LARAVEL EVIDENCE PLATFORM · 正式合作方文档

存证中台·合作平台开放接口开发文档

面向合作平台后端开发者，提供从服务端签名、存证受理、订单跟踪到回调确认的完整对接规范。本文只定义合作平台与存证中台之间的现行协议。

版本 v1.0 | 正式发布 | HMAC-SHA256 | 服务端集成

文档日期
2026-11-28

目标读者
合作平台后端开发者、联调与运维人员

正式 Base URL
`https://api.zjzxjf.cn`

接口路径前缀
`/open/v1`（签名时必须包含）

01 快速开始与开通资料

合作平台仅需使用中台签发的接入凭证，通过 HTTPS 从自己的服务端调用接口。**无需申请、持有或管理人民链凭证**，链上提交与上游交互由存证中台负责。

凭证边界

`SecretKey` 与 `CallbackSecret` 只能保存在合作方服务端的安全配置或密钥管理系统中，不得下发至浏览器、App、小程序，不得写入前端代码、公开仓库或业务日志。示例中的 `AK_REPLACE_ME`、`SK_REPLACE_ME` 均为占位符。

1.1 开通时准备的资料

按运营方开通流程提交以下资料；是否启用出口 IP 白名单、具体权限与 QPS 以实际开通结果为准：

- 合作平台名称、业务场景、对接环境及预计调用量；
- 技术联系人、运维联系人及有效联系方式；
- 服务端固定出口 IP（如启用白名单）、回调接收 URL；
- 需要的接口权限、文件类型范围与联调时间窗口。

开通后妥善接收 `AccessKey`、`SecretKey`；启用回调时同时接收独立的 `CallbackSecret`。回调验签密钥不得替代 API 签名密钥。

1.2 最小接入路径

1. **读取凭证。**从服务端环境变量或密钥系统读取 AK/SK，准备以毫秒为单位的可信时钟与唯一 nonce 生成器。
2. **验证连通性。**调用无需业务签名的 `GET /open/v1/ping`，确认域名、TLS 与网络路由可用。
3. **实现签名。**严格按本文构造 BodyHash、Canonical 与 `X-Signature`，先完成固定样本的自比对。
4. **提交一单。**使用全局唯一 `bizNo` 调用文本或文件存证；收到 `code:0` 表示已受理，不表示已经上链。
5. **跟踪终态。**结合订单查询与回调处理状态；完成验签、eventId 持久化幂等和标准 ACK 后再上线。

02 架构与调用流程

合作平台后端是签名请求的唯一发起端。中台先完成认证、业务校验和订单受理，再异步推进上链；合作平台通过订单接口与回调获取后续结果。

合作平台服务端

生成业务单号、保存原始请求、计算签名、调用 API、持久化订单和处理回调。

存证中台

认证与防重放、校验与受理、在途订单管理、上游提交、状态查询、证书及回调投递。

上游存证服务

由中台封装；合作方不直接接入，也无需人民链凭证。

2.1 请求与订单生命周期

1. **构造实际载荷。**JSON 固定最终字节；multipart 固定文件顺序和普通字段；GET 或无 body 请求使用空串哈希。

2. **计算 BodyHash。**按请求类型采用本文第 05 章算法，不以框架重新编码后的内容代替。

3. **生成签名头。**取得 13 位毫秒时间戳和唯一 nonce，以 `METHOD + PATH + timestamp + nonce + bodyHash` 的五行 Canonical 计算 HMAC。

4. **发送并保存关联信息。**请求路径包含 `/open/v1`，但不带域名或 query；保存 `requestId`、`bizNo` 和原始订单响应。

5. **受理后异步处理。** `accepted` 只说明订单被接受。状态进入 `CONFIRMED` 才代表成功，`FAILED_FATAL` 才代表明确失败。

6. **收敛不确定状态。** `UNKNOWN` 或 `QUERY_EXHAUSTED` 必须继续查询、刷新或转人工核查，不得直接换 `bizNo` 重发。

> **ping 与 assets 例外：** `GET /open/v1/ping` 和静态 `assets` 资源无需业务签名；除这两类外，本文业务接口均执行完整 AK/SK 认证。

03 AK/SK 认证

除 `GET /open/v1/ping` 与静态 `assets` 资源外，所有业务接口必须携带以下四个签名请求头。凭证只用于服务端对服务端的 HTTPS 请求。

请求头	必填	规范	说明
`X-Access-Key`	是	中台签发的 AccessKey	标识合作方身份，不等于 SecretKey。
`X-Timestamp`	是	13 位毫秒 Unix 时间戳	例如 PHP 中由毫秒整数生成；服务端允许与当前时间相差 ±5 分钟。
`X-Nonce`	是	本次请求唯一随机串	5 分钟窗口内用于防重放；同一合作方不得复用。
`X-Signature`	是	64 位小写 hex	使用 SecretKey 对 Canonical 计算 HMAC-SHA256。

3.1 时间戳与 nonce

PHP 请求头生成示例

```
$accessKey = 'AK_REPLACE_ME';
$secretKey = 'SK_REPLACE_ME'; // 仅从服务端安全配置读取

$timestamp = (string) (int) floor(microtime(true) * 1000); // 13 位毫秒
$nonce = bin2hex(random_bytes(16));                        // 每次请求重新生成
$signature = hash_hmac('sha256', $canonical, $secretKey); // 小写 hex

$headers = [
    'X-Access-Key: ' . $accessKey,
    'X-Timestamp: ' . $timestamp,
    'X-Nonce: ' . $nonce,
    'X-Signature: ' . $signature,
];
```

- 服务器应使用可信 NTP 校时；不要使用秒级 `time()` 直接代替毫秒时间戳。
- nonce 必须不可预测且每次请求重新生成，即使重试同一业务订单也应生成新的请求签名。
- 时间戳偏差超过 ±5 分钟，或 nonce 在 5 分钟防重放窗口内重复，请求会被拒绝。
- 签名验证失败按认证失败处理，不要通过降低时间或 nonce 校验来绕过。

04 签名 Canonical

Canonical 是由五个部分按固定顺序组成的字符串，四处分隔符均为单个 LF（`"\n"`）。**路径包含 `/open/v1` 前缀，不包含域名和 query。**

规范定义

```
Canonical = strtoupper(METHOD) + "\n"
          + PATH + "\n"
          + timestamp + "\n"
          + nonce + "\n"
          + bodyHash

signature = hash_hmac('sha256', Canonical, SecretKey)
```

组成	精确规则	常见错误
METHOD	转为大写，如 `POST`、`GET`。	使用小写方法名或附加空格。
PATH	实际请求路径，含 `/open/v1`；不含 scheme、域名、`?` 与 query。	把完整 URL 或查询串放入 Canonical。
timestamp	与 `X-Timestamp` 完全一致的 13 位毫秒字符串。	签名和请求头使用不同时间戳。
nonce	与 `X-Nonce` 完全一致。	重试时复用 nonce，或计算后重新生成。
bodyHash	按 JSON、GET/无 body、multipart 三类规则得到的 64 位小写 hex。	哈希规范化后的内容而非实际协议规定内容。

PHP 拼接

```
$canonical = strtoupper($method) . "\n"
    . $path . "\n"              // 例：/open/v1/evidence/text
    . $timestamp . "\n"
    . $nonce . "\n"
    . $bodyHash;                // 末尾不再追加换行

$signature = hash_hmac('sha256', $canonical, $secretKey);
```

Canonical 视觉示例（实际值替换占位符）

```
POST
/open/v1/evidence/text
1764321000123
4f21c9d42f774fd6b1b1a5ce2f6f97c1
BODY_HASH_64_LOWERCASE_HEX
```

> **逐字节一致**：方法、路径、时间戳、nonce、bodyHash 与 SecretKey 任一变化都会改变签名。不要对 Canonical 整体做 URL 编码、Base64、trim 或额外换行；签名结果直接使用 `hash_hmac` 默认返回的小写 hex。

4.1 带 query 的 GET

请求 `GET /open/v1/evidence?status=CONFIRMED&from=START` 时，Canonical 的 PATH 仍仅为 `/open/v1/evidence`。query 参数仍应正确放入实际请求 URL，但不参与 Canonical，也不改变空 body 的哈希规则。

05 JSON / GET / multipart 三种 BodyHash

BodyHash 是签名最容易出现分歧的部分。应先确定请求类型，再使用对应算法，不能对 multipart 的完整网络报文直接求 SHA-256。

5.1 JSON：原始 body 字节

```
$bodyHash = hash('sha256', $rawBodyBytes);
```

`$rawBodyBytes` 必须是**实际发送的原始 JSON body 字节**。完成哈希后不得重新 `json_encode`、调整缩进、增删空格、改变 Unicode/斜杠转义或更换字段顺序。推荐先生成一次 body，再把同一个字符串同时用于签名和传输。

```
$rawBody = json_encode($payload, JSON_UNESCAPED_UNICODE);
$bodyHash = hash('sha256', $rawBody);

// 签名与 HTTP 客户端均使用同一个 $rawBody，禁止再次编码。
```

5.2 GET 或无 body：SHA-256 空串

```
$bodyHash = hash('sha256', '');
// e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

GET 的 query 不作为 body；无 body 的 POST 同样使用空串哈希。不要发送 `{}` 后仍使用空串哈希，因为 `{}` 本身已经是非空 body。

5.3 multipart：文件指纹连接 + 普通字段规范化 JSON

设上传文件严格按照 `files[]` 的实际顺序为 `file1 … fileN`，普通字段规范化结果为 `fieldsJson`，则：

```
bodyHash = sha256(
    hash('sha256', file1Bytes)
    . hash('sha256', file2Bytes)
    . ...
    . '|'
    . fieldsJson
)
```

每个文件指纹是 `hash('sha256', $bytes)` 默认返回的**原始小写 hex 字符串**，按上传顺序直接连接。这里不是连接文件内容，也不是连接 multipart 边界；不要 Base64，不要 `hex2bin`，不要在文件指纹之间插入额外分隔符。

PHP 精确实现

```
// $filesInUploadOrder 按 files[] 实际上传顺序排列，每项为文件原始字节。
$fileFingerprints = '';
foreach ($filesInUploadOrder as $bytes) {
    $fileFingerprints .= hash('sha256', $bytes); // 小写 hex 原文，不 hex2bin
}

// 当前接口仅发送 bizNo、dataName 两个普通字段；认证信息只在请求头。
$fields = [
    'dataName' => $dataName,
    'bizNo' => $bizNo,
];
ksort($fields);
$fieldsJson = json_encode($fields, JSON_UNESCAPED_UNICODE);
// 只加 JSON_UNESCAPED_UNICODE；不要添加 JSON_UNESCAPED_SLASHES。

$bodyHash = hash('sha256', $fileFingerprints . '|' . $fieldsJson);
```

> **multipart 四项约束**
>
> - 文件顺序必须与最终 `files[]` 上传顺序一致。
> - 认证信息只放在请求头，不放入表单；当前接口普通字段只发送 `bizNo`、`dataName`，所有实际发送的非文件字段都参与规范化并执行 `ksort`。
> - 字段 JSON 使用 `json_encode(..., JSON_UNESCAPED_UNICODE)`，不得添加不转义斜杠的 flag。
> - `|` 只位于全部文件指纹与 fieldsJson 之间；bodyHash 最终再对连接后的字符串计算 SHA-256。

06 PHP 完整客户端使用

示例 SDK 打包为 附录 A 的 ZIP 下载包，包含 PHP 客户端、回调接收示例和可直接运行的示例脚本。它是帮助合作平台减少重复编码的**可选辅助代码，不是对接前提**；合作平台也可以按本页协议用任意语言自行实现。正文只保留签名与验签的关键片段，完整实现请以 ZIP 为准。

> 以下凭证只是占位符。生产环境应由服务端环境变量或密钥系统注入，禁止把真实 SecretKey 写入本文、代码仓库或前端。

初始化与文本存证

```
<?php
require __DIR__ . '/EvidenceClient.php';

$client = new EvidenceClient(
    'https://api.zjzxjf.cn',
    'AK_REPLACE_ME',
    'SK_REPLACE_ME'
);

$accepted = $client->submitText(
    'PARTNER202611280001',
    '合作协议关键条款',
    '需要固化保存的业务原文'
);

// 客户端返回 httpStatus、json、rawBody 三个键。
$bizNo = $accepted['json']['data']['bizNo'];
```

文件存证：数组顺序即 files[] 上传顺序

```
$accepted = $client->submitFiles(
    'PARTNER202611280002',
    '协议及附件',
    [
        __DIR__ . '/agreement.pdf',
        __DIR__ . '/appendix.pdf',
    ]
);
```

订单、刷新与证书

```
$order = $client→getOrder($bizNo);
$refreshed = $client→refreshOrder($bizNo);
$certificate = $client→certificate($bizNo);
```

6.1 客户端调用约定

方法	用途	关键输入
`submitText($bizNo, $dataName, $content)`	提交文本存证	三个位置参数依次为业务单号、名称、原文
`submitFiles($bizNo, $dataName, $paths)`	提交一个或多个文件	业务单号、名称与按顺序排列的本地文件路径数组
`getOrder($bizNo)`	查询完整订单	原业务单号
`refreshOrder($bizNo)`	主动刷新订单	原业务单号
`certificate($bizNo)`	获取订单证书	原业务单号

客户端运行要求为 PHP 8.2 与 cURL 扩展。每次调用返回 `httpStatus`、`json`、`rawBody`；业务响应位于 `json` 中。列表和 callbacks 查询按第 09 章的 GET 签名规则接入。客户端只负责协议封装，不改变业务语义：HTTP 200 且 `code:0` 仍为受理；业务代码、订单状态和重复标志必须继续按本文处理。每次网络重试应重新生成 timestamp、nonce 和签名，但沿用相同 bizNo 与相同业务内容。

07 文本存证

POST `/open/v1/evidence/text` JSON · 需签名

将一段业务原文作为一个存证订单提交。Content-Type 使用 `application/json; charset=UTF-8`；实际发送的 body 必须与参与 BodyHash 的原始字节完全一致。

7.1 请求字段

字段	类型	必填	约束与说明
`bizNo`	string	是	1–64 位，仅允许 `/^[A-Za-z0-9_-]+$/`；合作方侧唯一且作为幂等键。
`dataName`	string	是	1–128 个字符，用于标识存证内容名称。
`content`	string	是	待存证字符串，最大 `1048576` 字节；按实际 UTF-8 字节数计算。

请求 body

```
{
  "bizNo": "PARTNER202611280001",
  "dataName": "合作协议关键条款",
  "content": "需要固化保存的业务原文"
}
```

PHP 中可用 `strlen($content)` 检查 UTF-8 字符串字节数，不要用字符数代替字节限制。`bizNo` 建议使用可追踪且不可预测碰撞的业务唯一编号，不要在其中放入空格、斜杠或中文。

7.2 受理响应

成功受理返回 **HTTP 200**，响应结构如下：

```
{
  "code": 0,
  "message": "accepted",
  "requestId": "req_01JCD8M6K7M4M3N2Q8W0X4E7J3",
  "data": {
    "bizNo": "PARTNER202611280001",
    "status": "PENDING",
    "serialNo": "SERIAL_NO",
    "duplicated": false,
    "acceptedAt": "2026-11-28T10:30:00+08:00"
  }
}
```

字段	含义
`requestId`	本次 API 请求追踪号，联调与排障时提供。
`data.bizNo`	订单对应的合作方业务单号。
`data.status`	当前订单状态；受理时仍属于处理中。
`data.serialNo`	中台订单流水标识，按响应原值保存。
`data.duplicated`	是否命中已有 bizNo；相同 bizNo 返回原订单时为 `true`。
`data.acceptedAt`	订单受理时间，不代表链上确认时间。

> **accepted ≠ 上链成功。**只有订单状态为 `CONFIRMED` 才能按成功处理；受理响应之后仍须查询或等待回调。

08 文件存证

POST `/open/v1/evidence/file` multipart/form-data · 需签名

一个文件存证订单可携带 1–10 个文件。普通字段与文件字段分开传输，文件不做 Base64 包装；BodyHash 严格采用第 05 章的 multipart 专用算法。

8.1 表单字段与限制

字段	类型	必填	约束与说明
`bizNo`	普通字段 string	是	1–64 位，匹配 `/^[A-Za-z0-9_-]+$/`。
`dataName`	普通字段 string	是	1–128 个字符。
`files[]`	文件数组	是	至少 1 个、最多 10 个；每个文件非空且最大 `10240 KB`（10 MiB）。

使用附录 A 客户端提交

```
$accepted = $client→submitFiles(
    'PARTNER202611280002',
    '协议及附件',
    [__DIR__ . '/agreement.pdf', __DIR__ . '/appendix.pdf']
);
```

- 客户端必须按文件路径数组顺序生成文件指纹，并以相同顺序写入 `files[]`。
- 认证信息只放在请求头；普通字段仅发送 `bizNo`、`dataName`，所有非文件字段执行 `ksort` 后编码。
- HTTP 客户端自动设置带 boundary 的 multipart Content-Type；不要把 boundary 报文或 content-disposition 纳入 BodyHash。
- 业务校验在任何文件为空、超过 10240 KB、总数小于 1 或大于 10 时不得继续按成功处理。

8.2 受理响应

响应外层与文本存证一致：HTTP 200，`{code:0, message:"accepted", requestId, data}`；`data` 包含 `bizNo`、`status`、`serialNo`、`duplicated`、`acceptedAt`。文件数量或大小不改变这一响应语义，也不会使内部计费按文件倍增。

09 订单查询、列表、刷新与证书

以下接口均需 AK/SK 签名。GET 请求无 body，BodyHash 固定为 SHA-256 空串；刷新接口为无 body POST，同样使用空串哈希。路径参数 `{bizNo}` 必须替换为原业务单号，放入 URL 时按需执行 `rawurlencode`，Canonical 使用实际请求路径。

9.1 查询完整订单

GET `/open/v1/evidence/{bizNo}`

返回指定 `bizNo` 的完整订单。合作方至少应持久化并识别以下核心字段，同时保持对响应中其他订单明细字段的向前兼容：

data 字段	处理要求
`bizNo`	与合作方业务订单关联，不以 requestId 代替。
`status`	按第 11 章状态机分支，不以 HTTP 200 推断链上状态。
`serialNo` / `evidenceNo`	保存中台流水标识与存证编号，用于对账和问题定位。
`dataType` / `dataName`	识别文本或文件订单及业务名称。
`createdAt` / `confirmedAt`	分别表示订单创建时间和确认时间；创建时间不能当作链上确认时间。
`chain` / `certificateUrl`	读取链上明细与临时证书地址；未生成时可能为 `null`。
`files`	文件订单包含 `name`、`sizeBytes`、`md5`、`sha256`、`downloadUrl`。
`callbackStatus` / `error`	跟踪回调状态；存在错误时读取 `error.code` 与 `error.message`。

```
GET /open/v1/evidence/PARTNER202611280001 HTTP/1.1
Host: api.zjzxjf.cn
X-Access-Key: AK_REPLACE_ME
X-Timestamp: 1764321000123
X-Nonce: 4f21c9d42f774fd6b1b1a5ce2f6f97c1
X-Signature: SIGNATURE_64_LOWERCASE_HEX
```

完整订单查询不返回提交回执中的 `duplicated` 与 `acceptedAt`；创建时间字段为 `createdAt`。

9.2 查询订单列表

GET `/open/v1/evidence`

所有参数均可选。query 只放入实际 URL，不参与 Canonical，也不改变空 body 的 BodyHash。

Query 参数	默认值 / 限制	用途
`page`	默认 `1`，小于 1 时按 1 处理	页码。
`pageSize`	默认 `20`，范围 1–100	每页条数。
`status`	第 11 章状态值	按订单状态精确筛选。
`bizNo`	原业务单号	按业务单号精确筛选。
`dataName`	名称片段	按数据名称模糊筛选。
`from` / `to`	日期，使用 `YYYY-MM-DD`	按订单创建日期筛选，分别包含当天起止时间。

```
GET /open/v1/evidence?from=2026-11-01&to=2026-11-
28&status=CONFIRMED&page=1&pageSize=20

// Canonical 的 PATH 仅为：
/open/v1/evidence

// data 返回 total、page、pageSize、items；
// items 包含 bizNo、status、serialNo、dataName、createdAt。
```

9.3 主动刷新订单

POST `/open/v1/evidence/{bizNo}/refresh`

仅 `SUBMITTED` 状态会触发新的查询任务；其他状态返回当前 `bizNo` 和 `status`，message 为“当前状态无需刷新”。同一订单的刷新触发有 10 秒防频繁限制。该请求无 body，Canonical 的 PATH 包含末尾 `/refresh`；触发刷新不代表上链成功。

9.4 获取存证证书

GET `/open/v1/evidence/{bizNo}/certificate`

证书尚未生成或订单不存在时返回业务码 `1004`、HTTP 404。证书可用时响应如下：

```
{
  "code": 0,
  "message": "ok",
  "requestId": "req_01JCD8M6K7M4M3N2Q8W0X4E7J3",
  "data": {
    "url": "https://api.zjzxjf.cn/open/v1/assets/...?expires=...&sign=...",
    "expiresIn": 3600,
    "sha256": "CERTIFICATE_FILE_SHA256"
  }
}
```


实际字段是 `data.url`、`data.expiresIn`、`data.sha256`，**不是** `downloadUrl`。`url` 是临时下载地址，`expiresIn` 当前为 3600 秒；下载后应使用 `sha256` 校验证书文件完整性。

9.5 查询回调投递记录

`GET` `/open/v1/callbacks/{bizNo}`

按原 `bizNo` 查询回调投递记录。`data` 返回 `bizNo`、`callbackStatus`、`tasks`；每个任务包含事件标识、状态、投递次数、最后 HTTP 状态、错误、送达时间及逐次 `attemptsDetail`。该接口使用业务 AK/SK 签名，与回调接收端的 CallbackSecret 是两条不同的认证链路。

9.6 无签名连通性检查

`GET` `/open/v1/ping` 无需业务签名

仅用于确认网络和服务可达。ping 成功不代表 AK/SK、签名、订单权限或上游存证状态已验证。

10 回调接收、验签与 ACK

中台向合作方预先登记的回调 URL 发起 **POST JSON**。合作方必须使用原始 body 验签，按 `eventId` 持久化幂等，并返回传输层与业务层均合格的 ACK。

完整回调接收代码见 附录 A 的 SDK 下载包；下图仅保留验签与幂等的关键片段。

10.1 回调报文与请求头

JSON 顶层字段包含 `eventId`、`eventType`、`seq`、`tenantCode`、`bizNo`、`traceId`、`occurredAt`、`data`：

```
{
  "eventId": "evt_01JCDA6N2K3M7RY0J8B4P5W9T6",
  "eventType": "EVENT_TYPE",
  "seq": 1,
  "tenantCode": "PARTNER_CODE",
  "bizNo": "PARTNER202611280001",
  "traceId": "req_01JCD8M6K7M4M3N2Q8W0X4E7J3",
  "occurredAt": "2026-11-28T10:30:00+08:00",
  "data": {
    "status": "CONFIRMED"
  }
}
```

请求头	规范	处理要求
`X-Event-Id`	事件唯一标识	与顶层 `eventId` 对应，作为持久化幂等键。
`X-Timestamp`	13 位毫秒 Unix 时间戳	参与签名原串；接收示例按 ±5 分钟校验，并保存用于追踪与防重放。
`X-Signature-Version`	固定 `v1`	不支持的版本应拒绝，不得猜测其他算法。
`X-Signature`	HMAC-SHA256 小写 hex	使用 CallbackSecret 验签。
`X-Trace-Id`	回调追踪标识	写入无敏感信息的应用日志，便于双方定位。

10.2 v1 回调签名

```
callbackSignature = HMAC-SHA256(
    X-Timestamp + "." + rawBody,
    CallbackSecret
)

// PHP：
$expected = hash_hmac(
    'sha256',
    $timestamp . '.' . $rawBody,
    $callbackSecret
);
```

`rawBody` 是网络请求到达时的原始字节。必须在 JSON 解码、字段重排、空白清理或重新编码之前计算，否则可能产生不同签名。使用 `hash_equals` 做恒定时间比较。

接收端关键 PHP 片段（完整结构见附录 B）

```
$rawBody = file_get_contents('php://input');
$timestamp = $_SERVER['HTTP_X_TIMESTAMP'] ?? '';
$version = $_SERVER['HTTP_X_SIGNATURE_VERSION'] ?? '';
$provided = $_SERVER['HTTP_X_SIGNATURE'] ?? '';
$eventId = $_SERVER['HTTP_X_EVENT_ID'] ?? '';
$callbackSecret = 'CALLBACK_SECRET_REPLACE_ME'; // 仅服务端保存

$expected = hash_hmac('sha256', $timestamp . '.' . $rawBody,
$callbackSecret);
if ($version !== 'v1' || $timestamp === '' || !hash_equals($expected,
$provided)) {
    http_response_code(401);
    exit;
}

$payload = json_decode($rawBody, true);
// 校验顶层字段、X-Event-Id 与 eventId 对应关系。
// 在数据库对 eventId 建唯一约束，并在事务内完成幂等检查与业务处理。

http_response_code(200);
header('Content-Type: application/json; charset=UTF-8');
echo json_encode(['code' => 0, 'message' => 'ok'], JSON_UNESCAPED_UNICODE);
```

10.3 处理顺序与 ACK 判定

1. **读取并保留 rawBody。**同时读取上表五个回调头，不要在验签前修改报文。
2. **先验签。**检查 `X-Signature-Version: v1`，以 CallbackSecret 验证签名；失败则拒绝，不进入业务处理。
3. **解析并校验事件。**确认顶层 eventId、eventType、bizNo、data 存在且头部与 body 的事件标识一致。
4. **按 eventId 持久化幂等。**使用数据库唯一键或等效持久化机制；重复事件不得重复执行副作用。已成功处理的重复事件可直接返回标准 ACK。
5. **同时满足 ACK 两层条件。**返回任意合适的 HTTP 2xx，并确保 JSON body 为 `{"code":0,"message":"ok"}`。只有 HTTP 200、但缺少该 JSON 或内容不符，均不算 ACK。

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8

{"code":0,"message":"ok"}
```

> **不要提前 ACK。**若业务事务失败，不要返回成功 ACK；否则中台可能停止重试，而合作方又未完成处理。若使用异步队列，应先持久化事件并保证可恢复，再按既定可靠性策略确认。

10.4 重试时间表与查询

```
[0, 60, 300, 900, 3600, 21600, 86400]  // 秒
```

该时间表表示**首次投递 + 6 次重试，共 7 次。**接收端必须在整个周期内保持 eventId 幂等，不能依赖“只投递一次”。最终仍失败按回调最终失败处理，可通过需业务签名的 `GET /open/v1/callbacks/{bizNo}` 查询投递记录并转人工补偿。

11 状态机

订单状态与 API 请求结果是两个维度：API 可以成功受理一个仍处于处理中的订单。合作方业务系统必须分别持久化“请求是否受理”和“订单当前状态”。

状态	分类	含义	合作方动作
`PENDING`	处理中	订单已受理，等待后续处理。	保存订单，等待回调或按策略查询。
`SUBMITTING`	处理中	正在执行上游提交过程。	不得当作成功或失败，不重复提交。
`SUBMITTED`	处理中	已提交但仍未获得最终确认。	继续跟踪，必要时主动刷新。
`CONFIRMED`	成功	订单已确认成功。	完成业务成功流转，按需要获取证书。
`FAILED_FATAL`	明确失败	订单得到明确、致命的失败结果。	停止自动上送，记录原因并按业务规则处理。
`UNKNOWN`	不确定	当前无法确认最终结果。	保持原 bizNo，继续核查或转人工，不得换单号重发。
`QUERY_EXHAUSTED`	不确定	轮询查询已经耗尽，仍不能据此认定失败。	主动刷新、查询回调记录并转人工收敛。

三条不可违反的状态规则

- `accepted` 只代表受理，不等于上链成功。
- 只有 `CONFIRMED` 按成功处理；`PENDING`、`SUBMITTING`、`SUBMITTED` 均是处理中。
- `UNKNOWN` 与 `QUERY_EXHAUSTED` 是不确定，不是 `FAILED_FATAL`；不得直接更换 bizNo 重发。

12 幂等与重试

12.1 bizNo 是业务幂等键

相同 `bizNo` 再次提交时，中台返回原订单并将 `duplicated` 设为 `true`，**不会再次上送。**因此合作方必须在本地建立业务操作与 bizNo 的一对一关联，重试时复用同一 bizNo 和同一业务载荷。

当前提交实现按合作方 + bizNo 查找并直接返回原订单，**不会逐项比较新旧业务载荷。**因此合作方必须自行保证同一 bizNo 永远对应相同业务内容；若本地发现同号不同内容，应停止提交并人工核对。服务端虽已定义 `2001` / HTTP 409 幂等冲突码，但不能依赖当前重复提交逻辑替合作方识别内容差异。

12.2 网络重试与业务重发不同

场景	是否可重试	正确处理
连接失败、响应超时	可有界重试	订单可能已被受理；使用同 bizNo、同业务载荷，重新生成 timestamp、nonce 与签名。
HTTP 429 / 限流	降速后有界重试	遵守返回信息，采用指数退避和随机抖动，避免并发重试风暴。
HTTP 500 或上游暂时失败	视订单状态处理	先查询原订单，避免把请求异常直接等同于未受理。
401 / 422 / 业务校验失败	不盲目重试	修正凭证、签名、时间或参数后再发起；原样重试通常无效。
处理中状态	不重新上送	轮询完整订单、等待回调或调用 refresh。
FAILED_FATAL	不自动重试原流程	明确失败，进入人工或业务补偿流程。
UNKNOWN / QUERY_EXHAUSTED	禁止换 bizNo 重发	沿用原订单刷新、查询、核查，直到结果明确。

API 客户端应限制最大重试次数并记录每次 `requestId`；退避策略由合作方结合 QPS 设置，不要套用回调的 7 次投递时间表。重试签名请求时，业务幂等键不变，但传输层 nonce 必须更新。

12.3 回调幂等独立于订单幂等

提交订单使用 `bizNo` 去重，接收事件使用 `eventId` 去重，两者不可互相替代。一个订单可能对应多次状态事件，同一个事件也可能被多次投递。

13 错误码

调用方应同时检查 HTTP 状态、响应 `code`、`message` 与 `requestId`。下表与当前服务端 `ErrorCode` 定义一致；排查时保留 requestId，但日志中不得记录 SecretKey、完整签名材料或敏感存证内容。

code	HTTP	含义	处理建议
`0`	200	成功 / 受理	提交接口仍须继续跟踪订单状态，不直接等同于上链成功。
`1001`	422	参数无效	检查必填字段、长度、正则、上传状态、files 是否缺失或含空文件。
`1002`	422	大小或数量超限	检查文本是否超过 1 MiB、文件是否超过 10 MiB 或总数是否超过 10。
`1003`	422	文件类型不允许	核对实际开通的文件类型范围。
`1004`	404	资源不存在	核对 bizNo 与所属合作方；证书场景也可能表示证书尚未生成。
`1401`	401	未授权	检查鉴权头是否齐全、AccessKey 是否有效或过期。
`1402`	401	签名或时间戳无效	核对 13 位毫秒时间戳、±5 分钟时间窗、Canonical 与 BodyHash。
`1403`	401	Nonce 重放	每次请求生成新的 nonce，不要复用原签名头。
`1404`	403	合作方已停用	停止调用并联系运营方确认开通状态。
`1405`	403	IP 不在白名单	核对服务端实际出口 IP 与开通白名单。
`2001`	409	幂等冲突（已定义）	如收到该码，核对同 bizNo 的原订单与本次载荷，停止盲目重试。
`4290`	429	限流	降低 QPS；refresh 场景等待 10 秒后再试。
`4291`	429	余额不足	暂停新增提交并联系运营方处理额度，不反复冲击接口。
`5001`	500	内部错误	保存 requestId，先查询原订单，再进行有界重试或联系技术支持。
`5003`	503	上游暂不可用	保留 requestId 并查询原订单，按明确失败或不确定状态分支。

不要混用订单状态和业务错误码： `UNKNOWN`、`QUERY_EXHAUSTED` 是订单状态，不是 3001 / 3004；回调最终失败通过订单的 `callbackStatus` 和 callbacks 投递记录判断，也不是业务码 5001。

即使 HTTP 为 200，也必须读取业务 code 和订单 status；即使 HTTP 非 200，也不应在未查询原订单前断言“从未受理”。

14 安全、QPS 与内部计费

14.1 密钥与数据安全

- **服务端隔离：**SecretKey 仅服务端保存，API 签名只能在合作方后端执行；CallbackSecret 同样仅用于回调接收端。
- **职责分离：**API 使用 AccessKey/SecretKey，回调使用 CallbackSecret；两套凭证不得混用。合作方无需人民链凭证。
- **配置与轮换：**使用环境变量或密钥管理系统，限制读取权限，建立泄露应急与轮换流程；不要硬编码真实凭证。
- **传输与校验：**仅通过正式 HTTPS Base URL 调用，客户端保持 TLS 证书校验；不得以关闭校验解决联调问题。
- **最小日志：**可记录 requestId、bizNo、eventId、X-Trace-Id、状态和耗时；脱敏或省略签名、密钥、rawBody 与敏感存证内容。
- **访问控制：**回调 URL 不应以回调中的业务数据作为可信输入；必须先验签，再做事件和订单关联校验。

14.2 QPS 与限流

具体 QPS、并发和文件吞吐额度以合作方开通配置为准，本文不虚构统一数值。客户端应设置连接/读取超时、最大重试次数和并发上限，使用指数退避与随机抖动。

出现业务码 `4290` 或 HTTP `429` 时，应立即降低速率，不得通过增加并发、换 nonce 或换 bizNo 规避限流。nonce 只解决请求防重放，不改变配额。

14.3 内部计费口径

> **文本与文件都按订单计费**
>
> 一单一价，**不按文件数量、文件大小或文本长度倍增。**一个文件订单即使包含多个文件，也仍按一个订单适用该内部计费口径。

1. **提交时检查。**提交订单时检查余额以及在途占用，避免在途订单超出可用能力。
2. **成功后扣费。**订单进入 `CONFIRMED` 后才实际扣费；受理、PENDING、SUBMITTING、SUBMITTED 均不是实际扣费完成依据。
3. **失败不扣费。**明确失败不实际扣费；不要仅因已提交或已查询就认定产生扣费。
4. **不确定状态人工收敛。**`UNKNOWN` 与 `QUERY_EXHAUSTED` 只有经人工核销为成功后才补扣；未明确成功不得据此补扣。

> **上游口径边界**：上游人民链套餐如何消耗，不属于本文已经验证的事实。本文不将其描述为确定规则，也不以上游套餐消耗推导合作方内部账单。

15 联调检查清单

上线前由开发、测试与运维共同核对。每项均应以实际请求日志、持久化记录和订单结果交叉确认。

15.1 凭证与签名

- 正式 Base URL 为 `https://api.zjzxjf.cn`，SecretKey 与 CallbackSecret 仅在服务端保存，合作方未配置人民链凭证。
- 除 ping/assets 外，所有业务请求均带 X-Access-Key、13 位毫秒 X-Timestamp、X-Nonce、X-Signature。
- 服务器时钟误差在 ±5 分钟内，nonce 每次更新且 5 分钟内不重放。
- Canonical 方法大写，恰好五段、四个 LF，PATH 含 /open/v1、不含域名与 query，末尾无额外换行。
- 签名使用 SecretKey 的 HMAC-SHA256，结果为小写 hex，未做 Base64 或 URL 编码。

15.2 BodyHash 与提交

- JSON 对实际发送的原始 body 字节求 SHA-256，签名后未再次编码或改变任何字节。
- GET 与无 body 请求均使用 SHA-256 空串；GET query 未放入 PATH 或 bodyHash。
- multipart 按 files[] 实际上传顺序连接每个文件 hash('sha256', bytes) 的小写 hex 原文，未使用 Base64/hex2bin。
- multipart 认证信息仅在请求头，普通字段仅发送 bizNo/dataName；所有非文件字段 ksort 后仅使用 JSON_UNESCAPED_UNICODE，文件指纹与 fieldsJson 之间只有一个 |。
- bizNo 为 1–64 位且匹配 /^[A-Za-z0-9_-]+$/，dataName 为 1–128 字符，文本 content 不超过 1048576 字节。
- 文件订单为 1–10 个 files[]，每个非空且不超过 10240 KB；同 bizNo 重试不重复上链且 duplicated=true。

15.3 状态、回调与异常

- 已验证 HTTP 200 + code:0 + accepted 仅按受理处理，CONFIRMED 才按成功处理。
- PENDING/SUBMITTING/SUBMITTED 按处理中，FAILED_FATAL 按明确失败，UNKNOWN/QUERY_EXHAUSTED 按不确定处理。
- 不确定状态不会直接更换 bizNo 重发；refresh、订单查询与 callbacks 查询均已接入。
- 回调在 JSON 解码前使用 timestamp + "." + rawBody 和 CallbackSecret 验签，检查 X-Signature-Version:v1。
- eventId 已建立持久化唯一约束；重复事件不重复产生副作用，X-Trace-Id 已进入脱敏日志。

- ACK 同时满足 HTTP 2xx 与 JSON {code:0,message:ok}；已确认“只有 HTTP 200”不会被当作完整 ACK。
- 已按首次 + 6 次重试共 7 次验证回调幂等，最终失败可查询投递记录并进入人工补偿。
- 401、403、404、409、422、429、500、503 与第 13 章全部业务错误码均有分支；限流采用有界退避，无重试风暴。

15.4 计费与交付

- 内部账单按订单一单一价，提交检查余额 + 在途占用，CONFIRMED 后才实际扣费，失败不扣，UNKNOWN / QUERY_EXHAUSTED 人工核销成功才补扣。
- QPS 与并发按实际开通额度配置，日志保留 requestId/bizNo/eventId 但不泄露密钥或敏感原文。
- 附录 A / B 的代码已复制到合作平台服务端组件目录，占位凭证已替换为安全配置注入。

16 验证结果

16.1 本次实测结果（2026-11-28）

> **结果：全部通过。**生产接口测试直接使用附录 A 的 `EvidenceClient.php`，通过既有订单幂等回放验证，不新增真实存证、不产生新扣费；测试前后订单总数均为 9。

验证项	结果	实测证据
文本 JSON 签名与幂等回放	通过	HTTP 200、code 0、duplicated=true，返回原文本订单。
文件 multipart 签名与幂等回放	通过	HTTP 200、code 0、duplicated=true，原文件数组与普通字段解析正确。
文本订单查询	通过	状态 CONFIRMED，serialNo 与 evidenceNo 均存在。
文件订单查询	通过	状态 CONFIRMED，返回 1 个文件及对应哈希。
证书下载与完整性	通过	使用 data.url 实际下载，文件 583 字节，SHA-256 与 data.sha256 一致。
错误签名	通过	按预期返回 HTTP 401。
过期时间戳	通过	按预期返回 HTTP 401。
Nonce 防重放	通过	首次请求 HTTP 200，相同 nonce 重放返回 HTTP 401。
回调有效通知与重复事件	通过	首次和重复通知均 HTTP 200、ACK code 0，eventId 幂等后仅保存 1 个事件。
回调安全负向测试	通过	错误签名和错误版本均 HTTP 401；请求头与 JSON 的 eventId 不一致时 HTTP 400。

附录 A SDK 与存档下载

以下文件与本站正文为同一版本：**v1.0**，日期 **2026-11-28**。正文包含全部接口协议与关键代码片段；需要完整实现时下载 SDK 包。

文件	内容	适用场景
partner-evidence-sdk-v1.0.zip	EvidenceClient.php、callback-receiver.php、可直接运行的示例脚本与使用说明	PHP 服务端快速接入参考
partner-open-api-guide-v1.0-2026-11-28.pdf	本文在线版 PDF 存档，版本与日期一致	内部评审、归档与离线查阅

A.1 SDK 包内容

```
partner-evidence-sdk-v1.0/
├── EvidenceClient.php          # 可选客户端：三种 BodyHash、Canonical、认证头
├── callback-receiver.php       # 回调接收：验签、eventId 幂等、标准 ACK
├── examples/
│   ├── submit-text.php         # 文本存证示例
│   ├── submit-file.php         # 文件存证示例（multipart）
│   └── query-order.php         # 订单查询与证书获取示例
└── README.md                   # 环境变量、运行方式与注意事项
```

A.2 运行前准备

示例脚本从环境变量读取 `EVIDENCE_BASE_URL`、`EVIDENCE_ACCESS_KEY`、`EVIDENCE_SECRET_KEY`，回调示例读取 `CALLBACK_SECRET`。密钥只在服务端注入，不要写入代码、前端或版本库。

> **SDK 不是协议本身**
>
> ZIP 中的代码是接入参考实现。若合作平台使用 Java、Go、Python 等其他语言，按本站第 03 至第 05 章自行实现签名即可，效果与使用 SDK 完全一致。