仓信AI / Cangxin AI Open API

API v1.0.0 · 接口地址：`https://cangxinai.com/api/v1`

仓信AI / Cangxin AI Open API v1 · 卖家 / ERP 开放接口

供您自己的系统或 ERP（领星、店小秘、马帮等）使用：创建订单和入库预报，查询库存和物流跟踪，接收 Webhook 推送。机器可读的接口规范： /api/v1/openapi.json （OpenAPI 3.1，可导入 Postman / Apifox）。

基础 URL： `https://cangxinai.com/api/v1`

0. ERP 对接要点（领星） / ERP integration (Lingxing) at a glance

A test account is delivered as exactly three values: test domain, appKey and appSecret. Nothing else is needed to call the API.
测试账号只交付三项：测试域名、appKey、appSecret。调用接口不需要其他参数。

领星字段 / Lingxing field	我方对应 / Our value	请求头 / Header
API domain / API 域名	Base URL of the environment, ending in /api/v1 (production, or the separate test environment for a test account) 接口地址，以 /api/v1 结尾（正式环境；测试账号则为独立测试环境的地址）	-
appKey	The seller code (GET /me returns it as key.appkey). Identifies the seller, not a secret 卖家代码（GET /me 返回的 key.appkey），用于识别卖家，不是密钥	`X-WMS-Key`
appSecret	The API key (shown once when created; stored hashed). Sent as a Bearer token, or not sent at all when requests are signed with HMAC-SHA256 (optional) API 密钥（生成时只显示一次，服务器只存哈希）。作为 Bearer 令牌发送；或可选用 HMAC-SHA256 请求签名，此时密钥本身不发送	`Authorization: Bearer <appSecret> | X-WMS-Signature`

```
curl https://cangxinai.com/api/v1/me \
  -H "X-WMS-Key: <appKey>" \
  -H "Authorization: Bearer <appSecret>"
```

请求签名为可选项，见 请求签名. / 请求签名为可选项，见下文"请求签名"。

领星对接功能与接口对应 / Lingxing functions and endpoints

#	领星功能 / Lingxing function	接口 / Endpoints	Webhooks / 推送

1	**1. 基础对接（商品、仓库、库存、订单）**
	Base data: SKUs, warehouses, inventory, orders
		`GET /api/v1/skus`
		`POST /api/v1/skus`
		`PUT /api/v1/skus/{id}`
		`GET /api/v1/warehouses`
		`GET /api/v1/inventory`
		`GET /api/v1/orders`
		`GET /api/v1/orders/{id}`
			-

2	**2. FBA 备货入库（入库预报）**
	FBA stock inbound: inbound notice (ASN) and received quantities
		`POST /api/v1/inbound`
		`GET /api/v1/inbound`
		`GET /api/v1/inbound/{id}`
		相关（仓库发往亚马逊 FBA 的补货单） / Related (FBA replenishment, warehouse → Amazon)：`POST /api/v1/transfers` `GET /api/v1/transfers/{id}`
			`inbound.received`

#	领星功能 / Lingxing function	接口 / Endpoints	Webhooks / 推送

3	**3. 出库发货（推单、跟踪号回传、截单）**
	Outbound: push orders, tracking number back, cancel / intercept
		`POST /api/v1/orders`
		`GET /api/v1/orders/{id}`
		`GET /api/v1/orders/by-external/{external_id}`
		`POST /api/v1/orders/{id}/cancel`
		`POST /api/v1/orders/{id}/hold`
			`order.label_created`
			`order.shipped`

4	**4. 退件入库**
	Returns: announce a return, inspection result, instruction
		`POST /api/v1/returns`
		`GET /api/v1/returns`
		`GET /api/v1/returns/{id}`
		`POST /api/v1/returns/{id}/instruction`
			`return.awaiting_instruction`

跟踪号：仓库买面单后，`GET /orders/{id}` 即返回 `carrier` 和 `tracking_numbers`。 / Tracking numbers are returned as soon as the warehouse buys the label.

1. 身份验证

卖家在卖家门户的 API 页面生成密钥（需登录，只有卖家账号所有者可以生成），交给调用接口的系统。接口按该卖家的权限执行：只能访问该卖家在已签约仓库的数据。

```
curl https://cangxinai.com/api/v1/warehouses \
  -H "Authorization: Bearer <API_KEY>"
```

appkey / appSecret：需要填两个值的 ERP，**appkey** 是卖家代码（`GET /me` 返回的 `key.appkey`，用于识别卖家，不是密钥），**appSecret** 是 API 密钥。使用 Bearer 方式时，也可以在请求头 `X-WMS-Key` 中带上 appkey；它必须与密钥所属卖家一致，否则返回 401 `appkey_mismatch`。

请求签名（可选）

也可以不发送密钥，而对每个请求签名。请求头：`X-WMS-Key`（appkey 或密钥前缀）、`X-WMS-Timestamp`（Unix 秒，与服务器时间相差不超过 300 秒）、`X-WMS-Nonce`（8–64 位 `[A-Za-z0-9_-]`，同一密钥不可重复）和 `X-WMS-Signature`。错误：401 `invalid_signature`、`signature_expired`、`nonce_reused`。

```
import hashlib, hmac, secrets, time

def sign(app_secret: str, method: str, path: str, query: str, body: bytes) -> dict:
    # path and query exactly as sent, e.g. "/api/v1/orders" and "status=pending&page=1"
    timestamp, nonce = str(int(time.time())), secrets.token_hex(16)
    message = "\n".join([method.upper(), path, query, timestamp, nonce,
                         hashlib.sha256(body).hexdigest()])
    key = hashlib.sha256(app_secret.encode()).hexdigest()   # signing key = sha256_hex(appSecret)
    signature = hmac.new(key.encode(), message.encode(), hashlib.sha256).hexdigest()
    return {"X-WMS-Key": "<appkey>", "X-WMS-Timestamp": timestamp,
            "X-WMS-Nonce": nonce, "X-WMS-Signature": "sha256=" + signature}
```

每把密钥还可以设置“只接受签名请求”（此时 Bearer 请求返回 401 `signature_required`）和 IP 白名单（其他地址返回 403 `ip_not_allowed`）。

权限	说明
`orders`	出库订单与 FBA / 转仓单
`inbound`	入库预报（ASN）与 SKU 资料
`stock`	库存与 SKU 查询
`returns`	退件预报与处理指令
`vas`	增值服务申请
`billing:read`	预收余额、流水与费用（只读）

2. 约定

- 请求和响应都是 JSON（UTF-8）。时间为 UTC 的 ISO 8601 格式（`2026-11-10T08:00:00Z`）。
- 列表：`?page=1&page_size=50`（最多 200）→ `{"data": [...], "pagination": {...}}`。

- 错误：`{"error": {"code": "...", "message": "...", "details": {...}}}`，HTTP 状态码 400 / 401 / 403 / 404 / 409 / 422 / 429。访问其他卖家的记录返回 404。
- POST 请求请带上唯一值的`Idempotency-Key`请求头（例如你的订单号）：7 天内用相同的键和请求体重试，会返回第一次的响应（响应头`Idempotent-Replayed: true`），不会重复创建。
- 每个密钥有频率限制（默认每分钟 120 次）：见`X-RateLimit-*`响应头；超限返回 HTTP 429，并带`Retry-After`。
- 加上`Accept-Language: zh-CN`即返回中文错误信息。
- 仓库：`GET /warehouses`返回每个签约仓库及其`public_code`（例如`WS-ACME-LAX1`）：全平台唯一、永不变化。下单时作为`warehouse_code`使用（或发送`warehouse_id`）。短代码`code`只在仓库公司内唯一：两个签约仓库代码相同时，接口返回 422`ambiguous_warehouse_code`。

3. 接口列表

方法	路径	权限	描述
GET	`/api/v1/me`	-	当前密钥、用户与卖家
GET	`/api/v1/warehouses`	-	已签约仓库
GET	`/api/v1/skus`	stock, inbound, orders	SKU 列表
GET	`/api/v1/skus/{id}`	stock, inbound, orders	SKU 详情
POST	`/api/v1/skus`	inbound	新建 SKU
PUT	`/api/v1/skus/{id}`	inbound	修改 SKU（只改传入的字段）
POST	`/api/v1/orders`	orders	创建出库订单
GET	`/api/v1/orders`	orders	出库订单列表
GET	`/api/v1/orders/{id}`	orders	订单状态与运单号
POST	`/api/v1/orders/{id}/cancel`	orders	取消订单（拣货开始前）
POST	`/api/v1/orders/{id}/hold`	orders	申请拦截订单
GET	`/api/v1/orders/by-external/{external_id}`	orders	按你的订单号查询订单状态与运单号
POST	`/api/v1/orders/by-external/{external_id}/cancel`	orders	按你的订单号取消订单（拣货开始前）
POST	`/api/v1/orders/by-external/{external_id}/hold`	orders	按你的订单号拦截订单
POST	`/api/v1/inbound`	inbound	创建入库预报
GET	`/api/v1/inbound`	inbound	入库预报列表
GET	`/api/v1/inbound/{id}`	inbound	入库状态与实收数量
GET	`/api/v1/inventory`	stock	库存（按仓库 / SKU / 批次）
POST	`/api/v1/returns`	returns	退件预报
GET	`/api/v1/returns`	returns	退件列表
GET	`/api/v1/returns/{id}`	returns	退件详情
POST	`/api/v1/returns/{id}/instruction`	returns	下达退件处理指令
GET	`/api/v1/vas/services`	vas	可申请的增值服务
POST	`/api/v1/vas`	vas	申请增值服务
GET	`/api/v1/vas`	vas	增值服务单列表
GET	`/api/v1/vas/{id}`	vas	增值服务单详情

方法	路径	权限	描述
POST	`/api/v1/transfers`	orders	创建 FBA / 转仓单（箱标 PDF 用 base64）
GET	`/api/v1/transfers`	orders	FBA / 转仓单列表
GET	`/api/v1/transfers/{id}`	orders	FBA / 转仓单详情
GET	`/api/v1/wallets`	billing:read	各仓库预收余额
GET	`/api/v1/wallets/{id}/transactions`	billing:read	预收余额流水
GET	`/api/v1/charges`	billing:read	费用明细

创建订单

订单与上传的订单文件走同一套流程：SKU 按 SKU 编码、条码或你的 SKU 映射匹配；同一个 `external_id` 重复提交返回 409。

```
curl -X POST https://cangxinai.com/api/v1/orders \
  -H "Authorization: Bearer <API_KEY>" -H "Content-Type: application/json" \
  -H "Idempotency-Key: 111-1234567-1234567" \
  -d '{"warehouse_code": "WS-ACME-LAX1", "external_id": "111-1234567-1234567", "channel": "amazon",
       "ship_by_date": "2026-11-12T07:59:59Z",
       "recipient": {"name": "Jane Smith", "street": "1 Main St", "city": "Los Angeles",
                     "state": "CA", "zip": "90001", "country": "US", "phone": "555-0100"},
       "lines": [{"sku": "MUG-01", "quantity": 2, "external_item_id": "10000000000001"}]}'
```

之后查询 `GET /orders/{id}` 或 `GET /orders/by-external/{external_id}`（状态 `pending → processing → shipped`，含 `tracking_numbers`），或订阅 `order.label_created` / `order.shipped` 推送。可按你自己的订单号用 `POST /orders/by-external/{external_id}/cancel` 和 `/hold` 取消或拦截（订单号需 URL 编码；含 "/" 的订单号请使用数字 id）。

4. 推送（Webhook）

在卖家门户的 API 页面登记 HTTPS 地址。我们会 POST 一个 JSON 事件，请在 10 秒内返回任意 2xx。失败的推送会在 1、5、15、60 分钟以及 3、6、12、24 小时后重试（共 8 次），推送结果可在 API 页面查看。同一事件可能重复推送：已处理过的 `id` 请忽略。

事件	何时
`order.label_created`	已购买面单：首次生成运单号（订单尚未出库）
`order.shipped`	订单发货（含运单号）
`order.held`	订单被拦截
`inbound.received`	入库已收货
`return.awaiting_instruction`	退件待指令
`wallet.low_balance`	预收余额低于提醒阈值

仓库买面单后立即推送 `order.label_created`（每个订单和跟踪号一次，包裹出库之前）；出库时再推送 `order.shipped`。

```
POST https://erp.example.com/wms/webhook
X-WMS-Event: order.shipped
X-WMS-Timestamp: 1762761600
X-WMS-Signature: sha256=5d41402abc4b2a76b9719d911017c592...

{"id": "evt_...", "type": "order.shipped", "created_at": "2026-11-10T08:00:00Z",
 "seller": {"id": 7, "code": "S001"},
 "data": {"order": {"id": 42, "external_id": "111-1234567-1234567", "status": "shipped",
                    "carrier": "UPS", "tracking_numbers": ["1Z999AA10123456784"], ...}}}
```

校验签名（用端点密钥对 `timestamp + "." + raw body` 做 HMAC-SHA256），并拒绝过旧的时间戳：

```
import hashlib, hmac, time
```

```
def verify(secret: str, headers: dict, raw_body: bytes) -> bool:
    timestamp = headers["X-WMS-Timestamp"]
    if abs(time.time() - int(timestamp)) > 300:
        return False
    expected = hmac.new(secret.encode(), timestamp.encode() + b"." + raw_body,
                        hashlib.sha256).hexdigest()
    return hmac.compare_digest("sha256=" + expected, headers["X-WMS-Signature"])
```