商户 API 接入手册

v0.1 · TEST 联调审阅版

面向接入方后端开发、联调与测试人员。
发布日期：2026 年 11 月 14 日

本手册说明商户如何通过平台创建订单、进入托管收银台、接收付款通知并核对结果。按首发维护分支当前代码与随包商户示例整理；正式接入前须由平台确认部署版本、地址、权限与开通范围。

本版覆盖

创建订单与查询订单；创建、查询及取消收银台会话；API 请求签名；付款通知验签；幂等、重试与重复通知处理；联调验收清单。

使用范围

本版用于 TEST 联调，不代表生产环境已开放。第二家支付通道、商户侧退款 API、转账与结算接口不在本版承诺范围内。付款后的退款目前由平台管理流程处理。

阅读顺序

章节	内容
02—05	准备接入、接口清单、订单字段与收银台
06—07	请求签名规范与 Python 示例
08—10	通知验签、业务确认与异常重试
11	联调验收、交付资料与在线文档

示例中的域名、UUID、商品、客户及密钥均为占位或合成数据。不要将这些示例参数直接用于生产交易。

接入准备与调用顺序

由平台单独提供	用途
API_BASE_URL	各环境 API 根地址；本版不填写未经核实的服务地址
API Key / API Secret	服务端请求身份与签名；Secret 为独立 32 字节密钥的无填充 base64url 编码
siteId 与开通范围	站点 UUID、币种、商品、返回地址及 API 权限
Webhook 配置	通知接收地址、端点标识、独立通知密钥和密钥版本
环境与部署标识	明确 TEST 或 PRODUCTION；生产通知另核对部署 UUID

最短接入流程

1. 商户后端保存自己的订单与商品明细，为本次创建生成并持久保存幂等键。
2. 签名调用创建订单接口，保存返回的平台订单 id。
3. 创建收银台会话，使用首次响应的 Location 将用户浏览器导向收银台。
4. 商户后端接收通知：验签、校验原订单、持久保存。
5. 使用查询订单接口复核当前结果，原子更新付款状态并生成唯一履约任务。

方法与路径	用途 / 权限
POST /v1/payment-orders	创建订单；payments:create
GET /v1/payment-orders/{id}	查询订单；payments:read
POST /v1/checkout-sessions	创建收银台会话；payments:create
GET /v1/checkout-sessions/{id}	查询会话；payments:read
POST /v1/checkout-sessions/{id}/cancel	取消会话；payments:create

路径中的 {id} 为对应资源的完整 UUID。示例接口不带查询参数。API Key 的所属合作商、站点及币种等授权必须同时满足。收银台查询和取消还要求使用创建会话的同一凭据。

创建订单：字段与校验

POST /v1/payment-orders。请求为 UTF-8 JSON，携带签名及 Idempotency-Key。首次创建成功返回 HTTP 201。

字段	要求
merchantOrderNo	商户订单号，1—100 字符；在同一环境与合作商范围内唯一
siteId	平台提供且当前凭据可访问的站点 UUID
amountMinor / currency	正整数最小货币单位金额与币种；GBP 的 1000 最小单位表示 10.00 英镑
returnUrl / cancelUrl	必须使用平台批准的地址；各不超过 2048 字符
customer	必填对象：name 1—100 字符，email 1—254 字符
shipping	必填对象：country 国家代码、city 1—100、postalCode 1—30、line1 1—200 字符
shipping.region / line2	可选；分别不超过 100 / 200 字符。无值时省略或传 null，避免空白字符串
items	1—100 项商品。externalProductId 1—100、name 1—200 字符；quantity 与 unitPriceMinor 按获准商品规则填写

金额必须一致

amountMinor 必须等于各商品 quantity × unitPriceMinor 之和。以整数处理最小货币单位，避免用浮点数直接计算付款金额。币种及商品是否开放，以平台提供的开通配置为准。

三种编号分清楚

merchantOrderNo 是商户自己的订单号；订单响应 id 是平台订单号；checkout 会话 id 是收银台会话编号。三个编号分别保存，不可互换。平台订单号应使用完整 UUID，不用前八位作为接口参数。

> 创建接口的幂等重放可能返回原始 UNPAID 快照。要判断是否已经付款，应读取 GET /v1/payment-orders/{id} 的当前状态。

创建订单：完整请求与响应

```
{
  "merchantOrderNo": "shop-order-20261114-001",
  "siteId": "11111111-1111-4111-8111-111111111111",
  "amountMinor": 1000,
  "currency": "GBP",
  "returnUrl": "https://merchant.example/payment/return",
  "cancelUrl": "https://merchant.example/payment/cancel",
  "customer": {"name": "Test Buyer", "email": "buyer@example.com"},
  "shipping": {
    "country": "GB", "region": null, "city": "London",
    "postalCode": "SW1A 1AA", "line1": "1 Example Street", "line2": null
  },
  "items": [{
    "externalProductId": "SKU-DEMO-001", "name": "Test product",
    "quantity": 2, "unitPriceMinor": 500
  }]
}
```

订单响应示例

```
{
  "id": "22222222-2222-4222-8222-222222222222",
  "merchantOrderNo": "shop-order-20261114-001",
  "siteId": "11111111-1111-4111-8111-111111111111",
  "status": "UNPAID",
  "amountMinor": 1000,
  "currency": "GBP",
  "createdAt": "2026-11-14T08:00:00Z"
}
```

查询订单沿用上述字段结构，返回当前 status。createdAt 为 ISO 8601 时间；示例 Z 表示 UTC。订单响应不包含 environment 字段，商户需根据自身环境配置隔离存储和请求。

幂等重试

首次发送前保存：业务订单号、请求路径、完整请求字节和幂等键。网络超时后保持这些内容不变，仅刷新签名时间戳与 nonce。不能通过新键反复创建同一业务订单。

托管收银台与浏览器跳转

```
POST /v1/checkout-sessions
{"orderId":"22222222-2222-4222-8222-222222222222","ttlSeconds":1800}

HTTP/1.1 201 Created
Location: <checkout-origin>/checkout/<session-id>/exchange?token=<one-time-token>

{
  "id": "33333333-3333-4333-8333-333333333333",
  "orderId": "22222222-2222-4222-8222-222222222222",
  "status": "OPEN",
  "expiresAt": "2026-11-14T08:30:00Z",
  "checkoutUrl": "https://checkout.example/checkout/33333333-3333-4333-8333-333333333333"
}
```

ttlSeconds 可不提供，默认 1800 秒；允许范围 300—3600 秒。创建要求订单未付款、收款开关与相关资格检查均满足。

> 必须保留首次创建响应的 Location，并让用户浏览器访问。JSON 中的 checkoutUrl 没有首次访问令牌，不能代替首次 Location。服务端不要自动跟随、预取或记录包含令牌的地址。

浏览器交换一次性令牌后，服务端通过 303 跳转到会话页并设置访问 Cookie。幂等重放只返回稳定 JSON，不再次发放 Location 或令牌；若首次 Location 丢失，应先查询并联系平台核查，不自动换订单或换键重复创建。

状态 / 操作	解释
OPEN	会话开放；查询时会结合订单状态和到期时间更新
COMPLETED	已关联到付款成功的订单
EXPIRED / CANCELLED	会话过期 / 已取消；不能据此推断发生退款
POST .../{id}/cancel，正文 {}	取消开放会话；已接受付款时会被拒绝（PAYMENT_ALREADY_ACCEPTED）

返回商户页面不等于付款成功。商户必须按照通知验签及服务端查单结果确认付款。过期收银台访问可返回 410 / CHECKOUT_ACCESS_EXPIRED。

API 请求签名规范

算法：HMAC-SHA256。将 API Secret 从无填充 base64url 解码为 32 字节密钥；签名结果再次编码为无填充 base64url。签名原文为以下 10 行，行间仅用 LF，最后一行后没有换行。

```
PAY-HMAC-SHA256
v1
<api_key>
<unix_timestamp_seconds>
<nonce>
<UPPERCASE_HTTP_METHOD>
<canonical_path>
<canonical_query_or_empty>
<idempotency_key_or_empty>
<lowercase_sha256_hex_of_raw_body>
```

请求头	填写方式
X-Pay-Api-Key	平台分配；TEST 使用 pk_test_ 前缀，生产使用 pk_live_ 前缀
X-Pay-Signature-Version	固定 v1
X-Pay-Timestamp	Unix 秒级时间戳；服务器保持时间同步
X-Pay-Nonce	每次发送生成新值；参考示例使用 16 字节随机数的 base64url 编码
X-Pay-Signature	对上述原文计算的 HMAC 值
Idempotency-Key	写请求必带、持久保存，不超过 128 字符；GET 签名中该行留空
Content-Type / Accept	JSON 写请求使用 application/json；Accept 使用 application/json

字节与路径规则

签名的 body 字节必须与实际发送完全一致。GET 使用零字节正文的 SHA-256。JSON 排序不是服务端要求；若客户端选择排序或紧凑序列化，应发送同一份结果，签名后不得重新序列化。

路径按段解码再以 UTF-8 百分号编码，保留非保留字符，百分号十六进制大写；拒绝重复斜线、点段、非法编码及解码后的斜线或反斜线。本手册固定路径直接使用。查询参数按编码后的键和值排序，格式 key=value；“+”不视为空格。无查询参数仍保留空行。

Python 示例：生成请求签名

仅使用标准库，示例负责生成请求正文与请求头，不发起交易。path 使用本手册固定路径或带完整 UUID 的路径，query 为空。将返回的 raw 原样发送，禁用自动重定向以读取首次 Location。

```
import base64, hashlib, hmac, json, secrets, time

def b64url(data):
    return base64.urlsafe_b64encode(data).decode("ascii").rstrip("=")

def decode_key(secret):
    raw = base64.b64decode(secret + "=" * (-len(secret) % 4),
                           altchars=b"-_", validate=True)
    if len(raw) != 32 or b64url(raw) != secret:
        raise ValueError("Invalid secret")
    return raw

def prepare(api_key, api_secret, method, path, payload=None, idem=""):
    # Fixed API paths only. This helper does not accept a query string.
    if "?" in path or "#" in path or not path.startswith("/v1/"):
        raise ValueError("Use a fixed API path")
    method = method.upper()
    if method == "GET":
        if payload is not None or idem:
            raise ValueError("GET uses an empty body and idempotency line")
        raw = b""
    else:
        if not idem or len(idem) > 128:
            raise ValueError("Persist an idempotency key before sending")
        raw = json.dumps(payload, ensure_ascii=False, sort_keys=True,
                         separators=(",", ":"), allow_nan=False).encode("utf-8")
    stamp = str(int(time.time()))
    nonce = b64url(secrets.token_bytes(16))
    canonical = "\n".join([
        "PAY-HMAC-SHA256", "v1", api_key, stamp, nonce, method, path, "",
        idem, hashlib.sha256(raw).hexdigest()
    ]).encode("utf-8")
    signature = b64url(hmac.new(decode_key(api_secret), canonical,
                                hashlib.sha256).digest())
    headers = {
        "Accept": "application/json", "X-Pay-Api-Key": api_key,
        "X-Pay-Signature-Version": "v1", "X-Pay-Timestamp": stamp,
        "X-Pay-Nonce": nonce, "X-Pay-Signature": signature
    }
    if method != "GET":
        headers["Content-Type"] = "application/json"
        headers["Idempotency-Key"] = idem
    return raw, headers
```

完整客户端还需实现：凭据安全保管、持久幂等记录、严格响应解析、超时与退避、凭据轮换、日志脱敏及订单数据库事务。勿将 Secret 放入浏览器、移动端包或 Git。

付款通知：验签协议

通知是平台向商户发起的服务端 HTTP 请求。使用独立 Webhook 密钥；先获取未经改写的原始正文并验签，再解析 JSON。不要把 API 请求的正文哈希规则用于通知。

通知请求头	接收方检查
X-Pay-Signature-Version	固定 v1
X-Pay-Timestamp	Unix 秒；参考接收端允许与当前时间相差不超过 300 秒
X-Pay-Delivery-Id	通知投递 UUID
Idempotency-Key	必须与投递 UUID 相同
X-Pay-Key-Version	正整数；只从平台预先配置的密钥版本中选取
X-Pay-Signature	43 字符无填充 base64url HMAC-SHA256
生产环境附加头	X-Pay-Environment: PRODUCTION；X-Pay-Deployment-Id 与本地配置一致

通知签名原文

```
prefix = (
    "PAY-WEBHOOK-SHA256\nv1\n"
    + timestamp + "\n" + delivery_id + "\n"
).encode("ascii")
signing_bytes = prefix + raw_body
expected = base64url(HMAC_SHA256(webhook_key_bytes, signing_bytes))
```

这里直接拼接原始正文，不是正文的 SHA-256。使用恒定时间比较签名。TEST 接收端按本地配置绑定 TEST；生产须额外核对环境与部署标识。不能通过来信自行选择任意环境或远程下载密钥。

解析前的边界检查

参考接收实现限制正文为 1—65536 字节；请求头名称不区分大小写，拒绝重复的 X-Pay-* / Idempotency-Key 以及含 CR/LF 的值。密钥轮换时，上一个版本和签名须成对且与当前版本不同，至少一组已配置的签名验证通过。

轮换附加头为 X-Pay-Previous-Key-Version 与 X-Pay-Previous-Signature。只接受平台事先配置且仍处于允许窗口的版本。

验签后的订单校验与持久接收

付款成功通知：示例正文

```
{
  "id": "44444444-4444-4444-8444-444444444444",
  "type": "payment_order.paid",
  "environment": "TEST",
  "orderId": "22222222-2222-4222-8222-222222222222",
  "siteId": "11111111-1111-4111-8111-111111111111",
  "merchantOrderNo": "shop-order-20261114-001",
  "amountMinor": 1000,
  "currency": "GBP",
  "status": "PAID"
}
```

id 必须等于通知头的投递编号；type 必须是 payment_order.paid；environment 匹配本地配置；订单编号、站点、商户单号、金额、币种必须与本地已保存的原订单一致。金额应为正整数，拒绝布尔值冒充整数。

接收端处理顺序

1. 检查大小和请求头，验证时间戳及签名。
2. 严格解析 JSON，并核对原订单。
3. 在数据库中持久保存原通知及处理任务；成功后返回 HTTP 204。
4. 异步查询平台订单，复核当前 PAID、id、siteId、merchantOrderNo、amountMinor、currency。
5. 在同一事务中标记订单已付款，并为该订单建立唯一履约任务。

> 通知去重与业务去重都要做：同一投递重复到达不重复入库；不同投递指向同一订单，也只能触发一次履约。查询响应不含 environment，环境从本地调用配置确定。

通知持久记录的唯一键应包含环境、部署/端点及 deliveryId。同编号仅接受相同正文的重复请求，并返回 204。无效签名或正文应拒绝；持久保存失败应返回 5xx，不能提前确认接收。服务重启后应能继续处理未完成任务。

超时、冲突与结果不确定

场景	接入方处理
网络超时 / 响应无法解析	可能已提交。保留原订单、路径、原始请求和幂等键；查询已知订单，不自动新建
相同业务写请求重试	使用原幂等键与原请求字节，每次生成新时间戳、nonce 和签名
HTTP 401 / 403	核对时间、环境、凭据、权限及站点授权；避免盲目循环重试
HTTP 409	检查幂等指纹冲突、订单唯一性或状态/资格限制；不要换键绕过
HTTP 429 / 503	尊重 Retry-After，设置重试上限及退避；仍保留原业务请求身份
首次收银台跳转凭据丢失	查询现有会话并联系平台核查；幂等重放不重新签发访问令牌
通知已到但查单暂未确认	保留通知与待核对任务，延后服务端复核，暂不履约
用户返回成功页 / 取消页	只用于页面展示；最终付款状态依赖服务端核对

持久幂等记录建议

记录业务引用、API 地址与环境、凭据标识、方法、路径、请求正文和幂等键。单机示例可用 SQLite；多实例商户应使用共享持久数据库并发控制。请求指纹涵盖方法、路径、查询串和正文哈希，变更正文可能引起冲突。

日志与支持信息

记录完整平台订单号、商户订单号、会话号、通知投递号、请求时间、HTTP 状态及已脱敏的错误。不要记录密钥、完整访问令牌或不必要的客户数据。遇到结果不确定，向平台提交上述编号与时间进行核查。

本版未承诺统一错误 JSON 字段结构及固定通知重试时刻；接入方应按 HTTP 状态分类，并保留脱敏响应供联调核对。

联调验收与交付说明

验收项	预期结果
正常付款	原订单、通知、查询结果的编号、金额、币种一致；只履约一次
未付款返回商户页	订单仍未确认付款，不能误发货
创建请求超时后重试	原键原正文得到同一业务结果，不新增重复订单
同键不同正文	被拒绝或报告冲突，不能静默覆盖原订单
通知重复 / 服务重启	有效通知可重放确认，业务只处理一次，未完成任务能恢复
签名、金额、站点被篡改	拒绝或隔离处理，不标记付款成功
收银台到期 / 取消	页面与会话状态一致；不能当作退款成功
TEST / 生产隔离	密钥、端点、数据和配置分别管理，跨环境请求被拒绝

平台向接入方交付的清单

本手册；环境地址；siteId；批准的币种、商品与返回地址；API 权限；独立安全渠道交付的 API / Webhook 密钥与版本；联调联系人和问题反馈方式。正式开放前共同确认实际部署版本及接入验收结果。

后续在线文档

保留同源 HTML 文档以便部署到固定 HTTPS 地址，后续可增加版本导航、更新记录及可复制示例。对外链接应以实际部署并验证后的地址为准；本次仅交付静态文件，不创建公开服务或发布测试密钥。

版本记录

v0.1 / 2026-11-14：第一版 TEST 商户对接手册。依据首发维护分支 115692b 的接口实现及 release-kit-118 商户 Python 参考示例核对。V1 尚未开放的能力不并入本版。

> 接入成功的判断标准：收到并持久保存有效通知，服务端复核原订单一致，并以原子、可去重的业务事务完成付款确认。