

商户支付接口文档

使用现有测试商户直接连接正式接口，从下单到查询与通知验签。提供可运行的 C#、Python、PHP 示例，帮助完成对接与验收。

[文档 2.9 · 2026-10-21](#)[UTF-8 表单 · MD5](#)[内置账号 demo · 同一套正式接口](#)[下载 PDF（与在线文档同步）](#)

CRYPTO QUICK START

数字货币 v2：下载即可运行的闭环示例

Python、PHP、C# 均提供下单、查询、HTTP 通知验签、SQLite 事务、事件去重、乱序保护和本地结果查询。默认测试通道 `2911`、`1.00 TEST-USDT`，可换 `2912/2913`，固定传入并签名 `crypto_callback_version=2`。

[下载数字货币 v2 完整示例](#) [完整运行说明](#)

共享钱包一次只服务一笔订单。换金额或单号不会解除占用。已提交过请求先查询原单号；其他开发者占用时等待原订单到期或联系平台。成功订单也可能占用至原截止时间。多个钱包才可轮转。

1. 解压并进入 `merchant-examples/crypto-v2`。运行 `selftest`，先离线验证 28 组通知，无需测试币或公网。
2. 选择一种语言运行接收器，将自己的 HTTPS 回调反代到本地监听地址，在 `config.json` 填入 `PAY_NOTIFY_URL`。默认占位地址会阻止在线下单，避免误以为已验收回调。
3. 运行 `balance`、`demo`；`demo` 保存商户单号，创建后查询，不自动转账。
4. 按实际订单响应支付测试币，运行 `inspect` 查看已验签落库的 `events`、`states` 和 `outbox`。确认中不发货，`confirmed` 才产生一次业务待办。

```
# 以下三组任选一种；运行目录 merchant-examples/crypto-v2
python3 python/app.py selftest
python3 python/app.py serve 8081 # /notify, 单独终端保持运行
python3 python/app.py demo
python3 python/app.py inspect

php php/app.php selftest          # curl + pdo_sqlite
php -S 127.0.0.1:8082 -t php      # /notify.php, 单独终端
php php/app.php demo
php php/app.php inspect

dotnet run --project csharp -- selftest # .NET 8 + SQLite NuGet
dotnet run --project csharp -- serve 8083
```

```
dotnet run --project csharp -- demo
dotnet run --project csharp -- inspect
```

三种语言各自使用独立 SQLite；同一接入的客户端与接收器必须共用配置和 PAY_STATE_DIR。源码：[Python](#) · [PHP](#) · [C#](#) · [事务表结构](#) · [离线签名通知数据](#)。

如何确认回调已经正确处理

旧默认地址 `/?c=Pay&a=notify_url` 仅支持 v1 固定字段验签及 status=success 检查，**不支持 v2，也没有开发者通知结果查询页面**。它不证明业务持久化或幂等。数字货币闭环必须使用自己的 v2 接收器。

新示例只有验签、订单核对、事件/状态/outbox 同事务提交成功后，才返回 HTTP 200 纯文本 success。inspect 的 events 中存在对应 event_id 是本地验签持久化证据；重复通知不能增加 outbox。平台订单的通知状态和回复内容展示投递结果，本地 outbox 代表待执行业务，示例不实际发货。

人工测试通知按已签名 reason=test_callback 放入独立 test 流，不改变 payment 状态；test outbox 不能用于真实发货。正常测试网交易通知仍应在测试业务账务中验收。数据库暂时不可写返回 503，非法签名/身份/状态返回 400；重试时间不套用下单请求 300 秒限制。

CRYPTO RESPONSES

数字货币数量与真实响应

测试通道	decimals	money	amount	expected_units
2911 Ethereum / 2912 TRON	6	"1.00"	"1.000000"	"1000000"
2913 BSC	18	"1.00"	"1.000000000000000000"	"1000000000000000000"

money 为 USDT/TEST-USDT 订单数量（现行请求最多两位小数）； $\text{amount} = \text{expected_units} / 10^{\text{decimals}}$ ，是展示字符串。expected_units 必须当作精确整数字符串，不用浮点数，也不将精度固定为 6。精度由本单合约/网络响应决定。服务费另看 crypto.fee，不从 amount 中扣除。

查询返回 original_amount、expected_units、decimals，不含下单展示用 amount；v2 回调的 expected_amount_raw 对应 expected_units，amount_raw 是实际转账最小单位数量（未付款为空）。示例要求精确相等，人工差额补单需商户额外审核。

下列为 2026-10-24 公开 demo 实际下单与签名查询响应。新订单按 TEST-USDT 冻结服务费；本轮未转账，随后取消测试订单并验证释放，故查询显示 cancelled / released。收银台 token 已脱敏。不要向示例历史地址付款。

▼ 下单完整 JSON（2911）

```
{
  "code": "1",
  "msg": "下单成功",
  "data": {
    "order_sn": "Ceff0a0c1acde5428414a52366efaf56b",
    "ptype": 2911,
    "ptype_name": "ethereum_sepolia TEST-USDT",
    "money": "1.00",
    "currency": "TEST-USDT",
```

```

"pay_url": "https://payment.271458.xyz/?c=Pay&a=crypto&token=REDACTED_TEST_TOKEN",
"crypto": {
  "fee": {
    "currency": "TEST-USDT",
    "fee_usdt": "0.010000000000000000000000",
    "environment": "testnet",
    "rate": "0.010000",
    "rule_version": "lifetime-whole-order-v1",
    "volume_before": "0.000000000000000000000000",
    "estimated_fee": "0.010000000000000000000000",
    "charged_fee": "0.000000000000000000000000",
    "state": "held"
  },
  "merchant_id": 1001,
  "is_testnet": 1,
  "address": "0xbcbc11e3a17428a258beebd3849d741581762d96",
  "decimals": 6,
  "expected_units": "1000000",
  "network": "ethereum_sepolia",
  "asset": "0xb464a579e9697694833f544d1b464cbd51405c84",
  "currency": "TEST-USDT",
  "amount": "1.000000",
  "expires_at": 1792856943,
  "payment_status": "awaiting",
  "payment_state": "unpaid",
  "confirmation_state": "unconfirmed",
  "callback_version": 2,
  "event_seq": 0
}
}
}

```

▼ 查询完整 JSON (2911)

```

{
  "code": "1",
  "msg": "ok",
  "data": {
    "mch_id": "demo",
    "out_order_sn": "USDTACCEPT_1792853342_2911",
    "orders": [
      {
        "order_sn": "Ceff0a0c1acde5428414a52366efaf56b",
        "money": "1.00",
        "currency": "TEST-USDT",
        "pay_money": "1.00",
        "order_time": "1792853343",
        "pay_time": "0",
        "status": "4",
        "status_flag": "已取消",
        "notice_status": "1",
        "notice_status_flag": "待通知",
        "notice_msg": null,
        "crypto": {
          "ptype": 2911,

```

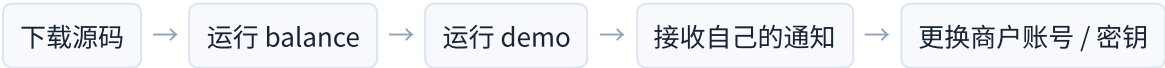
```
    "network": "ethereum_sepolia",
    "asset": "0xb464a579e9697694833f544d1b464cbd51405c84",
    "address": "0xbcb11e3a17428a258beebd3849d741581762d96",
    "currency": "TEST-USDT",
    "original_amount": "1.00",
    "expected_units": "1000000",
    "decimals": 6,
    "expires_at": 1792856943,
    "payment_status": "cancelled",
    "payment_state": "unpaid",
    "confirmation_state": "unconfirmed",
    "callback_version": 2,
    "event_seq": 0,
    "merchant_id": 1001,
    "is_testnet": 1,
    "fee": {
      "currency": "TEST-USDT",
      "fee_usdt": "0.010000000000000000000000000000",
      "environment": "testnet",
      "rate": "0.010000",
      "rule_version": "lifetime-whole-order-v1",
      "volume_before": "0.0000000000000000000000000000",
      "estimated_fee": "0.0100000000000000000000000000",
      "charged_fee": "0.0000000000000000000000000000",
      "state": "released"
    },
    "receipts": []
  }
}
]
```

三链完整 JSON：[2911 下单](#) · [2911 查询](#) · [2912 下单](#) · [2912 查询](#) · [2913 下单](#) · [2913 查询](#)

01 / GET STARTED

下载后直接联调

源码包已内置平台现有测试商户 **demo** 和现有签名密钥，直接走正式接口。无需另建账号、领取配置或切换环境；对接后换成自己的商户账号和密钥，接口与签名代码继续使用。



用途	地址 / 约定
下单	https://payment.271458.xyz/?c=Pay
查询	https://payment.271458.xyz/?c=Pay&a=query
余额查询	https://payment.271458.xyz/?c=Pay&a=querybalance
普通 v1 示例默认账号 / 通道	demo / 2601 (GPay) ； 示例金额 10000.00 ， 随商户业务调整。

用途	地址 / 约定
请求编码	POST, <code>application/x-www-form-urlencoded</code> ; <code>charset=UTF-8</code> 。format=json 控制响应，不是发送 JSON 请求体。
时间戳	UTC 秒级 Unix 时间戳，与服务器允许偏差最多 300 秒。

1. 下载源码包并解压，选择 Python、PHP 或 C#。
2. 运行 `selftest` 离线校验签名，运行 `balance` 验证正式接口身份。
3. 运行 `demo`，自动生成随机商户单号，下单后立即查询。
4. 对接自己的回调服务；完成后在 `merchant-config.json` 更换账号和密钥。

code=1 仅表示本次请求成功。下单成功不是支付成功；发货/入账前必须正确验签并核对通知，或通过查询确认订单状态。

下载 C# / Python / PHP 源码包

查看已内置的账号与密钥

下载 PDF

02 / SHARED TEST MERCHANT

现有测试商户 · 同一套正式接口

后台试用：[登录商户后台](#)，账号 `demo`，密码 `demo`，填写页面图形验证码即可登录并查看此测试商户的交易记录。内部 ID 为 1001。API 签名仍使用示例配置中的 `PAY_KEY`，不能使用登录密码签名。

平台指定 `demo` 作为公开联调账号，它的现有 key 已包含在源码包的 `merchant-config.json`。三种语言读取同一份配置，不需要修改程序逻辑或启用额外开关。

配置项	默认值 / 使用方式
PAY_GATEWAY	<code>https://payment.271458.xyz/</code> ，测试账号和正式商户使用同一网关。
PAY_MCH_ID	<code>demo</code> ，联调完成后改成自己的商户登录账号。
PAY_KEY	已填入上述账号现有密钥；更换商户时同时改成自己的 key。
PAY_PTYPE / PAY_MONEY	<code>2601 / 10000.00</code> ，是当前测试账号配置；自己的商户需开通对应通道，金额依实际限额。
PAY_NOTIFY_URL	默认平台测试通知入口；开始接入自己的业务通知时改成自己的公网 HTTPS 地址。

配置优先级：环境变量 > 可选私有 `merchant.env`（或 `PAY_ENV_FILE` 指定文件）> 源码包内置 JSON。直接运行不需要任何环境变量。数字通知商户 ID 可额外核对，但不是开始联调的必填配置。

共享测试账号的订单与余额用于联调，可能被其他联调者改变；自己的正式商户身份、订单与账务分别处理。
`demo` 自动生成随机订单号，避免复制固定例子造成撞单。

平台将该商户用于测试账务；它仍使用真实 API 和当前通道。当前 GPay 配置指向正式上游，代码没有按此商户强制切为上游虚拟环境的分支。示例不会自动付款、代付或充值；返回的收款账号/二维码可能接收真实资金，不能当作零资金效果的模拟码。验证账号和签名可先运行 balance，无需付款。

▼ 可选独立模拟环境（不属于默认接入流程）

之前的 /sandbox.php、sandbox.env 仍可显式使用，仅适合本地模拟订单、通知和幂等测试。它不代表真实通道验收；simulate / notify / reset 只在该环境有效，默认示例不使用。

03 / CREATE ORDER

下单接口

POST

/?c=Pay

签名使用下表除 sign 以外的九项字段；可空字段也需要发送并参与签名。不要把路由 c、a 加入业务表单。

字段	传入	说明
mch_id	必填	商户登录账号；示例已填写平台现有测试账号 demo。
ptype	必填	通道编号。示例 2601；当前接口只接受大于 900 的通道，再按商户开通配置校验。
order_sn	必填	商户业务订单号，最多 32 字符；建议字母数字、下划线或短横线，全局唯一；共享测试账号也须使用随机单号。
money	必填	十进制金额字符串，如 1.00。币种、实际限额以商户开通约定为准；不要用浮点数拼接签名。
goods_desc	可空，必须传	商品描述，空值发送 goods_desc=。
client_ip	可空，必须传	下单用户 IP，空值发送 client_ip=。
format	必填	显式发送 json 或 page，参与签名。
crypto_callback_version	可选，仅数字货币	省略或 1 保持旧协议；填 2 启用三状态通知。传入时必须作为原始字符串参与下单签名。只支持 2901/2902/2903/2911/2912/2913，见数字货币 v2。
notify_url	必填	商户接收通知的 HTTPS 地址；使用原始 URL 值签名，再进行表单编码。不是收银台地址。
time	必填	当前 UTC 秒级 Unix 时间戳，以字符串发送。
sign	必填	32 位小写 MD5；sign 自身不参与计算。

当前生产代码会对 notify_url 额外执行一次 URL 解码及 HTML 实体解码。请使用简单、稳定的通知路径，避免 URL 内嵌百分号编码、+ 或 HTML 实体导致签名值变化；普通查询参数中的 & 应按表单正确编码。示例按照原始字符串签名。

生产普通明文接口的公共参数层还会 trim、引号转义、百分号转义和 HTML 转义。goods_desc 等字段含引号、&、<、>、% 或首尾空白时，值可能在验签前发生变化。示例默认使用空描述；生产特殊字符须单独联调。UTF-8 固定向量验证的是签名算法，不代表旧参数过滤对所有特殊字符已兼容。

签名规则与固定样例

1. 确定业务字段，保留空字符串，排除 sign。
2. 参数名按 ASCII 字典序升序排列，拼接原始 key=value，使用 & 连接。
3. 末尾追加 &key=签名密钥，不加空格、引号或换行。
4. 以 UTF-8 计算 MD5，输出小写十六进制；再做表单编码发送。

离线签名固定向量

此 key 和历史时间戳仅用于离线算法校验，不能提交网关。所有语言的 selftest 都包含该向量。

```
client_ip=&format=json&goods_desc=&mch_id=demo_merchant&money=500.00&notify_url=https://merchant.example/notify&order_sn=DEM0202609170001&ptype=12&time=1789574400&key=test_key_only_20260917
```

预期结果： 1ff18ca5f270edfb71f29b2f7f3a5e11

500、500.0、500.00 签名不同；空串、null 和省略不同；URL 原始值与已编码值不同。不要签名后再格式化金额、改写 URL，或对整段 JSON 计算 MD5。

请求	参与签名的字段
下单	原九项非 sign 字段；传入 crypto_callback_version 时额外包含该字段
订单查询	mch_id、out_order_sn、time
余额查询	mch_id、time
支付通知	mch_id、pt_order、sh_order、money、pay_money、goods_desc、status、time；当前 msg 不参与

密钥只保存在服务端。验签使用恒定时间比较；不要输出密钥或完整的含 key 签名原串到日志。

下单响应与收银台

format=json

```
{
  "code": "1", "msg": "下单成功",
  "data": {
    "order_sn": "2026101700010012", "ptype": "2601",
    "ptype_name": "GPay支付", "money": "10000.00",
    "pay_url": "https://通道返回的收银台或二维码地址"
```

```
}  
}
```

保存 data.order_sn（平台单号），优先使用通道返回的 data.pay_url。**pay_url 的具体形式取决于通道：可能是收银台页面，也可能是通道返回的二维码图片 URL。**当前 GPay 集成返回上游提供的二维码链接，不要统一假定它是 HTML 页面或本平台 /info 地址。它也不是商户通知地址。部分通道可能额外返回 extra，不保证所有通道返回相同收款字段。

format=page

成功返回 HTTP 302 和 Location，目标与通道下单返回的链接一致。后端调试需关闭自动跳转检查首跳；失败仍可能返回 HTTP 200 的 JSON。相对 Location 以网关地址解析。

```
{"code": "-1", "msg": "签名错误"}
```

同时检查 HTTP 状态、JSON 结构和 code。当前 JSON 可能标注 text/html，不要只按 Content-Type 判断，也不要吧下单成功当作实际付款成功。

06 / QUERY

查询与订单号对应关系

POST /?c=Pay&a=query

发送 mch_id、out_order_sn、time，计算这三个字段的签名后加入 sign。out_order_sn 填下单时的商户订单号。

业务含义	下单请求	下单响应	查询请求	查询响应	通知
商户订单号	order_sn	—	out_order_sn	data.out_order_sn	sh_order
平台订单号	—	data.order_sn	—	data.orders[].order_sn	pt_order

```
{  
  "code": "1", "msg": "ok",  
  "data": {  
    "mch_id": "demo", "out_order_sn": "MY_ORDER_001",  
    "orders": [{  
      "order_sn": "2026101700010012", "money": "10000.00", "pay_money": "10000.00",  
      "order_time": "1792195200", "pay_time": "1792195260",  
      "status": "9", "status_flag": "已支付",  
      "notice_status": "4", "notice_status_flag": "回复成功", "notice_msg": "success"  
    }]  
  }  
}
```

当前返回 data.orders[] 数组。订单字段位于 data.orders[] 的每个元素内，不直接位于 data。统一类型后比较 String(order.status) === "9"，不要将 code=1 当作付款成功。生产可能返回多个子订单，需按商户业务规则

逐单关联。

付款成功只看订单 `status=9`，其他中间/失败状态按平台配置处理。通知状态是投递结果，与付款状态分开：`4` 回复成功、`3` 回复异常、`2` 未成功送达。

超时与重复订单

发送前持久化商户订单号。网络超时可能已经创建成功，应先按原商户订单号查询，不要自动换单号重下。当前生产下单重复检查是全平台范围，建议使用带商户前缀的随机单号；重复提交会报错，不保证返回原订单。

余额查询

`a=querybalance`，签名字段为 `mch_id,time`，返回 `data.mch_id` 和 `data.balance`。共享测试账号余额用于联调，不能当成自己商户的可用资金。

07 / WEBHOOK

支付通知与业务处理

平台向下单时的 `notify_url` POST UTF-8 表单。以下是旧版 v1 字段；未选择 v2 的接入保持不变。数字货币 v1 仅通知最终成功，不通知确认中或超时。

字段	参与签名	说明
mch_id	是	数字商户用户 ID，不是请求中的商户登录账号。示例默认通过商户 key、订单号和金额核对身份；可选 PAY_CALLBACK_MCH_ID 额外校验该数字 ID。
pt_order	是	平台订单号，核对下单 / 查询时保存的单号。
sh_order	是	商户业务订单号。
money	是	订单金额，保持原始字符串验签。
pay_money	是	付款金额，调额场景须按约定处理；示例要求与本地金额一致。
goods_desc	是	商品描述，可能为空。
status	是	success 表示成功，fail 表示失败。
time	是	订单完成时间，秒级 Unix 时间戳；通知可能延迟重试，不套用请求的 300 秒窗口。
sign	否	签名本身。
msg	否	当前服务端在签名后追加的说明字段；不能用它判断资金状态。

v1 通知只对上表八项业务字段验签：`mch_id`、`pt_order`、`sh_order`、`money`、`pay_money`、`goods_desc`、`status`、`time`；排除 `sign` 和 `msg`，保留空字符串。三语言 v1 示例与本规则一致。数字货币 v2 通知使用另一套字

段集合：所有非 sign 字段（包括 msg）均参与签名，详见[数字货币 v2](#)。不要混用两版规则。

- 1. 解析表单并保留原始字符串，拒绝重复键和结构异常。
- 2. 计算签名并恒定时间比较，失败时不处理业务。
- 3. 核对商户 ID、本地订单号、平台单号、金额和 status。
- 4. 在事务中幂等更新支付状态及相关业务；重复有效通知不可重复入账或发货。
- 5. 持久化成功后响应 HTTP 200，响应体仅为小写 `success`。不能返回 JSON、HTML 或额外文本。

示例对有效重复通知再次返回 `success`；验签、金额或本地订单校验失败返回 HTTP 400 `fail`。本地示例只保存支付状态，不替商户执行业务入账。正式环境应将发货、入账等业务写入同一个数据库事务。

正式接口由平台支付结果触发通知，投递和重试受后台队列配置影响。普通示例默认 notify_url 是平台原有 v1 测试验签入口，不支持 v2，不提供开发者通知记录查询；业务持久化、幂等与 v2 验收必须换成自己的公网 HTTPS 通知服务。

07B / CRYPTO V2

数字货币三状态通知（可选）

2026-10-18 新增。下单增加 `crypto_callback_version=2`，并将该字段加入排序签名，仍使用原网关、商户账号、API Key 和 notify_url。省略或填 1 保持原协议，已有订单不追补 v2 事件。订单号重试必须保持原版本；切换版本或网络时使用新商户订单号。RSA 商户在原加密前的参数内加入此字段。

链	测试网 ptype	主网 ptype
Ethereum	2911 · Sepolia TEST-USDT	2901 · USDT
TRON	2912 · Nile TEST-USDT	2902 · USDT
BSC	2913 · Testnet TEST-USDT	2903 · Binance-Peg USDT

在后台添加/启用对应网络钱包，等待监听就绪后下单。测试网和主网可同时使用；调试成功后配置主网钱包并改新订单的 ptype，重新计算签名。测试币没有实际价值。

公开测试商户 `demo` 已为 2911、2912、2913 各配置一个测试收款钱包（2026-10-18 恢复启用并验证公网下单）。客户端无需先调用添加钱包接口；自己的商户在后台“数字货币收款”配置钱包，等待订阅 active、start_block 非空且监听就绪。下单后以响应中的 network、asset、address 和 amount 为付款依据，不要向历史地址直接转账。

共享测试账号的每个钱包同时只分配给一个订单，多个钱包可轮转。若提示“收款地址暂被占用”，更换金额或订单号不会释放地址，应重试原订单或等待其到期；成功订单也保留占用至原到期时间。若提示“商户未配置已启用的收款钱包”，请在后台添加/启用；若提示“钱包监听尚未就绪”，等待订阅与网络就绪。同一地址重复使用相同金额可能转人工核对，见下述付款与确认规则。

USDT 服务费余额与整单阶梯收费

数字货币 API 的 `money` 一律表示 USDT 数量（测试网为 TEST-USDT），不是人民币或商店原价。平台不进行法币换算；商户在调用前自行计算应付 USDT。付款人支付完整订单数量到商户业务钱包，平台服务费从独立 USDT 服务费余额扣除，不减少链上应付数量。非数字货币业务使用平台法币设置，默认 CNY。

下单前累计已确认收款	整单费率
低于 10,000 USDT	1%
10,000 至低于 100,000 USDT	0.1%
100,000 USDT 及以上	0.05%

累计按商户内部 ID 汇总三条主网成功订单，终身累计；测试网使用独立余额和累计。充值、转出、个人收款、未支付订单和测试回调不计入正式累计。一笔订单只用一个档位：下单时按此前成功累计锁定整单费率，不分段。累计 9,990 USDT 时下一单 100 USDT 收费 1 USDT；成功后后续订单使用 0.1%。首次大额订单仍为 1%。无固定手续费、最低订单手续费或 USDT/CNY 换算，不再配置数字货币商户自定义费率。

下单原子冻结服务费，余额不足拒单且不占用钱包；detected 不扣费、不累计；最终核验成功才结算和累计。取消或超过确认宽限期释放冻结。释放后补单按原费率重新扣费，接受差额则用实际核验数量计算，补单页面可预览费用；余额不足不能补单。金额用精确十进制字符串处理，账本保留 24 位小数。历史人民币手续费订单继续按原快照结算，不重复扣 USDT；历史无费用快照订单不追扣。

`data.crypto.fee` / `data.orders[].crypto.fee` 返回 currency (USDT 或 TEST-USDT)、environment、rate、rule_version、volume_before、estimated_fee、charged_fee、fee_usdt 和 state。状态为 held (冻结)、charged (结算)、released (释放)。旧订单仍可能返回 CNY、fee_cny、usdt_cny、fixed_cny，必须按 currency 区分。v1/v2 商户回调字段和签名规则不变。

使用原余额查询签名 (mch_id、time) 调用 `?c=Pay&a=querybalance`：原 balance 仍为平台法币余额，新增 currency 明确其单位；service_fee_accounts.mainnet / testnet 分别提供 available、frozen、deposited、spent、volume、rate、next_threshold，所有金额均为字符串。不要把原 balance 当作 USDT 服务费余额。

商户在“数字货币 → USDT 服务费账户”选择网络获取专属充值地址。Admin 在“充值管理与币种设置”导入平台控制的地址池、开通充值网络；地址池不足、监听未 active 或网络未就绪时不展示付款地址。充值地址永久归属，不与业务收款钱包共用。detected 显示确认中，finalized 后独立 RPC 核验成功按实际到账数量入账；重复通知和补查不重复入账。无平台充值手续费，付款人承担网络费。BSC 主网指定 Binance-Peg USDT 按 1:1 名义服务费单位入账。

不提供提现、自动退款、归集或人民币与 USDT 互转。已分配地址停用后保留归属并监听，迟到或低于最低充值额进入人工核查；其他未监控代币不自动入账。正式充值需 Admin 配置地址后启用。测试余额不能用于主网。平台法币币种有余额或交易时不能直接修改，历史金额不兑换，所选支付通道仍须支持该法币。

记录展示与通知时机

crypto.payment_status	付款 / 确认状态	v2 通知 status	商户业务动作
awaiting	未付款、未确认	不通知	等待付款
confirming	已检测到付款、未确认	confirming	显示确认中，不入账、不发货
confirmed	已付款、已确认	confirmed	验签、订单核对后，在事务内仅完成一次业务
expired	已超时；是否曾检测到付款看 payment_detected	expired	停止提示继续转账；已付款的转入工核对

三类通知不是每单必发三次，也不承诺固定到达顺序。快速最终确认可直接产生 confirmed；通常为 confirming → confirmed，未付款通常只有 expired。无候选订单的个人收款、错误合约或金额不匹配不产生确认中通知。确认中必须经过平台独立 RPC 核验和订单匹配；finalized 经独立核验后才产生 confirmed。

有效期内上链的转账，在默认 1200 秒确认宽限期内继续等待，倒计时结束不立即发“未付款超时”。宽限期仍未确认，发 expired，reason=confirmation_timeout、payment_detected=1，转人工核对。未检测到付款的到期订单发 payment_timeout。若通知较晚才发现有效期内已经上链的转账，宽限期内可从 expired 恢复 confirming/confirmed；真正超时后上链只走客服 Hash 核验和人工补单，补单通过后会发更高 event_seq 的 confirmed。

v2 字段与签名

POST application/x-www-form-urlencoded; charset=UTF-8 ，所有字段均为字符串。除 sign 外，下面所有字段均参与签名，空字符串也参与；v2 不含 msg。按字段名 ASCII 升序拼接 key=value&，最后追加 key=商户APIKey，计算小写 MD5；先对原始值签名，再做表单编码。保持原始字符串验签，不先转换数值或格式化金额。RSA 商户仍以 crypted 表单字段传递原加密结果，解密后的字段及签名规则相同。

字段	含义
callback_version	固定 "2"
event_id / event_seq	平台生成 cn_ 开头的唯一事件 ID；每单从 1 递增的状态版本。重试不改变 ID、版本或正文业务字段。
status	expired / confirming / confirmed，区分大小写
mch_id / pt_order / sh_order	内部数字商户 ID / 平台单号 / 商户单号；mch_id 不是下单账号，可从下单/查询的 crypto.merchant_id 保存
money / pay_money / goods_desc	原订单金额与描述。数字币数量不是法币金额；人工接受差额时必须另核对 amount_raw。
time / expires_at	事件生成时间 / 订单付款截止时间，Unix 秒。重试保留原 time，不使用请求的 300 秒过期规则拒绝延迟事件。
ptype / network / is_testnet	通道编号 / 网络标识 / "1" 测试网、"0" 主网；必须与本地订单一致
asset / address / decimals	代币合约 / 收款地址 / 精度，必须与下单快照一致
expected_amount_raw / amount_raw	应付 / 已核验转账的最小单位整数数量字符串；未检测到付款时 amount_raw 为空。不要用浮点数比较。
tx_hash / event_index	平台核验的交易 Hash / 日志位置；未检测到付款时空。不是 WalletShield 的 event_id。
payment_detected / confirmed	"0" 或 "1"。曾检测到付款不代表最终成功；只有 status=confirmed 且 confirmed=1 才能完成业务。
reason	payment_timeout / confirmation_timeout / payment_detected / payment_confirmed / manual_confirmed
sign	小写 MD5 签名，不参与自身签名

幂等、重试与乱序

后台“测试回调”例外：管理员或订单所属商户可在“数字货币 → 收款订单”重复发送模拟成功通知。它不代表真实付款，不改变平台订单、余额、手续费或正式通知状态。沿用订单的 v1/v2 协议及验签方式，HTTP 头增加 `X-VNPAY-Test-Callback: 1`。v1 正文保持 `status=success`；v2 为 `status=confirmed`、`reason=test_callback`，使用新的 `event_id`、独立测试序号和 `test:cn_...` 形式的模拟 `tx_hash`，数量按订单应付数量填充。v2 的 `reason` 参与签名，HTTP 测试头不参与签名。商户端仍可能按成功通知入账或发货，因此应使用测试业务订单；若区分测试流，请在验签后将测试事件、序号和业务状态与正式通知隔离，避免测试序号使后续正式通知被当成旧事件。每次点击只发送一次，不自动重试；再次点击是新的测试发送。查询接口始终返回真实订单状态。

1. 验签后，核对商户、两种订单号、通道、网络、合约、地址和数量。不要将测试网通知应用到主网业务。
2. 在同一个数据库事务内保存 `event_id`、规范正文摘要和本单最大 `event_seq`。相同事件重复到达只确认接收；同 ID 内容不同或同版本不同事件应拒绝并报警。
3. 比本地版本小的合法事件只留存并应答，不覆盖当前状态。已 `confirmed` 的订单不会回退；有效重复通知不能重复发货或入账。`expired` 后更高版本的合法确认允许恢复；不要把超时视为永远不可能付款。
4. 只在首次接受 `confirmed` 时完成业务，并与事件去重记录原子提交。人工差额确认需商户自己的处理规则，不能把 `expected_amount_raw` 当实际收到数量。
5. 持久化成功后返回 HTTP 200、纯文本小写 `success`。其余应答、网络失败均重试；默认从 30 秒指数退避，上限每小时一次，任务保留直到成功。请求超时为 20 秒。确认中失败重试不会阻塞之后的最终成功事件。

通知事件和正文在订单状态事务内入库，发送在事务外执行。每个阶段有独立投递状态；原 `notice_status` 仅表示最终成功通知结果，不表示此前超时或确认中通知是否成功。旧 v1 回调不会与同一订单的 v2 回调混发。

数字货币下单响应还包含 `crypto.merchant_id`、`is_testnet`、`address`、`decimals`、`expected_units`，用于保存回调核对快照；`ptype` 来自同一响应的 `data.ptype`。订单查询保留原顶层 `status`；数字货币请同时读取 `data.orders[].crypto` 中的 `payment_status`、`payment_state` (`unpaid/detected/paid`)、`confirmation_state` (`unconfirmed/confirmed`)、`callback_version`、`event_seq`。顶层历史 `status=3` 可能仍在确认宽限期，不能单凭它把确认中的链上付款判为未付款。

可下载经回归测试的 [PHP v2 验签、幂等与乱序处理函数](#)。将返回的 `state` 和 `should_fulfil` 对应业务写入同一个商户数据库事务；此函数本身不提供 HTTP 服务或数据库存储。原普通通道示例保持 v1。新增 [数字货币 v2 三语言完整包](#)，包含 HTTP 接收器、数据库事务、离线通知、真实响应及 `inspect` 命令，可直接进行 v2 闭环调试。

08 / SOURCE EXAMPLES

C# · Python · PHP 对接源码

提供完整客户端、签名函数、订单查询、余额查询和可启动的回调服务。下载包包含运行说明和固定测试向量；现有测试账号与 `key` 已随 `merchant-config.json` 内置，解压即可运行。

[下载全部源码 ZIP](#)[查看内置账号 / 密钥](#)[查看完整运行说明](#)

Python: 3.10+ 标准库；PHP: 8.1+ 与 `cURL`；C#: .NET 8+，无需 NuGet 包。回调服务默认只监听本机，联调时请在前面配置公网 HTTPS 反向代理。

三个示例的命令一致：`selftest`、`balance`、`demo`、`create`、`query ORDER`。先在一个语言示例中创建并保存订单，再由对应示例回调服务处理。

文件锁存储用于单机示范，不能替代生产数据库事务。示例支持普通 MD5 表单接入；生产 RSA、代付与充值不在这三个示例范围内。不要将配置文件放到可公开下载的 Web 目录。

WOOCOMMERCE

WordPress / WooCommerce 支付插件

[下载 WooCommerce 支付插件 ZIP](#) 版本 1.0.0。支持普通通道、USDT 主网与测试网、经典结账、新版区块结账、HPOS 订单存储。插件界面支持中文和英文；下载包不包含商户密钥。

- WordPress 后台 → 插件 → 安装插件 → 上传 ZIP 并启用；先安装并启用 WooCommerce。
- WooCommerce → 设置 → 付款 → VNPAYMENT：填写商户登录账号、商户签名密钥、通道编号；接口网关填 `https://payment.271458.xyz/`。回调地址由插件自动生成，商店必须公网 HTTPS 可达。
- 普通通道填写平台约定的收款币种，必须与商店币种相同，不自动换汇。数字货币填写商店币种代码及固定汇率：**1 USDT = N 商店币种**，例如 CNY、7，7 元订单收 1.00 USDT。美元等币种按相同规则配置；USDT 商店使用 1:1。
- 一次配置一个通道：主网 2901 / 2902 / 2903；测试网 2911 / 2912 / 2913，分别对应 Ethereum、TRON、BSC。修改配置只影响新订单。平台需先启用商户通道、可用钱包，并备足对应环境的 USDT 服务费余额；无需设置数字货币自定义费率。

固定汇率与平台手续费分开：应付 USDT = 商店订单总额 ÷ 插件固定汇率，四舍五入到两位。插件锁定每笔订单的原金额、币种及汇率；平台不执行汇率换算，商户 USDT 服务费余额按统一阶梯规则扣费。商店总额目前仅支持精确到两位小数，换算结果不足 0.01 USDT 会拒绝下单；不提供自动汇率服务。

阶段	WooCommerce 行为
下单成功	保留待付款；显示应付金额、固定汇率与平台付款页/二维码链接。商户订单号和平台订单号可在 WooCommerce 订单详情核对；平台后台也能看到该笔下单。
数字货币 confirming	显示已检测到付款，等待最终确认；订单进入保留/on-hold。
confirmed / 普通通道 success	验签、核对订单字段，并查询平台确认成功后调用 <code>payment_complete</code> ；依商品类型进入处理中或已完成。浏览器跳转不能直接确认付款。
expired	取消未付款订单；后续平台人工补单成功且查询核验通过，可恢复付款。
人工测试回调	v2 签名 <code>reason=test_callback</code> 只写订单备注，不改付款状态。v1 成功通知同样必须通过平台查询核验，人工测试通知不能绕过真实订单状态。

测试网订单默认保持 on-hold，不自动标记已付款；仅在独立测试店铺开启“允许测试网订单标记已付款”，用于验证完整订单及库存流程。该开关不会让人工测试回调触发发货。测试网只转指定测试 USDT；共享钱包占用时查询已有订单或等到期，更换单号/金额不能解除占用。

通知按原字段验签、持久化、事件去重并按订单加锁；按序号处理，晚到旧通知不回退状态。付款状态应用失败保留重试任务。后台订单操作支持“查询并核验付款”。配置服务器每分钟执行 WordPress 到期 cron；自动补查每批最多十笔，正常两分钟一次、失败五分钟后重试，待付/确认中订单最多自动补查七天。低访问量站点需配置系统 cron；客户页面仅查询本地状态。

安装要求：64 位 PHP 8.1+、WordPress 6.6+、WooCommerce 9.0+、InnoDB 和数据库 advisory lock 支持。当前实测版本为 WordPress 7.1.3、WooCommerce 11.2.1、PHP 8.3；其他版本请先在测试站验收。插件不含自动退款、订阅扣款或 RSA 加密；不保管链上私钥。更换商户账号前先处理完旧订单。

验收顺序：独立测试店铺配置 → 测试通道下单 → 查询平台对应订单 → 测试 USDT 付款 → 确认中通知 → 最终通知及查询核验 → 重复/乱序通知检查 → 查看 WooCommerce 和平台后台。人工测试回调仅验连接、签名与订单备注，不能代替真实付款。完整安装与维护说明也包含在 ZIP 的 `readme.txt` 中。

09 / OPTIONAL ENCRYPTION

RSA 可选加密

仅当商户开启 RSA 时使用；当前示例测试商户未启用 RSA。加密规则如下：

- 1. 先计算 MD5 sign，放入完整业务参数对象。
- 2. 将该对象转换为 UTF-8 JSON（中文不转义，等价 PHP `json_encode(params, JSON_UNESCAPED_UNICODE)`），再 Base64 编码。
- 3. 使用平台提供的 PEM 公钥（已启用 RSA 的商户向管理员获取）、**RSA PKCS#1 v1.5** 填充，按 `密钥位数 / 8 - 11` 字节分块加密；拼接每块密文字节后整体 Base64 编码。
- 4. 以表单 `crypted=密文` 提交原业务接口；不要把密文当成签名。
- 5. RSA 开启时，平台通知用商户上传的 PEM 公钥加密；商户用自己的私钥按每块 `密钥位数 / 8` 字节解密，拼接后 Base64 解码、JSON 解析，最后按通知规则验证 MD5。

2048 位公钥对应 245 字节明文块、256 字节密文块。不要混用 OAEP，不要对每块单独 Base64 后拼字符串。私钥永远留在自己服务端。三个下载示例尚不包含 RSA 联调；正式启用前须单独验证密钥、分块和双向通知。

10 / TROUBLESHOOTING

排错与验收

现象	检查与处理
签名错误	先运行 selftest；检查空字段、金额尾零、原始 URL、时间戳及通知 msg 排除规则。发送表单，不是 JSON。
缺少参数	检查字段名、Content-Type 及表单解析；goods_desc 和 client_ip 可以为空，但仍发送。
通道不存在 / 未开通	示例测试商户当前使用 2601；自己的商户使用已开通的大于 900 的通道。1、2、12 等历史编号不适用于当前接口；离线向量中的 12 仅用于算法校验。
HTTP 200 但失败	检查 JSON code/msg；code=1 也只表示本次操作成功。
page 返回空体 / HTML	关闭自动重定向，检查首跳 302 和 Location；HTML 可能来自跟随后的收银台。

现象	检查与处理
下单超时	保留原商户订单号，先查询；不要自动换号或无条件重下。
通知验签失败	按八项签名字段验签，排除 msg；确认测试 / 生产 key 没有混用，保留原始字符串。
收款地址暂被占用	共享钱包同时一笔订单，改金额或单号不会释放。查询原订单或等待到期；不循环重下单。
v2 通知接收失败	不能使用旧默认测试地址或 v1 接收器。使用数字货币包，检查 inspect、订单快照及数据库权限；503 修复后等待重试。
通知拒绝 / 未入账	核对数字商户 ID、本地订单、平台单号、两项金额、数据库事务及纯文本 success。先查询付款状态。
上游 get token fail	表示请求已进入通道认证，但当前 GPay 认证未成功；需平台核对上游账号、凭据或网络，不是让商户盲目换自己的签名算法。

上线前验收

- 固定签名向量一致，中文、空字段、金额尾零及表单编码一致。
- 创建订单、收银台、查询、余额和失败路径符合预期。
- 通知成功、重复投递、错误签名、金额不匹配和未知订单处理正确。
- 超时后能够按原单号查询并补齐平台单号。
- 正式配置、实际通道、RSA 要求、调额规则、结算及通知重试策略已单独确认。

排错提供发生时间及时区、接口、请求方法、脱敏参数、商户单号、HTTP 状态、code/msg 和 Location。不要提供生产密钥、私钥或完整含密钥的签名原串。

11 / VERSION & COMPATIBILITY

更新与兼容说明

- 2026-10-24 · 文档 3.1:** 数字货币按 USDT 计价并使用独立服务费账户，新增整单累计阶梯、自动充值、主测隔离和法币币种设置；旧订单按原快照处理。
- 2026-10-21 · 文档 2.9:** 公开 API 测试商户由 1312222222 改名为 demo，后台密码 demo。内部商户 ID 1001、原订单、通道及 API 签名密钥保持不变。新 API 请求使用 mch_id=demo，登录密码不能代替 API 签名密钥。
- 2026-10-19 · 文档 2.8:** WP 插件安装说明独立为文档及 PDF，通道对接和数字货币文档不再重复收录安装正文。
- 2026-10-19 · 文档 2.7:** 新增 WooCommerce 支付插件下载与配置、固定汇率及回调验收说明；保持现有商户接口协议不变。
- 2026-10-19 · 文档 2.6:** 在线通道 / 数字货币文档与 PDF 统一从同一正文生成，下载不再提供 DOCX。同步当前网关、通道编号、data.orders[] 查询结构、v1 固定八字段验签与 v2 全部非 sign 字段验签，修正历史文档差异。文档同步不改变接口协议。
- 2026-10-19 · 文档 2.5:** 新增数字货币 v2 独立三语言完整示例、SQLite 事务与 outbox、28 组离线通知、测试流隔离、三链真实下单/查询响应、6/18 位精度说明及默认 v1 通知地址能力边界。

2026-10-18 · 文档 2.4: 数字货币新订单必须配置有效费率并有足额人民币余额；新增固定汇率、预扣与退费、手续费查询字段及后台设置，默认 1 USDT = 7 CNY。

2026-10-18 · 文档 2.3: 数字货币新增可选 v2 三状态通知，包含签名、重试、幂等、乱序和确认宽限期规则；新增 PHP 验证函数，旧 v1 接入保持不变。

2026-10-17 · 文档 2.2: 按平台现有操作方式，改为直接使用测试商户 13122222222 及其现有 key、正式网关和 2601 通道。源码包内置配置；新增 demo 自动随机下单并查询，取消额外环境启用开关，数字回调商户 ID 改为可选额外核对。联调后替换账号和 key，保持同一接口协议。

沿用 2.1 对原 DOCX 的修订：查询为 data.orders[]；旧 v1 通知包含数字 mch_id、pay_money、goods_desc，msg 不签名（v2 的 msg 参与签名）；通道编号大于 900；时间偏差 300 秒；重复单号按当前接口全平台检查。补充 pay_url 可能为通道二维码图片链接，不能一律当成 HTML 收银台。

文档版本不等于 API 版本。账号验签、余额查询、上游下单、实际付款、通知和结算分别验收，不能用历史或模拟结果证明当前真实通道可用。

商户开发文档 · payment.271458.xyz · 2026-10-19

共享测试商户仅供联调；业务付款以正确验签及核对后的订单结果为准。