GoPay黄金支付

- `GoPay黄金支付` 是全网领先的支付技术解决方案，致力于帮助企业以最低成本快速接入一套稳定可靠的支付系统，并以可视化的数据辅助其完成商业决策。作为一家以数据服务为核心竞争力的支付服务商，我们不仅关注支付系统的高并发高可用，更关注挖掘支付背后的巨大价值，关注每一位伙伴企业的成长。
- `GoPay API` 面向企业应用开发者提供微信支付、支付宝、银联支付、京东钱包、QQ钱包、百度钱包、银行卡支付等主流支付渠道代接入服务，一站式解决支付接入、信息核验、数据分析等交易问题。
- 同时，我们坚信，支付不仅是交易的结束，更是交易的开始。未来，我们将专注于以数据驱动用户业务的增长，辅助企业商业决策，发挥每一笔支付交易的价值。

> 黄金发发科技：https://www.gopay88.io

目录

项目介绍

GoPay黄金支付

项目介绍

- `GoPay黄金支付` 是全网领先的支付技术解决方案，致力于帮助企业以最低成本快速接入一套稳定可靠的支付系统，并以可视化的数据辅助其完成商业决策。作为一家以数据服务为核心竞争力的支付服务商，我们不仅关注支付系统的高并发高可用，更关注挖掘支付背后的巨大价值，关注每一位伙伴企业的成长。
- `GoPay API` 面向企业应用开发者提供微信支付、支付宝、银联支付、京东钱包、QQ钱包、百度钱包、银行卡支付等主流支付渠道代接入服务，一站式解决支付接入、信息核验、数据分析等交易问题。

- 同时，我们坚信，支付不仅是交易的结束，更是交易的开始。未来，我们将专注于以数据驱动用户业务的增长，辅助企业商业决策，发挥每一笔支付交易的价值。

> 黄金发发科技：https://www.gopay88.io

在线演示

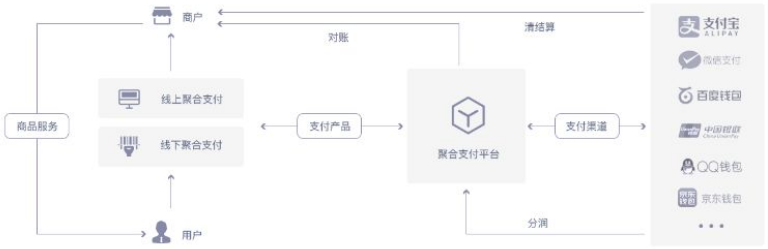
商户： https://pay.gopay88.io (demo@gopay88.io nouser)

码商： https://pay.gopay88.io/ma (mademo@gopay88.io nouser)

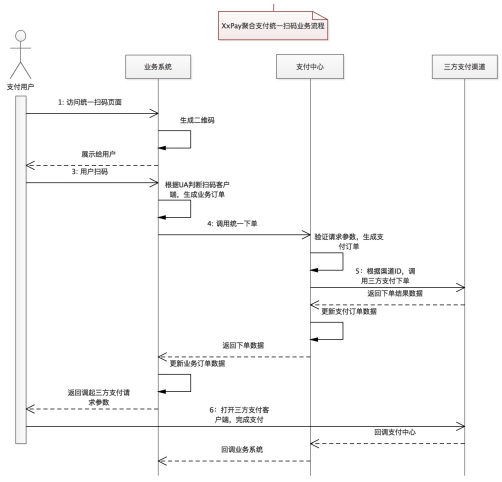
版权信息

场景流程

业务场景



业务流程



接口规则

协议规则

传输方式	为保证交易安全性，采用HTTPS传输
提交方式	采用POST方法提交
数据格式	提交和返回数据都为JSON格式，基本数据{ "result_code":"OK", "result_msg":"SUCCESS", "charge":{}}
字符编码	统一采用UTF-8字符编码
签名算法	SHA1WithRSA等
签名要求	请求和接收数据都需要校验签名
授权要求	所有API操作都要求使用授权
判断逻辑	先判断协议字段返回，再判断业务返回，最后判断交易状态

HEADER规范

1、请求Content-Type： `content-type`
设置该Header值为JSON格式。例：application/json; charset=UTF-8
2、请求Accept： `accept`
设置该Header值为JSON格式。例：application/json; charset=UTF-8
3、请求URL地址： `x-ca-resturl`
设置该Header值为统一下达或订单查询请求的URL地址。例：
https://pay.gopay88.io/pay/unifiedorder 或
https://pay.gopay88.io/pay/orderquery
4、请求时间戳： `x-ca-timestamp`
设置该Header值为当前请求的毫秒或更高精度时间戳（毫秒/微秒/纳秒）。
例：毫秒时间戳 1586007620038
5、请求随机数： `x-ca-noncestr`
设置该Header值为一个32位系统随机数，防重请求。例：MD5(系统随机数)
6、请求授权KEY： `x-ca-auth`
设置该Header值为商户申请API接口获取的 `key` 。例：
772ae1d32322f49508307b2f31a0107f
7、请求签名： `x-ca-signature`
设置该Header值为使用SHA1WithRSA签名算法计算的签名，该签名将由服务端进行校验。具体算法规则参见下一节

请求/返回的签名和校验算法

第一步、商户系统API接口会分配 `key` 和支付平台数据公钥 `platform_pubKey` ，使用支付宝RSA签名验签工具自己生成商户数据RSA密钥对（私钥 `priKey` 和公钥 `pubKey` ，注：非JAVA适用 2048 pkcs1s）：

> `key` ： 商户系统API接口对应的key
> `priKey` ： 对商户系统请求数据进行签名的RSA私钥
> `pubKey` ： 支付平台对商户系统请求数据进行验签的RSA公钥
> `platform_pubKey` ： 商户系统对支付平台返回数据进行验签的RSA公钥

第二步、拼接待签名的字符串 `string` ，规则如下：

> `string` = UTF8格式的URI +
> 换行符 `\n` + UTF8格式的URI查询参数 +
> 换行符 `\n` + UTF8格式的 `x-ca-noncestr` Header值 +
> 换行符 `\n` + UTF8格式的 `x-ca-timestamp` Header值 +
> 换行符 `\n` + UTF8格式的POST请求JSON数据
>
> 例：（注：空行是该请求的URI查询参数为空）
> `string` = /pay/unifiedorder
>
> C8E1D385785625AFD64A484B58F91882
> 1.58600995149E+12
> {"out_trade_no":"20200704011813158619349 3","subject":"VIP\u5145\u503c - 100001","body":"VIP\u5145\u503c - 100001","amount":"1.66","currency":"CNY","channel":"ma_ali_h5","extparam":[],"mchid":"100001","return_url":"https://pay/demo.html","notify_url":"https://pay.gopay88.io/cashier/demonotify","client_ip":"127.0.0.1"}

第三步、使用SHA1WithRSA签名算法计算得出 `x-ca-signature` 签名，如下：

> 1、对待签名字符串 `string` 进行 `Base64` 运算得到 `sign` 值
> 2、读取商户私钥 `priKey` 并转换为openssl密钥 `private key`
> 3、将 `sign` 值和 `private key` 密钥传入SHA1WithRSA签名算法计算出 `signature`
> 4、对 `signature` 进行 `Base64` 运算得到 `x-ca-signature` 值

第四步、支付平台收到商户系统请求后，将对请求数据进行验签。如果验签不通过，返回失败结果；如果验签通过，执行相应的支付业务逻辑并返回相应的结果。支付平台对返回结果进行了SHA1WithRSA签名算法，返回结果和Header示例如下：

```
Body:
{
        "result_code": "OK",
        "result_msg": "SUCCESS",
        "charge": {
                "channel": "gopay_payout",
                "out_trade_no": "05Apr2021084746550",
                "client_ip": "3.6.93.106",
                "amount": "100",
                "currency": "CNY",
                "subject": "gopay_payout_测试",
                "body": "gopay_payout_测试",
                "extparam": {
                        "accountname": "胡三",
                        "bankaccount": "6228480415647314871
                        "bankname": "中国工商银行"
                },
                "credential": {
                        "out_trade_no": "D20210405084748405
                }
        }
}

Header:
x-ca-timestamp: 1617583668305
x-ca-noncestr: 963613FA553D6405C6E0D345BA32B6DB
x-ca-signature: f+boCXBdpfZItp4TykM7BB03qGTZ9Yf6eBZD8D6mPQC
```

第五步、商户系统收到返回结果后，为保证支付安全，强烈建议商户对返回数据进行验签后再进行相应的业务处理。验签算法如下：

> 拼接待验签的字符串 `string` ，规则如下：
> `string` = UTF8格式的 `x-ca-noncestr` Header值 + 换行符 `\n` + UTF8格式的 `x-ca-timestamp` Header值 + 换行符 `\n` + UTF8格式的返回结果的charge JSON数据
>
> 然后对待验签字符串 `string` 进行 `Base64` 运算得到 `verify` 值，传入 返回Header的 `x-ca-signature` 值、 `verify` 值、支付平台数据公钥 `platform_pubKey` 这三个参数执行SHA1WithRSA验签算法。

错误码

错误码

HTTP状态码	错误码	错误类型	错误提示	错误原因	解决方案
404	100000	MissException	Global: Your Required Resource Are Not Found	HTTP 404状态码时抛出此异常，即找不到对应请求资源	检查请求URL链接是否正确
400	200000	OrderException	Order does not exist.	订单不存在	检查订单号是否正确
400	200003	OrderException	Order Error.	订单状态错误	检查订单状态
400	400006	OrderException	Route Payment Error. [No available channels]	订单支付通道不可用	检查订单的支付通道参数和配置
400	400008	OrderException	Route Payment Error. [No available merchants account.]	订单支付账号不可用	检查订单的支付账号参数和配置
400	400008	OrderException	Route Payment Error. [No available ma code.]	订单没有可用的码卡	检查码卡状态和配置

HTTP状态码	错误码	错误类型	错误提示	错误原因	解决方案
400	400008	OrderException	Route Payment Error. [Ma code disconnected, Pls reorder.]	订单选择的码卡突然掉线	请重新下单
400	400008	OrderException	Route Payment Error. [Payment account was misconfigured.]	订单支付账号或码卡配置错误	检查订单的支付账号配置或码卡状态
400	100000	ParameterException	invalid parameters	请求参数不合法或缺少参数	确保传入参数合法性和完整性
400	400000	ParameterException	Invalid Request.[Request header [xxx] Failure.]	请求Header头 xxx不合法	确保请求Header头 xxx合法或正确配置
400	400003	ParameterException	Invalid Request.[Auth Key No permission or nexistencet.]	请求授权Key参数不合法	确保请求授权码Key参数合法或正确配置
400	400003	ParameterException	Invalid Request.[Payment Code Does Not Allowed.]	请求支付通道码参数不合法	确保请求支付通道码参数合法或正确配置
400	400003	ParameterException	Invalid Request.[Payment Code Does Not Allowed.]	请求支付通道码参数不合法	确保请求支付通道码参数合法或正确配置
403	100003	ForbiddenException	Invalid Request.[Trigger Restriction And Flow Control.]	同一IP每秒接口访问频率超过10次限制	稍后再试或请联系平台管理员协查

HTTP状态码	错误码	错误类型	错误提示	错误原因	解决方案
403	400003	ForbiddenException	Invalid Request.[Request IP not authorized.]	请求IP未授权，被阻止	确保请求IP在授权列表中
403	400003	SignatureException	Invalid Request.[Request Data And Sign Verify Failure.]	商户支付请求数据验签失败	仔细阅读API接口文档的接口规则，检查相关请求参数，如还无法解决请联系平台管理员协查
404	600000	UserException	User do not exist.	用户不存在	检查用户信息是否正确
400	999999	BaseException	Error .[There may be a problem with the system.]	通用系统错误	请联系平台管理员协查

统一下单

业务说明

统一下单接口，商户系统先调用该接口在支付服务后台生成预支付交易单

接口网关

URL地址：https://pay.gopay88.io/pay/unifiedorder **注意：** 此接口为测试接口，请向业务人员索要

请求参数

字段名	变量名	类型	必填	示例值	说明
商户UID	mchid	string	True	100001	商户UID

字段名	变量名	类型	必填	示例值	说明
商户订单号	out_trade_no	string	True	1009660380201506130	商户订单号
商品描述	subject	string	True	会员充值	商品简单描述
商品信息	body	string	True	会员充值	支付商品信息
支付金额	amount	float	True	1000.00	支付总金额,两位精度
支付产品	channel	string	True	gopay_payout	(1) 支付方式: gopay_banktransfer gopay_bankupi gopay_quickupi (2) 代付方式: gopay_payout
附加数据	extparam	string	True		附加参数: (1)如果是支付请求(channel为gopay_bankupi等),则extparam需要添加userid参数(商户侧的用户唯一ID),参数格式为json (2)如果是代付请求(channel为gopay_payout),则extparam需要添加userid bankname bankcode bankaccount accountname参数,参数格式为json
货币代码	currency	string	True	CNY	支付币种: CNY, INR

字段名	变量名	类型	必填	示例值	说明
终端IP	client_ip	string	True	127.0.0.1	发起支付IP
通知地址	notify_url	string	True		异步通知地址
回调地址	return_url	string	True		同步回调地址

> [warning] extparam参数说明

当channel是代收gopay_bankupi等时，extparam需要添加userid参数(商户侧的用户唯一ID):

```
{"userid":"demopay000001"}
```

当channel是代付gopay_payout时，extparam需要添加userid bankname bankcode bankaccount accountname参数:

```
{"userid":"demopay000001","bankname":"中国工商银行","bankcode
```

请求数据

> 以下是下单请求的数据样例

```
{
    "mchid": 100003,
    "out_trade_no": "05Apr2021082410936",
    "subject": "gopay_payout_测试",
    "body": "gopay_payout_测试",
    "channel": "gopay_payout",
    "extparam": {
      "accountname": "胡三",
      "bankaccount": "6228480415647314871",
      "bankname": "中国工商银行"
    },
    "amount": 100,
    "currency": "CNY",
    "client_ip": "3.6.93.106",
    "return_url": "http://gopay.local/demo.html",
    "notify_url": "http://gopay.local/cashier/demonotify"
  }
```

返回参数

字段名	变量名	类型	必填	示例值	说明
错误码	result_code	string	True	OK	结果码. OK:成功, 其他:失败
返回消息	result_msg	string	True	SUCCESS	提示信息. SUCCESS:成功
数据对象	charge	object	True		返回支付对象（请看下方具体数据） [warning]注意： status为订单业务状态 0-关闭订单 1-等待支付 2-支付成功 3-支付失败

返回结果

> [success] 这里是**成功返回信息** 在返回JSON数据中 **result_code=OK** 和 **result_msg= SUCCESS** 时才有 ***charge***
>
> [warning]注意：status为订单业务状态
> 0-关闭订单
> 1-等待支付
> 2-支付成功
> 3-支付失败


```
{
        "result_code": "OK",
        "result_msg": "SUCCESS",
        "charge": {
                "channel": "gopay_payout",
                "out_trade_no": "05Apr2021084746550",
                "client_ip": "3.6.93.106",
                "amount": "100",
                "currency": "CNY",
                "subject": "gopay_payout_测试",
                "body": "gopay_payout_测试",
                "extparam": {
                        "accountname": "胡三",
                        "bankaccount": "6228480415647314871
                        "bankname": "中国工商银行"
```

```
                },
                "credential": {
                        "out_trade_no": "D20210405084748405
                }
        }
}
```

> [error] 这里是**错误返回信息** 在返回错误数据中仅有 **error_code** 和 **error_msg**

```
{
    "error_msg": "Invalid Request.[ Request header [authent
    "error_code": 400000
}
```

订单回调

应用场景

支付完成后， `支付中心` 会把相关支付结果和相关信息发送给商户，商户需要接收处理，并返回应答。对后台通知交互时，如果支付中心收到商户的应答不是成功或超时，支付中心则认为通知失败，会在一定的策略定期重新发起通知，尽可能提高通知的成功率，但不保证通知最终能成功。（通知频率为 `15/15/30/180/1800/1800/1800/1800/3600` ，单位： `秒` ）

> [warning]注意：同样的通知可能会多次发送给商户系统。商户系统必须能够正确处理重复的通知。推荐的做法是，当收到通知进行处理时，首先检查对应业务数据的状态，判断该通知是否已经处理过，如果没有处理过再进行处理，如果处理过直接返回结果成功。在对业务数据进行状态检查和处理之前，要采用数据锁进行并发控制，以避免函数重入造成的数据混乱。

> [danger]特别提醒：商户系统对于订单回调的内容一定要做签名验证，并校验返回的订单金额是否与商户侧的订单金额一致，防止数据泄漏导致出现“假通知”，造成资金损失。

接口网关

统一下单接口提交的参数 `notify_url` 设置，如果无法访问链接，您的业务系统将无法接收到支付中心的通知。

通知和应答格式

字段名	变量名	类型	必填	示例值	说明
返回码	result_code	string	True	OK	结果码. OK:成功, 其他:失败
返回消息	result_msg	string	True	SUCCESS	提示信息. SUCCESS: 成功

字段名	变量名	类型	必填	示例值	说明
数据对象	charge	object	True		返回支付对象（请看下方具体数据） [warning]注意： status为订单业务状态 0-关闭订单 1-等待支付 2-支付成功 3-支付失败

通知数据

> 以下是成功通知的数据样例
> [warning]注意：status为订单业务状态
> 0-关闭订单
> 1-等待支付
> 2-支付成功
> 3-支付失败


```
{
    "result_code": "OK",
    "result_msg": "SUCCESS",
    "charge": {
      "puid": 0,
      "pmid": 0,
      "mid": 200001,
      "out_trade_no": "20210504085721161758424l",
      "in_trade_no": "",
      "subject": "VIP/u5145/u503c - 100000",
      "body": "VIP/u5145/u503c - 100000",
      "channel": "gopay_banktransfer",
      "paytype": 1,
      "extra": "[]",
      "amount": "100.000",
      "pay_amount": "100.000",
      "keyword": null,
      "channel_in": "0.000",
      "agent_channel_in": "0.000",
      "income": "0.000",
      "user_in": "0.000",
      "agent_in": "0.000",
      "platform_in": "0.000",
      "currency": "CNY",
      "client_ip": "3.6.93.106",
      "return_url": "https://golden88.io/demo.html",
      "notify_url": "https://golden88.io/cashier/demonotify
```

```
      "status": 2
    }
  }
```

应答数据

> 以下是成功应答的数据样例

```
{
    "result_code": "OK",
    "result_msg": "SUCCESS"
  }
```

> 以下是失败应答的数据样例

```
{
    "result_code": "OK",
    "result_msg": "FAIL"
  }
```

订单查询

接口网关

URL地址：https://pay.gopay88.io/pay/orderquery **注意：** 此接口为测试接口，请向业务人员索要

请求参数

字段名	变量名	类型	必填	示例值	说明
商户UID	mchid	string	True	100001	商户平台注册UID
商户订单号	out_trade_no	string	True	1009660380201506130	商户订单号
支付产品	channel	string	True	gopay_payout	(1) 支付方式: gopay_banktransfer gopay_bankupi gopay_quickupi (2) 代付方式: gopay_payout

返回格式

字段名	变量名	类型	必填	示例值	说明
返回码	result_code	string	True	OK	结果码. OK:成功, 其他:失败
返回消息	result_msg	string	True	SUCCESS	提示信息. SUCCESS:成功
数据对象	charge	object	True		返回支付对象（请看下方具体数据） [warning]注意： status为订单业务状态 0-关闭订单 1-等待支付 2-支付成功 3-支付失败

返回结果

> [success] 这里是**成功返回信息** 在返回JSON数据中 **result_code=OK** 和 **result_msg= SUCCESS** 时才有 ***charge***
>
> [warning]注意：status为订单业务状态
> 0-关闭订单
> 1-等待支付
> 2-支付成功
> 3-支付失败


```
{
        "result_code": "OK",
        "result_msg": "SUCCESS",
        "charge": {
                "trade_no": "C2020040701181440794",
                "out_trade_no": "20200704011813158619349 3",
                "subject": "VIP充值 - 100001",
                "body": "VIP充值 - 100001",
                "extparam": "[]",
                "amount": "1.660",
                "channel": "ma_ali_h5",
                "currency": "CNY",
                "client_ip": "127.0.0.1",
                "status": "1"
```

```
        }
}
```

> [error] 这里是**错误返回信息** 在返回错误数据中仅有 **error_code** 和 **error_msg**

```
{
    "error_msg": "Invalid Request.[ Request header [authent
    "error_code": 400000
}
```