appId);
reqJson.put("total_fee", total_fee);
reqJson.put("fee_type", fee_type);
reqJson.put("trade_no", trade_no);
reqJson.put("nonce_str", nonce_str);
reqJson.put("body", body);
reqJson.put("validtime", validTime);
reqJson.put("spbill_create_ip", spbill_create_ip);
reqJson.put("notify_url", notify_url);
reqJson.put("redirect_url", redirect_url);
reqJson.put("sign_type", sign_type);
reqJson.put("device_type", device_type);
reqJson.put("app_name", app_name);
```

```
        reqJson.put("app_bundle", app_bundle);
        reqJson.put("attach", attach);
        reqJson.put("limit_pay", limit_pay);
        String sign = sign(reqJson, appSecret);
        reqJson.put("sign", sign);
        String responseText=httpPost(unifiedOrderUrl,reqJson.toString());
        JSONObject respJson=JSONObject.fromObject(responseText);

if("SUCCESS".equals(respJson.optString("return_code"))&&"SUCCESS".equals(respJson.optString("
result_code"))){
            System.out.println("支付链接为:"+respJson.getString("jump_url"));
        }
    }

    public static String httpPost(String requestUrl, String requestText) {
        // 将 JSON 进行 UTF-8 编码,以便传输中文
        HttpClient client = new DefaultHttpClient();
        HttpPost httpPost = new HttpPost(requestUrl);
        httpPost.addHeader(HTTP.CONTENT_TYPE, "application/json");

        try {
            StringEntity se = new StringEntity(requestText, "utf-8");
            httpPost.setEntity(se);
            HttpResponse response = client.execute(httpPost);
            HttpEntity entity = response.getEntity();
            InputStream is = entity.getContent();
            BufferedReader reader = new BufferedReader(new InputStreamReader(is, "utf-8"));
            StringBuffer sb = new StringBuffer();
            String line = null;
            while ((line = reader.readLine()) != null) {
                sb.append(line);
            }
            return sb.toString();
        } catch (Exception e) {
            e.printStackTrace();
        }

        return null;
    }

    public    String sign(JSONObject jsonObject, String appSecret) throws Exception {
        HashMap<String, String> signMap = new HashMap<String, String>();
```

```
        for (Object key : jsonObject.keySet()) {
            Object value = jsonObject.get(key);
            if (value instanceof String) {
                signMap.put((String) key, (String) value);
            } else if (value instanceof JSONArray) {
                JSONArray jsonArray = (JSONArray) value;
                for (int i = 0; i < jsonArray.size(); i++) {
                    Object value1 = jsonArray.get(i);
                    if (value1 instanceof JSONObject) {
                        JSONObject jsonObject1 = (JSONObject) value1;
                        for (Object key1 : jsonObject1.keySet()) {
                            Object value2 = jsonObject1.get(key1);
                            if (value2 instanceof String) {
                                signMap.put((String) key1 + i, (String) value2);
                            }
                        }
                    }
                }
            }
        }
        signMap.put("app_secret", appSecret);
        ArrayList<String> keyArray = new ArrayList();
        for (String key : signMap.keySet()) {
            if (!"sign".equals(key)) {
                keyArray.add(key);
            }
        }
        Collections.sort(keyArray);
        String signStr = "";
        for (String key : keyArray) {
            signStr += "&" + key + "=" + signMap.get(key);
        }
        signStr = signStr.substring(1);
        String sign = DigestUtils.md5Hex(signStr.getBytes("UTF-8")).toUpperCase();
        return sign;
    }

    //一种简单的生成随机字符串的方式,实际使用中建议不要使用自己的随机数产生器
    private static String buildRandomStr() {
        return UUID.randomUUID().toString().replaceAll("-", "");
    }

}
```

渠道接收通知例文

```
package com.sinosoft.zhonglu.payment.demo;

import net.sf.json.JSONObject;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.BufferedReader;
import java.io.PrintWriter;
/**
 * 使用到的 jar 包及功能:
 * net.sf.json-生成及解析 json
 * commons-codec-md5 计算
 * org.apache.http-发起 http post 请求
 * 可根据项目中情况自己选用合适的 jar 包完成功能
 */
public class PaymentDemo {

    public static void main(HttpServletRequest req, HttpServletResponse resp) throws Exception {
        //渠道相关信息

        String appSecret = "8JRrQId4p0FWqzjjOLxglzLXSQEB2m0Z ";//支付平台提供的应用密
钥

        HttpServletRequest request = req;
        HttpServletResponse response = resp;
        request.setCharacterEncoding("utf-8");
        JSONObject reqJSON = null;
        PpOrder order = null;
        String app_id = null;
```

```
String trx_id = null;
String nonce_str = null;
String sign = null;

JSONObject returnJson = new JSONObject();
try {
	String strJSON = readJson(request);
	if ("".equals(strJSON)) {
		strJSON = "{}";
	}

	reqJSON = JSONObject.fromObject(strJSON);

	// 解析报文,并进行校验

	app_id = reqJSON.getString("app_id");
	trx_id = reqJSON.getString("trx_id");
	nonce_str = reqJSON.getString("nonce_str");
	sign = reqJSON.getString("sign");

	// 校验签名

	String serverSign = sign(reqJSON, appSecret);
	if (!serverSign.equals(sign)) {
		returnJson.put("return_code", "FAIL");

		returnJson.put("return_msg", "签名校验失败");

		return;
	}

	//进行数据保存操作，正常操作后返回成功标志位

	returnJson.put("return_code", "SUCCESS");
	return;
} catch (Exception e) {
	e.printStackTrace();
} finally {
	response.setContentType("text/html;charset=utf-8");
	response.setHeader("ContentType", "text/json");
	PrintWriter out = response.getWriter();
	out.print(returnJson.toString());
	out.flush();
	out.close();
}
```

```
}

/**
 * 读取 JSON 字符串
 * @param request
 * @return
 * @throws Exception
 */
private static String readJson(HttpServletRequest request) throws Exception {
	BufferedReader read = request.getReader();
	StringBuffer buffer = new StringBuffer();
	String string = "";
	while ((string = read.readLine()) != null) {
		buffer.append(string);
	}
	read.close();
	return buffer.toString();
}

/**
  *签名验证
  */
public	String sign(JSONObject jsonObject, String appSecret) throws Exception {
	HashMap<String, String> signMap = new HashMap<String, String>();
	for (Object key : jsonObject.keySet()) {
		Object value = jsonObject.get(key);
		if (value instanceof String) {
			signMap.put((String) key, (String) value);
		} else if (value instanceof JSONArray) {
			JSONArray jsonArray = (JSONArray) value;
			for (int i = 0; i < jsonArray.size(); i++) {
				Object value1 = jsonArray.get(i);
				if (value1 instanceof JSONObject) {
					JSONObject jsonObject1 = (JSONObject) value1;
					for (Object key1 : jsonObject1.keySet()) {
						Object value2 = jsonObject1.get(key1);
						if (value2 instanceof String) {
							signMap.put((String) key1 + i, (String) value2);
						}
					}
				}
			}
```

```
            }
        }
        signMap.put("app_secret", appSecret);
        ArrayList<String> keyArray = new ArrayList();
        for (String key : signMap.keySet()) {
            if (!"sign".equals(key)) {
                keyArray.add(key);
            }
        }
        Collections.sort(keyArray);
        String signStr = "";
        for (String key : keyArray) {
            signStr += "&" + key + "=" + signMap.get(key);
        }
        signStr = signStr.substring(1);
        String sign = DigestUtils.md5Hex(signStr.getBytes("UTF-8")).toUpperCase();
        return sign;
    }


}
```