

远程打印接入规范

规范版本： 3.2（零存量基线） | 对应代理： v1.8.0 | 日期： 2026-10-18

本文的适用前提

业务系统里没有任何一台已绑定的打印电脑，没有任何一张待打印单据，没有任何一条打印结果或状态快照，办公室里没有任何一台电脑装过打印代理，终端也没有任何账号需要迁移。

也就是说：这不是一份「升级说明」，而是一份**从空库开始、一直走到第一张纸**的施工图。文中每一个字、每一个错误码、每一个数字，都可以在一套刚建好、还没跑过一单的环境里原样复现。

2.x 及更早的草稿是双方的协商过程稿，已归档，**不再作为接入依据**；与本文冲突时，以本文为准。

3.2 相比 3.1 改了什么（按贵方《外部服务接入要求 v5》2026-10-18 23:10 落地）

项	变化
本机标识	<code>machineId</code> 回到随机 UUID v4 （首次生成即落盘，之后永不变）。不再用主板/系统安装 ID 派生——v5 R2 明确禁止；重装系统按「新电脑」处理，到上限时后台解绑旧记录
凭证存放	<code>deviceToken</code> 加密落盘 ：Windows DPAPI、macOS 钥匙串、Linux 独立文件 0600； <code>config.json</code> 与日志里都不再出现明文，面板只显示前 6 位 + 长度
幂等集合保留期	本地 <code>jobId</code> 记录由「按条数砍」改为 保留 30 天 （v5 R8：服务端不回收旧 <code>jobId</code> ，删早了会把老单重打一遍）
补打入口	业务系统的单据 只在业务后台补打 （ <code>attempt+1</code> → 新 <code>jobId</code> ），代理面板不再对这类单提供「重打」——否则看板「今日出纸」与代理面板次数对不上
凭证被拒自愈	带着 <code>deviceToken</code> 仍被判 <code>no_token</code> 时，按 v5 §2.2「清 token 重绑一次」，不再直接停机
代理版本	v1.8.0

接口数量、字段、错误码**均未变化**（v5 已冻结的 5 个接口）；本次只是代理侧实现口径与文档口径对齐，服务端不需要为 3.2 做任何改动。

0 一页速览

项	结论
接口数量	5 个，共用同一个入口地址，用 ?a= 区分
鉴权	一条 Authorization: Bearer <凭证>；未绑定用授权码，绑定后用 deviceToken
连接方向	只有办公室电脑发起。业务系统不回调、不推送给代理
绑定	一码一机，强制绑定。装好即绑，绑不上就停，不存在「绑不上也照样打」
版式	内容与分页全部由业务系统决定；代理按 HTML 原样出纸，不猜页数、不缩放
幂等键	jobId (orderId#attempt)，每次出纸一个；orderId 必须稳定
错误分流	只看 body.code，不看 HTTP 状态码
验收	接口 22 条 + 真机 8 项 + 异常 8 项，附一键脚本 zerotest.py

接入只有三步：

1. **服务端**：实现 5 个接口（第 4 章），错误码按第 8 章。
2. **后台**：给客户开通模块、生成一份授权码（一客户一码）。
3. **现场**：办公室电脑装代理，填「接口地址 + 授权码」，点「验证授权」——它会自己完成绑定并开始拉单。

按角色选读：

角色	先看	产出
后端开发	3 → 4 → 7 → 8	5 个接口、错误码、队列与幂等
测试 / 联调	9 → 10	19 条用例的执行记录、8 项走查
产品 / 实施	1 → 5 → 11	授权码发放、绑定与换机流程

1 起点：什么叫「零」

1.1 三条基线，开工前逐条确认

维度	开工时应为	怎么确认
客户	0 个正在使用远程打印的客户（可以有测试租户）	后台客户列表里远程打印状态全为「未开通」，或只有一个待开通的测试租户
数据	0 绑定设备、0 待打印单据、0 打印结果、0 状态快照	四张表 <code>count(*)</code> 全为 0（见 3.6）
用户	0 个需要迁移的账号	代理侧不引入账号体系，办公室电脑只需要一条授权码

必须已经存在的只有三样东西：

- 一个已开通（或可开通）远程打印模块的**租户**；
- 一份该租户的**授权码**（后台生成，一客户一码）；
- 办公室里**一台能上网的 Windows 或 macOS 电脑** + **一台针式打印机**（三联复式纸，241 × 140 mm）。

1.2 目标：一次跑通，不留半成品

本节结束时应当达到的状态：

- 后台设备列表里出现 **1 台电脑**，`bound 1 / 1`；
- 司机在手机上开一张单，**10 秒内**办公室出纸；
- 业务系统收到 `print_result ( status=done , pages=1 )`，队列自动清空；
- 看板上这台电脑 `printer.state=idle`、`biz.authMode=device`；
- 抽掉纸张再打一张，单据变 `failed` 且原因写「**缺纸**」，放回纸点「**重试**」能出纸。

1.3 本期明确不做

把边界写死，避免开发时被拉去做无关的事：

不做	原因
历史单据补打、数据迁移	没有存量，不存在这件事
代理回调 / WebSocket / 长连接推送	办公室没有固定公网 IP，链路只能单向（2.2）
云端预渲染并长期存档单据	改单后会打旧金额；渲染放在拉单时（4.3）
手机直连打印机 / 蓝牙打印	手机不参与打印，只负责开单
代理侧账号、密码、员工权限	一码一机已足够；人员变动走「解绑 + 换机」（5.6）
多机负载均衡、按仓库路由	一客户一台电脑是本期唯一形态（11.1 说明第二台怎么处理）

2 系统形态

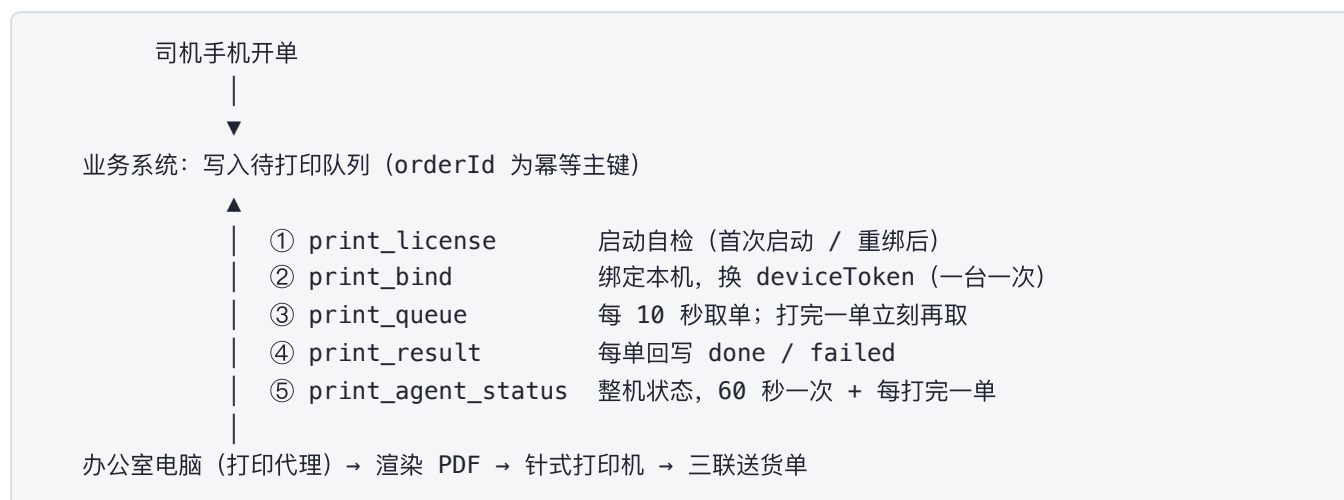
2.1 三个角色

角色	位置	职责
业务系统	公网服务器 / 公司机房	开单、生成单据内容、维护待打印队列、接收打印结果与整机状态
打印代理	办公室电脑 (Windows / macOS)	取单、渲染 PDF、驱动针式打印机、回写结果、上报状态
针式打印机	办公室	三联复写纸出单

版式与数据都归业务系统。代理不做排版决策：给什么 HTML 就打什么，给几页就打几页。

2.2 一条硬约束：只有办公室电脑能发起连接

办公室宽带没有固定公网 IP，路由器也不会做端口映射，因此业务系统**无法主动连到办公室电脑**。整条链路只有一个方向：



四条设计结论，后面所有章节都从这里推出来：

1. **不要给代理留回调地址**，也不要设计「服务端通知代理」的接口——它没有可被连接的入口。
2. **积压由业务系统保管**。办公室电脑关机、断网期间单据留在队列里，代理恢复后按顺序追平，业务系统不需要补偿重发。
3. **实时性 = 轮询间隔**。空闲 10 秒一轮；打完一单立刻再拉一次，连续出单不会一张一张干等。
4. **服务端不需要为「推」做任何幂等之外的额外机制**，重复返回同一张单是安全的（第 7 章）。

2.3 部署与网络要求

项	要求
连通性	办公室电脑能访问接入地址（浏览器打开不报错即可）
协议	http / https 都支持；公网务必用 https
自签证书	内网自签可在代理里勾「跳过 HTTPS 证书校验」；公网不要开
响应时间	每个接口 1 秒内返回为宜；代理超时阈值 8—15 秒（见 3.5）
并发	单台电脑同一时刻只有一个拉单请求；队列写入无需分布式锁
字符集	请求与响应统一 UTF-8
出网白名单	若办公室有防火墙，放行接入地址的 80 / 443 出站即可

3 接入准备

3.1 五个动作

五个动作共用一个入口地址，用查询参数 a 区分。以 https://erp.example.com/api/print.php 为例：

动作	方法	完整地址	用途
print_license	GET	<入口>?a=print_license	授权自检，返回绑定状态
print_bind	POST	<入口>?a=print_bind	授权码换 deviceToken
print_queue	GET	<入口>?a=print_queue&limit=20	取待打印单据
print_result	POST	<入口>?a=print_result	接收打印结果
print_agent_status	POST	<入口>?a=print_agent_status	接收整机状态（看板数据源）

动作名大小写敏感，一律小写；除这五个外，代理不会调用任何其它动作。

3.2 入口地址

- 主站：http://<host>/wb/api.php
- 租户站：http://<租户域名>/api.php（不带 /wb/）

客户在后台复制的「拉单地址」通常带 ?a=print_queue&limit=20。代理会自动截掉查询串，再按动作重新拼接，所以后台给完整链接即可，现场不需要手工裁剪。

3.3 鉴权

所有请求都带同一个头：

Authorization: Bearer <凭证>

凭证有两种，代理按手里有什么用什么：

阶段	用的是	来源
尚未绑定	授权码（一客户一份，示例为 48 位十六进制）	客户后台复制，填进代理
绑定成功之后	deviceToken（示例为 64 位十六进制）	print_bind 的响应，代理自动落盘

长度不是契约。代理对凭证只判「非空字符串」，不校验长度，也不按定长存储（配置是 JSON 字符串，设置页输入框无长度限制）。以后换 token 生成方式、把授权码改成别的长度，都不需要通知我方改代码；真要给个下限，≥ 32 位即可。

两条必须遵守的实现要求：

1. **授权码放 body.code 或放请求头，两种都要能认。**代理的做法是**两个都给**（同一个值），因此对「只读 body」「只读头」「两个都读」三种实现都成立。两者同时出现且不一致时，以 body.code 为准。
2. **异常路径也必须返回 JSON。**返回 HTML 错误页、网关 502 页面时，代理只能提示「接口返回的不是 JSON，确认接入地址是否填对」，现场排查成本极高。

服务端建议的校验顺序：

Bearer 能匹配某个 deviceToken → 按设备身份处理
否则能匹配某个授权码 → 按授权码身份处理
两者都不匹配 → 401 + license_invalid
完全没带凭证 → 401 + no_token

no_token 与 license_invalid **不要混用**：代理面板对「没填码」和「码填错了」显示两句不同的提示，混用会把客户引到错误的排查方向。

3.4 统一响应包络

成功（HTTP 200）：

{ "ok": true }

失败（4xx / 5xx，同时给机器码与中文原因）：

{ "ok": false, "code": "license_invalid", "error": "授权码不正确或已被更换" }

约定：

- code 供代理判断行为（是否停机、是否重绑），**必须稳定**，不要随文案调整而变化；
- error 会原样显示在代理面板与日志里，请写成运营看得懂的中文；
- 代理一律按 body.code 分流，**不看 HTTP 状态码**。所以码值要准，状态码只需语义合理（本规范给出的状态码是建议值）。

- 服务端只回状态码、不给 `code` 时，代理会按状态码停机并提示「服务端未返回错误码」——这属于不完整实现。

3.5 传输约定

项	约定
编码	Content-Type: application/json; charset=utf-8
压缩	代理请求带 Accept-Encoding: gzip 并自行解压（内部上限 16 MB），服务端可放心开启
超时	自检 10 秒、绑定 15 秒、拉单 15 秒、回写 8 秒、上报 10 秒
响应体	单个响应 gzip 后 ≤ 100 KB 为宜；未压缩可到 600 KB 量级（limit=20 × 5–30 KB/单）
幂等	print_result 与 print_bind 必须幂等，不认识的 orderId 也应返回成功
方法	GET 不带 body；POST 必须带 JSON body（空对象也要给 {}）

3.6 空库最小数据结构（建议）

从零开始建表，下面四张表足够支撑全部 5 个接口。字段名可自定，语义要对上。

表	关键字段	说明
print_license	tenant_id、code、licensed、bind_mode、device_limit、paused	一租户一行；code 建唯一索引
print_device	device_id、tenant_id、machine_id、name、os、ver、token_hash、bound_at、last_seen_at、revoked_at	machine_id 唯一；token_hash 唯一
print_order	order_id、attempt、tenant_id、order_no、customer、driver、amount、biz_date、html、status、printed_at、pages、error	order_id 用业务主键；attempt 是出纸次数（首次 1，后台每点一次「重打」+1），job_id = order_id#attempt；status 见 7.4
print_status	device_id、snapshot、updated_at	一设备一行，只存最新快照，不存历史

三条实现建议：

1. 令牌只存哈希（token_hash），另存后 4 位用于后台展示（token_hint）。deviceToken 只在 print_bind 的响应里出现一次，之后任何接口都不再下发。
2. 队列查询走（tenant_id, status, created_at）复合索引，否则单量上来后拉单会变慢。
3. print_status.snapshot 用 JSON 列（MySQL 5.7+ / PostgreSQL 均可），不要为上报字段逐个建列——代理会持续加字段，建列会变成负担。

3.7 代理侧怎么存 deviceToken (v5 R3)

deviceToken 是长期凭证，等价于「这台电脑能拉这个客户的全部单据」。落盘方式按平台区分，**都不落明文**：

平台	存放方式	说明
Windows	DPAPI（当前用户）	换 Windows 账号登录解不开，代理会当成「没有令牌」，用授权码重绑一次
macOS	钥匙串（Keychain）	条目按安装实例区分（服务名 + 配置路径哈希），同机多套代理不互相覆盖
Linux	独立文件 data/device.token + 0600	目录 0700；系统没有等价凭据库时的标准做法

三条必须遵守：

1. config.json 里 biz.deviceToken **永远是空字符串**，令牌只在加密存储里；面板只显示 前 6 位...(长度)。
2. 任何一步加密失败都**退回文件 0600 并继续跑**，绝不因为「加密不可用」把客户搞停产。
3. 解绑（hooks.setDeviceToken('')）必须三处一起清：内存、config.json、加密存储；只清一处会在下次启动被读回来。

4 接口规范

4.1 print_license — 授权自检

什么时候调：代理启动后第一次拉单之前、重新绑定之后、现场点「验证授权」时。**不是**每轮拉单都调。

请求

```
GET <入口>?a=print_license
Authorization: Bearer <deviceToken 或 授权码>
```

响应

```
{
  "ok": true,
  "licensed": true,
  "authMode": "device",
  "needsBind": false,
  "bindMode": true,
  "limit": 1,
  "bound": 1,
  "paused": false,
  "device": { "id": "DEV-8F21", "name": "办公室电脑" },
  "devices": [{ "id": "DEV-8F21", "name": "办公室电脑", "lastSeen": 1792309238000 }],
  "server": { "name": "某某 ERP", "ver": "1.7.0", "time": 1792309239000 }
}
```

字段	类型	必须	说明
ok	bool	是	固定 true
licensed	bool	是	该客户是否开通远程打印。false 时代理立即停止，提示「套餐未开通」
authMode	string	是	device（本次用的是 deviceToken） / license（只有授权码）
needsBind	bool	否	是否要求先绑定才能拉单
bindMode	bool	否	是否强制绑定，见第 5 章
limit	int	否	允许绑定的电脑台数
bound	int	否	已绑定台数（面板显示「绑定 1 / 1」）
paused	bool	否	后台暂停了远程打印。true 时代理不取单，恢复后自动继续
device	object	否	当前这台电脑的绑定信息
devices	array	否	已绑定电脑清单，绑定满时用于提示「先解绑哪台」
server	object	否	仅面板展示。建议给 { "name": "...", "ver": "1.7.0", "time": ... }，ver 会显示成「业务系统 v1.7.0」，联调时双方一眼对齐版本

硬约束

- authMode 必须如实反映本次请求**实际使用的凭证类型**——代理据此决定要不要走绑定流程。谎报会让代理跳过绑定，随后所有拉单都撞 bind_required。
- 不带 Authorization → 401 + no_token，并带响应头 WWW-Authenticate: Bearer。代理在授权码为空时**不会发这个请求**（面板直接提示「还没填授权码」），所以这条主要影响手工 curl 与第三方工具。
- 授权码无效 → 401 + license_invalid；本机已被解绑 → 403 + device_revoked。两个码不要合并：代理对它们的恢复动作相同，但面板文案不同（「授权码无效」vs「授权已解除」），排障时能一眼分清是码填错还是被后台解绑了。
- 这个接口会被现场在排障时频繁手动触发，**不要在这里渲染单据或做重查询。**

4.2 print_bind — 绑定本机

为什么要有绑定： 把「一个客户一份授权码」变成「一台电脑一份凭证」，才能做到后台看清是哪台电脑在打、限制台数、单独撤销某一台。

请求

```
POST <入口>?a=print_bind
Authorization: Bearer <授权码>
Content-Type: application/json; charset=utf-8

{
  "code": "48 位授权码",
  "machineId": "9f2c1e40-7b3a-4d55-9a11-2c6e8f0b1a77",
  "name": "办公室电脑",
  "os": "Windows 10.0.19045",
  "ver": "1.8.0"
}
```

字段	说明
code	授权码，与请求头里的值一致（两个位置都要能读，见 3.3）
machineId	随机 UUID v4 （36 字符，含 4 个连字符），首次启动生成、写进配置文件，之后永不改变。不用 MAC、主机名、CPU 序列号、安装路径派生——换网卡、改电脑名本来是同一台机器，用这些派生会被判成「第二台」，一码一机（limit=1）直接把客户卡死。服务端只需按字符串比较，长度 ≤ 80、字符集 A-Za-z0-9_-
name	现场填写的本机名称，缺省为主机名
os / ver	操作系统与代理版本，用于后台展示与排障

响应

```
{
  "ok": true,
  "deviceToken": "64 位十六进制串",
  "deviceId": "DEV-8F21",
  "bound": 1,
  "limit": 1,
  "bindMode": true,
  "server": { "name": "某某 ERP", "ver": "1.7.0" }
}
```

字段	必须	说明
deviceToken	是	长期凭证。 只在绑定响应里出现 ，之后不要通过其它接口下发。示例 64 位十六进制；长度以贵方生成为准，代理只判非空
deviceId	否	设备编号，面板与后台对照用
bound / limit	否	绑定数与上限
bindMode	否	供代理判断「绑不上」是否致命

四条语义约定

- 1. 按 `machineId` 幂等：同一台电脑重复调用，`bound` 保持不变，**不占用第二个名额**。`deviceToken` 允许**轮换**——每次重绑发一个新 token、旧 token 立即失效（失效后返回 `401 license_invalid` 或 `403 device_revoked` 都行，代理两者走同一条恢复路径）。客户重装系统、清了配置后拿到的就是新令牌，旧令牌不会继续留在机器上拉单。代理拿到响应后立即把新 token 落盘，**不需要重启**。
- 2. 绑定失败的错误码只看 `bindMode`，与上限是 1 还是 N 无关，四种组合见 5.4。
- 3. 授权码已经绑在别的电脑上：强制绑定模式下统一回 `409 + device_mismatch`，并带上 `devices` 清单，方便现场知道该去后台解绑哪一台。
- 4. 绑定成功即视为已被客户授权，不需要二次人工确认，也不需要发短信/邮件通知。

4.3 print_queue — 取待打印单据

五个接口里调用最频繁的一个：空闲 10 秒一次（可配，最小 3 秒），打完一单立刻再调一次。

请求

```
GET <入口>?a=print_queue&limit=20
Authorization: Bearer <deviceToken 或 授权码>
```

`limit` 由代理配置决定，取值 **1—50**（超出按 50 处理），默认 20。积压多时代理按 20 循环拉，不会一次要 200。

响应

```
{
  "ok": true,
  "paused": false,
  "date": "2026-10-18",
  "orders": [
    {
      "jobId": "oabcd1234#2",
      "attempt": 2,
      "orderId": "oabcd1234",
      "orderNo": "S0-20261018-0023",
      "customer": "永辉超市（朝阳店）",
      "driver": "张伟",
      "amount": 7408.0,
      "date": "2026-10-18",
      "html": "<html>...完整单据 HTML...</html>",
      "paper": { "width": "241mm", "height": "140mm" }
    }
  ]
}
```

字段	必须	说明
<code>ok</code>	是	固定 <code>true</code>
<code>paused</code>	否	后台暂停了远程打印。 <code>true</code> 时代理只看向不取单，下一轮再试
<code>date</code>	否	本批单据的业务日期，代理仅记录，不参与逻辑
<code>orders</code>	是	待打印数组，可以为空数组，不要返回 <code>null</code>
<code>orders[].jobId</code>	是	本次出纸的标识，形如 <code>orderId#attempt</code> 。代理只当不透明字符串用（不解析），据此去重与回写。补打一次，这个值就变一次
<code>orders[].attempt</code>	否	第几次出纸，从 1 开始。代理只用于面板显示「第 2 次」，逻辑不依赖它
<code>orders[].orderId</code>	是	单据主键，必须稳定。它回答「这是哪张单」， <code>jobId</code> 回答「这是第几次打」
<code>orders[].orderNo</code>	否	单号，印在纸面、显示在面板。面板优先展示 <code>orderNo</code> ，为空时退回 <code>orderId</code> 前 8 位；去重与回写仍认 <code>jobId</code>
<code>orders[].customer / driver / amount / date</code>	否	面板与日志展示用
<code>orders[].html</code>	三选一	渲染好的完整 HTML（推荐）
<code>orders[].pdfUrl</code>	三选一	已生成好的 PDF 直链
<code>orders[].templateUrl + orders[].data</code>	三选一	模板地址 + 填充数据，代理本地渲染
<code>orders[].paper</code>	否	纸张尺寸，缺省 241 × 140 mm
<code>orders[].pages</code>	否	业务系统自报页数，仅供参考；代理以实际渲染结果为准

没有 `jobId` 会怎样：代理退回用 `orderId` 去重（与早期版本一致）。出纸照常，但后台点「重打」不会再出纸——这正是 3.1 要解决的问题。服务端上线时请把 `jobId` 一起发出来。

实现要点

1. 返回 ≠ 已打印。代理出纸成功后才回写 `print_result`；在此之前同一个 `jobId` 可以继续返回——代理本地有记录会跳过，不会重复出纸，并会补写一次 `done` 帮你清队列（同一 `jobId` 10 分钟内最多补一次）。最省事的做法：收到 `print_result` 再把这一次的状态置为已打印。
2. 补打就是换一个 `jobId`。后台点「重打」时，`orderId` 不动、`attempt + 1`、`jobId` 变成 `orderId#N+1`，状态回到待打印。代理看到新 `jobId` 就会当成一次新的出纸——这是补打能出纸的唯一机制，不需要额外接口。
3. 按开单时间正序，先开的先出；同一张单的多次 `attempt` 之间不要再排序。

4. 渲染放在哪一步由贵方决定，但要满足两条： – 单轮总耗时 < 500 ms（单页 5.5 KB / 两页 11 KB 时典型值 < 300 ms）。10 秒一轮、打完立刻再拉，单轮拖到秒级就会把队列拉出积压。 – 如果在开单时预渲染，必须同时做「改单后失效重渲染」（渲染签名或版本号）。否则单据在打印前被改了数量或价格，打出来的是旧金额——这个风险比性能风险大，我方建议维持拉单时现渲染。补打时更要重新渲染，否则打出来的是上一次的旧内容。 – 无论哪种方式，不要在这个接口里做无关的重查询（全表扫描、跨库 join、实时汇总）。
5. 放开响应体大小限制并开启 gzip。
6. 返回前做一次「已打印状态」过滤，能显著减少代理的无谓跳过。
7. jobId 只要求稳定且每次出纸唯一，不要依赖代理解析格式；若贵方更愿意用 printId / taskId 之类的名字，也可以，先确认再改（见回执 Q1）。

4.4 print_result — 接收打印结果

代理在每张单出纸成功、或重试次数耗尽后调用。

请求

```
POST <入口>?a=print_result
Authorization: Bearer <deviceToken 或 授权码>
Content-Type: application/json; charset=utf-8
```

成功：

```
{ "jobId": "oabcd1234#2", "orderId": "oabcd1234", "status": "done", "printedAt": 1792309180000, "pages": 2 }
```

失败：

```
{ "jobId": "oabcd5678#1", "orderId": "oabcd5678", "status": "failed",
  "printedAt": 1792309190000, "error": "打印机离线: OKI MICROLINE 5320" }
```

字段	必须	说明
jobId	推荐	对应 print_queue 里的 jobId。服务端以它为准定位是哪一次出纸
orderId	否	单据主键。代理总会一起带上，便于服务端兜底与日志对账；服务端不要在 jobId 缺失时用它去推断「第几次」
status	是	done 已打印 / failed 打印失败
printedAt	否	毫秒时间戳
pages	否	实际出纸张数，多页单据会有值
error	否	failed 时的中文原因，可直接展示给运营

响应

```
{ "ok": true, "jobId": "oabcd1234#2", "attempt": 2, "applied": true }
```

字段	说明
ok	固定 true。幂等冲突也用 true，不用 4xx 表达
applied	true = 这次回写改了状态；false = 没改（重复上报、或已被补打）
stale	true = 这次回写的是过期/已处理的那一次，建议日志里留一条 warn
jobId / attempt	服务端当前认可的那一次，代理据此核对落地情况

四种回写情形，一律 200

代理回写的内容	服务端行为	响应
当前 jobId，done	标 done、记 printed_at	applied:true
当前 jobId，重复 done	幂等，状态不变	applied:false，stale:true
过期 jobId（该单已被补打成 #N+1）	不改状态，记 warn	applied:false，stale:true
不存在的 jobId	不改状态，记 warn，不报错	applied:false，stale:true

为什么冲突也返 200： 如果返 409/400，代理会把它当「回写失败」进入重试或停机，而服务端状态其实是好的。代理侧只在 applied:false 时写一条 warn，不停机、不重试、不影响后续出纸。

语义约定

- 1. 幂等按 jobId：同一个 jobId 重复收到 done 必须视为成功；不同 jobId 是两次独立的出纸，各自记各自的 printed_at。
- 2. failed 之后不要自动重发。失败通常是缺纸、卡纸、打印机离线这类现场问题，自动重发只会反复出纸。请把这一次标为「待人工处理」，由办公室人员在代理面板点「重试」，或在业务后台点「重打」（attempt + 1）——这是系统里唯一允许重打的入口。
- 3. 状态机建议：单据级 pending → done / failed，failed 允许人工改回 pending；出纸级用 attempt 记录每一次，不要把「第 2 次」写成同一条记录的状态覆盖。

4.5 print_agent_status — 整机状态上报

拉单成功只能证明「代理活着」，证明不了「打印机能不能出纸」。这个接口让看板在客户打电话之前就知道缺纸了。

请求

```
POST <入口>?a=print_agent_status
Authorization: Bearer <deviceToken 或 授权码>
X-Print-Agent: <本机 machineId>
Content-Type: application/json; charset=utf-8
```

凭证规则

上报时的情况	服务端应返回	看板
带 <code>deviceToken</code> (已绑定)	200	正常显示, 状态实时
带授权码 (未绑定的应急形态)	200	正常显示, 标注「未绑定」
不带凭证, 但 <code>machineId</code> 已绑定	200	正常显示 (兼容分支; 代理本身永远带凭证)
不带凭证, <code>machineId</code> 未绑定	401 + <code>no_token</code>	不显示, 不要新增设备
凭证错误	401 + <code>license_invalid</code>	不显示 (不要假装成功)

报文

```
{
  "type": "print.agent.status",
  "schema": 1,
  "agent": { "id": "9f2c1e40-7b3a-4d55-9a11-2c6e8f0b1a77", "name": "办公室电脑",
    "version": "1.8.0", "platform": "win32", "os": "Windows 10.0.19045", "host":
"OFFICE-PC" },
  "time": 1792309239000,
  "mode": "pull",
  "dryRun": false,
  "holding": false,
  "printer": { "configured": true, "name": "OKI MICROLINE 5320", "state": "idle",
    "label": "就绪", "detail": "", "jobs": 0, "checkedAt": 1792309238000 },
  "queue": { "pending": 0, "todayDone": 23, "todayFailed": 0, "todayTotal": 23,
    "failedAll": 1, "lastDoneAt": 1792309180000 },
  "lastJob": { "orderId": "oabcd1234", "orderNo": "S0-20261018-0023",
    "customer": "永辉超市 (朝阳店)", "driver": "张伟",
    "status": "done", "pages": 2, "at": 1792309180000, "error": "" },
  "biz": { "phase": "idle", "fatal": false, "licensed": true, "authMode": "device",
    "paused": false, "bound": 1, "limit": 1, "lastPollAt": 1792309235000,
    "lastError": "", "errorCode": "" }
}
```

`printer.state` 是决定要不要报警的字段

值	含义	看板表现
idle	就绪	绿「就绪」
printing	正在打印	蓝「正在打印」
paused	代理侧队列暂停	黄
unconfigured	还没选打印机	黄「还没选打印机」
unknown	查不到状态	黄
offline	没开机 / 断线 / 手动脱机	红「离线」
error	缺纸、卡纸、机盖未开	红「故障：缺纸」（detail 写原因）
notfound	系统里找不到该打印机	红「找不到这台打印机」（驱动被删或改名）

映射优先级：`notfound > offline > error > printing > paused > unconfigured > idle`。「纸张将尽 / 墨粉将尽」这类软提醒只写 `detail`，不改 `state`；拿不准一律 `unknown`——假绿灯比没灯更坏。

其它约定

- 频率：默认 60 秒一次（最小 10 秒）；**每打完一单立即补推一次**；失败不重试，下一轮再推。超过 180 秒没收到可判「失联」。
- 只看 HTTP 2xx 判成功，响应体可不解析；建议回 `{"ok":true,"receivedAt":...,"level":"ok","levelText":"就绪","nextReport":60000}`。
- 字段全部可缺省，服务端**只存最新快照**（一台电脑一行）。
- **报文不含授权码与 `deviceToken`**，可放心落库。
- **报文上限 32 KB**：代理已做收口（超限先丢 `lastJob`），服务端仍建议保留截断兜底。
- `machineId` 是首次启动生成的随机 UUID 并持久化（不用 MAC、不用主机名），换网卡、改机器名不会变成「新电脑」。

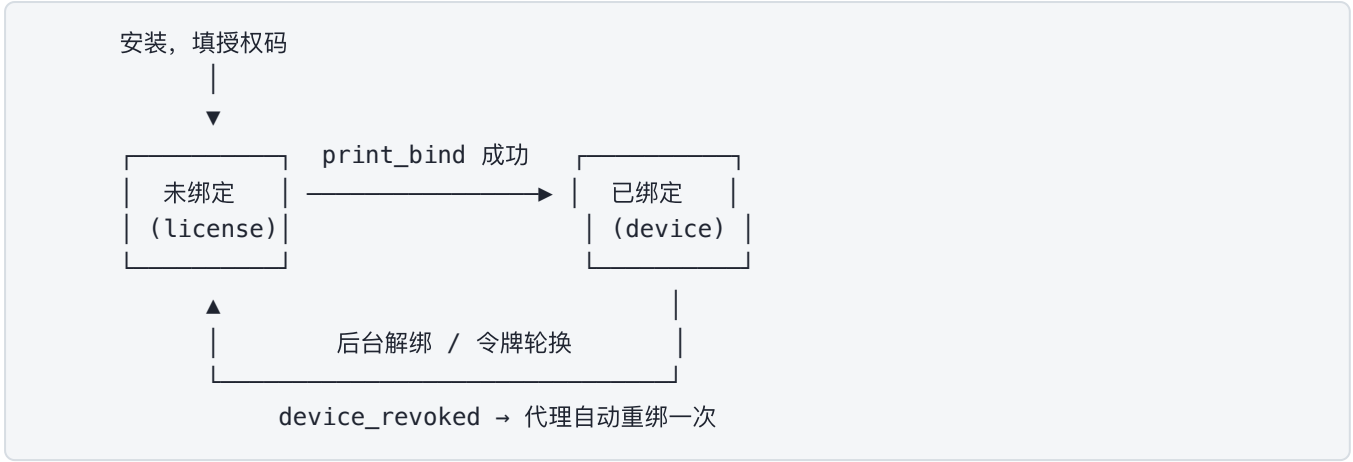
5 一码一机

5.1 为什么必须绑定

只靠授权码也能拉单，但那样后台只看到「某个客户在打」，看不到「哪台电脑在打」，也就无法限台数、无法单独撤销某一台、无法解释「为什么这台打印了两次」。

绑定把「客户级凭证」拆成「设备级凭证」，是后面所有运维动作（换机、解绑、限台数）的基础。

5.2 绑定状态机



代理的恢复策略：

服务端返回	代理动作
<code>device_revoked</code>	清本地 token， 自动用授权码重绑一次 并立即重试；若这次重绑仍被拒（同一窗口内第二次），停机等人工在面板点「重新绑定」
<code>license_invalid</code>	同上（清 token → 重绑一次 → 仍失败才停机）
<code>bind_required</code>	就地绑定后立即重试，用户无感
<code>device_mismatch</code>	停机，列出后台下发的 <code>devices</code> ，等人解绑

5.3 令牌轮换

同一 `machineId` 重新绑定时，服务端**应当换发新的 `deviceToken`**，并让旧 token **立即失效**。原因很直接：重装系统、清空配置后拿到的必须是新令牌，旧令牌不能继续留在某台机器上拉单。

对代理来说这不是异常：它收到新 token 后立即落盘，下一次请求就用新的，**不重启、不丢单**。旧 token 失效后返回 `401 license_invalid` 或 `403 device_revoked` 都能被正确处理。

一个必须注意的边界：代理对自动重绑有**限流**——同一台机器**第一次**自动重绑不设限（对应「已拿到新 token、还没写盘就被断电/杀进程」这类情况：重启后带旧 token 被拒，必须允许它立刻重绑一次）；如果重绑之后 30 秒内**再次**被拒，才判定「不是简单的被解绑」并停机，避免把服务端刷爆。此时在面板点一次「重新绑定」即可恢复。业务系统做数据清理时如果刚好命中这个窗口，会看到这个现象，属预期行为。

5.4 名额满时的码值：只看 bindMode

bindMode	上限	已绑	再绑一台返回	HTTP	代理的正确反应
true（强制绑定）	1	1	device_mismatch	409	停机，提示「已绑在另一台电脑」，等人解绑
true（强制绑定）	N	N	device_mismatch	409	同上
false（应急降级）	1	1	device_limit	409	退回授权码继续出纸，不停机
false（应急降级）	N	N	device_limit	409	同上

要点：上限是 1 还是 N 完全不影响码值，分叉只由 bindMode 决定。两种码都应附 bound / limit / bindMode / devices。

5.5 本项目的确定口径

系统没有存量客户，所以不需要「先兼容、后强制」的过渡期：

- print_bind 从第一天起就按强制绑定运行（bindMode: true）。每台电脑装好即绑定，绑不上就停下来提示，不会出现「绑不上但照样打」的模糊状态。
- bindMode: false 只保留为应急降级开关：万一绑定接口本身故障，临时打开让客户靠授权码继续出纸，修好后立刻关掉。它不是常规运行形态，也不要在线上初期为了「方便」而默认打开。

5.6 三种运维动作

场景	后台怎么做	现场会看到什么
客户换了电脑	先解绑旧电脑，再在新电脑上填同一份授权码	新电脑绑定成功，bound 仍是 1 / 1；旧电脑若还开着，会收到 device_revoked 并提示「授权已解除」
客户重装系统	后台解绑这台旧记录，现场重新填同一份授权码	machineId 是配置里的随机 UUID，重装即新值，服务端按新电脑处理：limit=1 时会撞上 403/409 device_mismatch，解绑旧记录后即恢复
客户要换授权码	生成新码，旧码作废	现场把新码填进代理点「验证授权」；旧码请求返回 401 license_invalid

machineId 为什么用随机 UUID 而不是硬件指纹：业务系统 v5 R2 的结论是「稳定但会误导」——用 MAC、主机名、CPU 序列号、安装路径派生时，换网卡、改电脑名、换系统镜像都会让它变，后台把合法维护当成「第二台」，一码一机直接卡死。随机 UUID 只跟配置文件走，语义单一：**配置在 = 同一台；配置没了 = 新电脑**。后者是一个明确的、可解释的人工动作（后台解绑），比「莫名其妙变成第二台」好排查。

因此：**重装系统 / 清空数据目录 = 新电脑**。这不是缺陷，是双方约定的语义。真要保留身份，备份 data/ 与配置即可。

另一条边界：deviceToken 在 Windows 上用 DPAPI 按**当前 Windows 用户**加密。同一台电脑换个 Windows 账号登录，令牌解不开——代理会当成「没有令牌」，用授权码重绑一次即可（服务端同一 machineId 重绑不占名额）。

6 单据内容与版式

6.1 三种内容来源（三选一）

优先级	字段	业务系统要做什么	适用
1	pdfUrl	提供 PDF 直链	已有预览/导出能力，版式 100% 可控
2	html	整页 HTML 字符串放进字段	已有 HTML 单据页，改造成本最低
3	templateUrl + data	模板托管在业务系统，代理拉取后填充	想一处改版、所有办公室同步生效

三者都没有时，代理退回内置样例模板——**仅用于装完机器点「打印测试页」验链路**，正式单据必须带其中一种。

6.2 纸张与分页

- 默认纸型 **241 × 140 mm**（三联连续纸一份的高度）；可用 paper 字段改，如 { "width": "210mm", "height": "297mm" }。
- HTML 里写了 @page { size: 241mm 140mm; margin: 0 } 时代理完全尊重；没写会自动补上，**不会缩放到 A4**。
- 分页责任在业务系统，代理不按行数猜页数。页数 = 你给的 HTML 渲染出来的实际页数。一页内容用一个 .paper 块承载：

```
.paper { width: 241mm; height: 140mm; box-sizing: border-box;
padding: 6mm; page-break-after: always; }
.paper:last-child { page-break-after: auto; } /* 避免末尾多出一张空白页 */
```

- 多页单据建议每页都印单号、客户、司机、页码，续页标注「承上页」；合计金额与签字栏只放末页。
- 走纸节奏由代理控制：一单打完后等 `settleMs`（默认 3.5 秒）再打下一单，多页单据每多一页追加 2.5 秒，避免连续纸叠印。
- 内置样例模板是「按行数分页」的：≤13 行单页、14–17 行紧凑单页、更多自动续页（首页 13 行 / 中间 16 行 / 末页 14 行）。这只适用于测试页，业务单据请自带 HTML。

6.3 一个可直接用的 HTML 骨架

```
<!doctype html>
<html><head><meta charset="utf-8">
<style>
  @page { size: 241mm 140mm; margin: 0 }
  body { margin: 0; color: #000;
        font: 9pt/1.45 "SimSun", "Songti SC", serif }
  .paper { width: 241mm; height: 140mm; box-sizing: border-box;
          padding: 6mm 8mm; page-break-after: always }
  .paper:last-child { page-break-after: auto }
  h1 { font-size: 13pt; margin: 0 0 2mm; text-align: center; letter-spacing: 2pt }
  .meta { display: flex; justify-content: space-between; font-size: 8.5pt }
  table { width: 100%; border-collapse: collapse; margin-top: 2mm }
  th, td { border: .25mm solid #000; padding: 1mm 1.5mm; text-align: left }
  th { border-width: .5mm; font-weight: 700; background: none } /* 针式不要底纹 */
  td.num { text-align: right }
</style></head>
<body>
  <section class="paper">
    <h1>送 货 单</h1>
    <div class="meta"><span>单号: S0-20261018-0023</span><span>2026-10-18</span></div>
    <div class="meta"><span>客户: 永辉超市（朝阳店）</span><span>司机: 张伟</span></div>
    <table>
      <thead><tr><th>品名</th><th>规格</th><th>数量</th><th>单价</th><th>金额</th></tr></thead>
      <tbody><!-- 明细行 --></tbody>
    </table>
    <div class="meta"><span>合计: 柒仟肆佰零捌元整</span><span>¥7408.00</span></div>
  </section>
</body></html>
```

两页就把 `<section class="paper">` 复制一份，第二页标题写「送 货 单（承上页）」，并在页脚标 `2 / 2`。

6.4 版式禁忌

针式打印是点阵击打 + 无碳复写，只能表达黑白线条与文字：

- **不要放照片或灰度图。**现场照片存在业务系统里，单据上最多印一行「现场照已存档」。
- **不要依赖外链字体/图片。**HTML 必须自包含（CSS 内联或写在 `<style>` 里）。
- **不要用浅灰底纹和大面积色块，**第三联会糊成一片。

- 细线 0.25mm、表头 0.5mm；字号 8—9 pt 是可读性的甜点区。
- 套打偏移属于驱动与纸型的对齐问题，用面板「拉一单预览」打样校准，**不要靠改 HTML 的 margin 去凑**。

6.5 模板语法（仅 `templateUrl` + `data`）

语法	含义
<code>{{orderNo}}</code>	取当前上下文（或根数据）中的字段
<code>{{customer.name}}</code>	点号路径取值
<code>{{#items}} ... {{/items}}</code>	循环数组，块内直接用元素字段
<code>{{^items}}</code> 暂无明细 <code>{{/items}}</code>	数组为空时渲染这一段
<code>{{#items}}{{#specs}}...{{/specs}}{{/items}}</code>	嵌套循环，支持任意层
<code>{{amountCn}}</code>	中文大写金额；不传则用内置转换（支持一个金额字段）

取值规则：先在当前循环上下文找，找不到回退根数据；字段缺失渲染为空字符串，不会中断。

7 幂等与回写

这是整套方案最容易出事的地方。

代理本地按 `jobId` 记录每一次出纸（排队中 / 打印中 / 已完成 / 失败）。为什么不是 `orderId`：纸是按「一次物理出纸」产生的，卡纸重打、撕单重打、客户要第二联，**业务单据没变，出纸次数变了**。用 `orderId` 去重，这几种情况都不会出纸。

场景	代理行为
<code>print_queue</code> 返回本地已有的 <code>jobId</code>	直接跳过， 不出纸 ；若本地是 <code>done</code> ，额外补写一次 <code>print_result</code>
业务系统重复下发同一个 <code>jobId</code> （抖动、重试）	同上，不会重复出纸
后台点「重打」（ <code>attempt + 1</code> ， <code>jobId</code> 变）	当成新的一次出纸： 正常出纸 ，回写用新 <code>jobId</code>
客户在代理面板点「重打」（仅限本机任务）	本地复制一份任务再打； 业务系统的单不提供这个入口 ，见 7.3
业务系统换了 <code>orderId</code> 下发同一张单	会被当成新单再打一遍 （换了单号就是换了单，代理无法识别）

7.1 对业务系统的两条硬要求

1. **orderId 必须稳定**：用订单主键（如 oabcd1234），不要用自增序号拼时间戳，也不要再在重发时重新生成。它回答「这是哪张单」。
2. **jobId 必须每次出纸唯一**：约定 orderId#attempt，attempt 从 1 开始递增。它回答「这是第几次打」。只要这两个值对，代理与服务端的账就能对上；orderNo 只用于纸面与面板展示，**不能当幂等键**。

7.2 failed 之后不要自动重发

失败通常是缺纸、卡纸、打印机离线这类**现场**问题。自动重发只会反复出纸、浪费三联单。正确做法：把这一次标为「待人工处理」，由办公室人员在代理面板点「重试」，或在业务后台点「重打」（attempt + 1）——这是系统里唯一允许重打的入口。

7.3 补打只在业务后台做

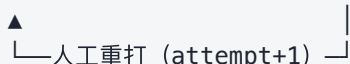
代理面板对**业务系统的单据**不提供「重打」。原因不是技术限制，而是口径问题：业务看板的「今日出纸」按 print_ticket 的出纸流水统计，而补打的流水只有业务后台点「补打」时才会生成。若允许现场在面板上再打一张，纸出来了、业务系统却不知道，两边数字永远对不上（v5 §8.3 最后一条）。

所以现场遇到「这张要再打一份」，统一动作是：**业务后台找到这张单 → 点「补打」→ 代理下一轮（≤10 秒）拉到 jobId 变成 #N 的新任务 → 正常出纸**。

代理面板的「重试」只用于 failed 的那一次——那一次纸根本没出来，不算「多打一份」。

7.4 建议状态机

单据级： pending —首次出纸成功→ done



出纸级： #1 done → #2 done → #3 failed —人工重试/重打→ #3 done

- 单据级只看「当前有没有一次成功的出纸」；出纸级用 attempt 逐次记录，**不要用一条记录覆盖上一次的结果**，否则后台看不出「这张单到底打了几次」。
- done 之后不要再把同一 jobId 改回 pending，否则代理会当成新的一次再打一遍。
- failed 允许人工重打：attempt + 1 即可，代理自然识别为新的出纸。
- **job_id 不要回收**：代理本地保留 30 天的 jobId 集合，服务端如果把 attempt 归零再用回 #1，这个「新的一次」会被代理当成已打过而**不出纸**。要重来就继续递增。

7.5 本地记录的保留期（v5 R8）

代理把 jobId 集合存在 data/jobs.json：

- **保留 30 天**：只清 30 天前的「已完成」记录；失败 / 排队中的不按时间清（还要在面板上等人处理）。
- 断电、崩溃、杀进程后重启：**不清空集合，也不「重打最近 N 单」**。服务端会重新下发仍为 pending 的单，集合命中即跳过——这才是正确的恢复方式（V5 R8 原文）。

- 硬上限 20000 条只是保险，正常业务量（每天几百单）30 天远到不了。

8 错误码与代理行为

业务系统的 `code` 决定代理下一步做什么：

错误码	HTTP	含义	代理行为
<code>no_token</code>	401	请求没带 <code>Authorization</code> （含匿名上报但本机未绑定）	手里有 <code>deviceToken</code> 却被判 <code>no_token</code> ：清令牌、重绑一次再试；重绑仍失败则停轮询（v5 §2.2）。本来就没有令牌：停机并提示填授权码
<code>module_off</code>	403	套餐未开通远程打印	停止拉单，提示联系服务商
<code>license_invalid</code>	401	授权码不正确或被更换	清本地 token；首次自动重绑不设限，重绑后 30 秒内再次被拒则停机
<code>device_revoked</code>	403	本机已被解绑（或令牌已轮换）	清本地 token 并自动重绑一次；首次不设限，重绑后 30 秒内再次被拒则停机
<code>device_limit</code>	409	名额已满， 只在 <code>bindMode:false</code> 时出现	退回授权码继续拉单，不停机；10 分钟后再试绑定
<code>device_mismatch</code>	409（绑定） / 403（数据接口）	授权码已绑定其它电脑，或强制绑定下名额已满	停机，提示「这个授权码已经绑定其它电脑」，并列出 <code>devices</code>
<code>bind_required</code>	403	拉单前必须先绑定	就地绑定后立即重试
<code>bad_request</code>	400	参数不完整	停机并显示服务端给的 <code>error</code>
（非 JSON 响应）	任意	返回了 HTML / 纯文本	判定地址填错，停机提示
（连接失败 / 超时）	—	网络层错误	按 10 → 20 → 40 → 60 秒退避重试，不刷日志

配套规则：

- **401 / 403 / 409 一律不重试**，只有网络错误与 5xx 才指数退避。
- `license_invalid` 与 `device_revoked` 对代理是**同一条恢复路径**：清本地 token → 用授权码重绑一次 → 仍失败才停机。服务端用哪个码不影响出纸，只影响排障可读性（403 说明被解绑或令牌轮换，401 说明码本身不对）。两者都不要合并成同一个码。
- 代理按 `body.code` 分流，不看状态码；只给状态码不给 `code` 时，代理按状态码停机并提示「服务端未返回错误码」。
- 手动点「重新绑定」成功后，代理自动解除停机并恢复拉单，**不需要重启**。

- 停机（`fatal`）的语义是「继续请求也不会有不同结果」；网络类错误则一直自动重试。
- 回写结果的 `applied:false` 不是错误：代理只记一条 warn，不停机、不重试、不影响后续出纸。服务端不要用 4xx 表达「重复/过期回写」，否则会把好状态报成故障。

9 从空库到第一张纸

这一章是施工顺序。**不要跳步**：每一步的产出都是下一步的前置条件，跳过去一定会在后面某个地方卡住，而且现象会很误导。

每一步给三样东西：做什么、应该看到什么、不符时先查什么。

S1 确认基线确实是「零」

```
SELECT COUNT(*) FROM print_device;    -- 期望 0
SELECT COUNT(*) FROM print_order;     -- 期望 0
SELECT COUNT(*) FROM print_status;    -- 期望 0
```

应该看到：三个 0，`print_license` 里该租户 `licensed=1`、`bind_mode=1`、`device_limit=1`。

不符先查：是否连到了测试库；是否有一个「演示租户」被误开通并绑了设备。

S2 后台生成授权码

生成一份新的、未使用过的授权码，记下它（后面 4 处要用）。

应该看到：后台「远程打印」页显示 0 台设备，授权码可复制。

不符先查：是否把「拉单地址」和「授权码」搞混——地址是 `?a=print_queue` 的链接，码是那串十六进制。

S3 建表建索引

按 3.6 建四张表；至少把 `print_license.code`、`print_device.machine_id`、`print_device.token_hash` 建成唯一索引，`print_order (tenant_id, status, created_at)` 建复合索引。

应该看到：四条 DDL 成功，表为空。

不符先查：字符集是否 `utf8mb4`（客户名里有 emoji 或生僻字时 `utf8` 会截断）。

S4 先实现 `print_license + print_bind`

只做鉴权与绑定，队列相关接口先返回空实现。

应该看到：用 curl 调自检能返回 `authMode=license`、`bound=0`；调绑定能拿到 `deviceToken`。

不符先查：`Authorization` 头是否被框架自动剥离（部分网关会吃掉非标准头）；`body.code` 是否被读成了数组/对象。

S5 再实现 `print_queue + print_result`

空队列先返回 `{"ok":true,"orders":[]}`，把回写做成幂等。

应该看到：空队列 200 + 空数组；对不存在的 `orderId` 回写也返回 200。

不符先查：是否返回了 `null` 而不是 `[]`；`print_result` 是否对重复 `done` 报了 409。

S6 最后实现 `print_agent_status`

先只存快照，不做看板。

应该看到：上报 200，`print_status` 里出现一行。

不符先查：是否把 `X-Print-Agent` 当成鉴权依据（它只是机器标识）；未绑定的 `machineId` 不带凭证时应回 401 而不是新建一行。

S7 用 curl 走一遍五个接口

```
BASE="https://erp.example.com/api/print.php"
CODE="你的 48 位授权码"

# 1 自检（未绑定）
curl -s -H "Authorization: Bearer $CODE" "$BASE?a=print_license"

# 2 绑定（两个位置都给，便于验证服务端读哪个）
curl -s -X POST -H "Authorization: Bearer $CODE" \
  -H 'Content-Type: application/json' \
  -d '{"code":"'${CODE}'","machineId":"'curl-test-001'","name":"'curl'","os":"'test'","ver":"'curl'"}' \
  "$BASE?a=print_bind"

# 3 用返回的 deviceToken 自检
TOKEN="上一步返回的 deviceToken"
curl -s -H "Authorization: Bearer $TOKEN" "$BASE?a=print_license"

# 4 空队列
curl -s -H "Authorization: Bearer $TOKEN" "$BASE?a=print_queue&limit=20"
```

应该看到：四步依次为 `authMode=license` → `deviceToken` → `authMode=device` → `orders:[]`。

不符先查：第 2 步若回 401，先看服务端有没有读 `body.code`；若回 409 `device_mismatch`，说明这份码已经绑过，回后台解绑。

S8 跑一键脚本，19 条用例

```
python3 zerotest.py "$BASE" "$CODE"
```

应该看到：结果：PASS 19 / FAIL 0 / SKIP 0，退出码 0。

不符先查：先跑 → 后台 `reset` 让状态回到零再跑第二遍；C13—C17 需要后台配合，脚本会标 `SKIP` 而不是 `FAIL`。

S9 装代理，填地址与授权码

办公室电脑装 `远程打印.dmg` (macOS) 或 `远程打印-Windows.zip` 里的 `安装.bat` (Windows)，打开面板 → 设置 → 填接口地址与授权码 → 点「验证授权」。

应该看到：面板显示「授权有效 · 已绑定本机」，绑定 `1 / 1`；后台设备列表出现这台电脑，名称与 `os` 正确。

不符先查：地址是否少了 `/api.php`；租户站是否多写了 `/wb/`；公司网络是否需要代理（面板可配）。

S10 打测试页，校准纸型

面板点「打印测试页」。

应该看到：出纸一张，241 × 140 mm，不缩放、不裁切；表头不歪、第三联可读。

不符先查：驱动里是否建了 241 × 140 mm 自定义纸型（不能选 A4）；套打偏移用面板「拉一单预览」调边距，不要改 HTML。

S11 开第一张真实单

在业务系统正常开一张单（1 页），保存后什么都不用做。

应该看到：10 秒内出纸；`print_result` 收到 `done`、`pages=1`；再拉单时该单不再出现。

不符先查：看面板 `phase` 是否 `polling`；看 `biz.lastError`；看 `print_queue` 返回的 `html` 是否为空字符串（空会被当无内容）。

S12 开一张两页单

单据里放两个 `.paper` 块（明细超过一页即可）。

应该看到：连续出纸 2 张；第 2 张标「承上页」；`pages=2`；两页之间没有叠印。

不符先查：是否只渲染出 1 页（块高超过 140 mm 会被压成一页或裁掉）；`.paper` 是否缺少 `page-break-after: always`。

S12b 补打一张（验证 jobId）

在业务后台对刚打完的那张单点「重打」，什么都不用改。

应该看到：10 秒内再出一张同样的纸；面板任务列表多出一行并标「第 2 次」；服务端 `print_result` 收到 `jobId=orderId#2`，`applied:true`。

不符先查：后台重打后 `print_queue` 里的 `jobId` 有没有变（没变说明服务端只改了状态、没加 `attempt`）；代理是否仍在按 `orderId` 去重（看面板版本是否 1.8.0 及以上）。

S13 核对看板

打开「打印电脑与打印机状态」页。

应该看到： 出现这台电脑，`printer.state=idle`、`printer.name` 是实际驱动名、`biz.authMode=device`、`queue.todayTotal ≥ 2`。

不符先查： 面板点「发送测试上报」看 HTTP 码；`printer.state=unknown` 时打印原始状态（Windows 看 `PrinterStatus`，macOS 用 `lpstat -p <name> -l`）。

S14 做三件破坏性演练，再签收

演练	操作	应该看到
缺纸	打印中抽走纸张	单据 <code>failed</code> 、 <code>error</code> 含「缺纸」；放回纸点「重试」能出纸
解绑	后台解绑这台电脑	1 分钟内自动重绑，恢复出纸，全程无人工
断网	拔网线 1 分钟	面板显示「连不上」；恢复后自动追平积压，不重复出纸

不符先查： 解绑后不自动重绑 → 服务端把 `device_revoked` 回成了别的码，或重绑限流窗口命中（面板手动「重新绑定」可恢复）。

全部通过后签附录 B，本节结束。

10 验收

10.1 接口层 22 条用例

编号	操作	期望	判定
C01	未绑定 + 授权码调 print_license	200; authMode=license、needsBind=true、 bindMode=true、bound=0	四值齐全才算过
C02	不带 Authorization 调 print_license	401 + no_token，带 WWW-Authenticate: Bearer	不是 403，也不是 500
C03	错误授权码调 print_license	401 + license_invalid	必须与 C02 的码 区分开
C04	带 Authorization + body.code 调 print_bind	200; deviceToken 非空且 ≥ 32 位、deviceId 非 空	代理两个位置都 给，只认其一的实 现也能过
C05	同一 machineId 再绑一 次	200; 返回可用的 deviceToken（允许轮换）； bound 仍为 1	不许占第二个名 额；轮换与否都算 过
C06	用 deviceToken 调 print_license	200; authMode=device、needsBind=false	
C07	用 deviceToken 拉单 (队列空)	200; orders=[]	空数组不要返回 null
C08	开一张单后拉单	200; 返回该单， jobId / attempt=1 / orderId / orderNo / html 齐备	jobId 形如 orderId#1
C09	回写前再拉一次	同一 jobId 仍返回	去重责任在代理， 服务端不要偷偷改 状态
C10	用当前 jobId 回写 done 后再拉	该单不再出现	
C11	重复回写同一 jobId	200 + applied:false、stale:true	幂等，不要报 409
C12a	已绑定 + 带 deviceToken 上报	200	
C12b	同一台不带凭证上报	200	兼容分支，允许放 行
C12c	未绑定 machineId 不带 凭证上报	401 + no_token	看板不应新增设备

编号	操作	期望	判定
C13	第二台机（新 <code>machineId</code> ）用授权码绑定， <code>limit=1</code>	409 + <code>device_mismatch</code> ，带 <code>devices</code> 、 <code>bindMode</code>	不是 <code>device_limit</code>
C14	后台解绑本机后，用旧 <code>deviceToken</code> 拉单	403 + <code>device_revoked</code>	不是 401
C15	解绑后用授权码重绑	200；发新 token；旧 token 失效（401 <code>license_invalid</code> 或 403 <code>device_revoked</code> 均可）	令牌必须换、旧令牌必须不能再拉单
C16	关掉强制绑定后第二台绑定（ <code>bindMode=false</code> ）	409 + <code>device_limit</code>	应急形态专用
C17	<code>bindMode=false</code> 下用授权码拉单	200	降级后必须能继续出纸
C18	后台对已打印的单点「重打」后再拉	200； <code>orderId</code> 不变、 <code>jobId</code> 变（#2）、 <code>attempt=2</code>	补打的核心用例
C19	用旧 <code>jobId</code> 回写 <code>done</code>	200 + <code>applied:false</code> 、 <code>stale:true</code> ；该单 仍是待打印	不能被误标成已完成
C20	用新 <code>jobId</code> 回写 <code>done</code> 后再拉	200 + <code>applied:true</code> ；该单离开队列	

10.2 一键脚本

随本文附 `zerotest.py`（Python 3，只用标准库，三个平台都能跑）：

```
python3 zerotest.py https://erp.example.com/api/print.php <授权码>
```

输出逐条 PASS / FAIL，末尾给出汇总，全部通过时退出码为 0。

- C13—C17 需要后台配合（解绑、切换开关），脚本拿不到后台接口时会标 **SKIP** 而不是 FAIL，请按 10.1 的表手工补做。
- C18—C20 需要「后台点一次重打」，同样标 SKIP；服务端若还没发 `jobId`，这三条会整组 SKIP（不算失败，但补打功能等于没验）。
- 脚本默认不走系统代理，避免办公室环境的 `HTTP_PROXY` 把请求带偏。
- C04 判据是「非空且 ≥ 32 位」而不是「正好 64 位」；C05 只要求「拿到可用 token 且 `bound` 不变」；C15 的旧令牌失效允许 401 或 403；C11/C19 只要求 `applied:false` 且状态不变，不限定具体文案。这几条是刻意放宽的，避免「实现正确但脚本判 FAIL」的假缺陷。

10.3 代理端 8 项走查 (Windows / macOS 各一台)

编号	操作	期望
T1	装好代理，首次启动	<code>machineId</code> 是随机 UUID v4 (36 字符，面板「本机 ID」可见) 并落盘；重启不变，删掉配置才变
T2	填接口地址 + 授权码，点「验证授权」	显示「授权有效 · 已绑定本机」， <code>绑定 1/1</code> ；后台出现这台电脑
T3	点「打印测试页」	出纸；241×140 mm，不缩放、不裁切
T4	业务系统开 1 张单	10 秒内出纸； <code>print_result</code> 收到 <code>done</code> ；队列里该单消失
T5	开 1 张两页单 (两个 <code>.paper</code> 块)	连续出 2 张，第 2 张标「承上页」， <code>pages=2</code>
T6	面板任务列表	显示 <code>orderNo</code> (不是 <code>orderId</code>)；补打的单标「第 2 次」
T7	看板	出现该机， <code>printer.state=idle</code> ， <code>biz.authMode=device</code>
T8	打印中抽走纸张	该单标 <code>failed</code> ， <code>error</code> 含「缺纸」；放纸后点「重试」出纸

10.4 异常演练 8 项 (上线前做一遍)

演练	操作	期望
幂等	让 <code>print_queue</code> 连续两轮返回同一 <code>jobId</code>	只出一次纸，任务列表无重复
补打	业务后台对已打印的单点「重打」(<code>attempt+1</code>)	10 秒内 再出一张 ；面板显示「第 2 次」；回写 <code>applied:true</code>
过期回写	补打后，用旧 <code>jobId</code> 手工回写一次 <code>done</code>	200 + <code>applied:false</code> ；该单仍是待打印，代理不停机
解绑恢复	后台解绑该电脑	代理自动重绑，1 分钟内恢复出纸
令牌失效	让该机 <code>deviceToken</code> 失效 (或后台解绑)	代理清令牌后自动重绑一次；仍失败才停机
令牌轮换	服务端对同一 <code>machineId</code> 重绑并换发新 token (旧 token 立即作废)	下次请求收 403 <code>device_revoked</code> → 自动重绑 → 拿到新 token → 继续出纸；全程不停机、不丢单
网络中断	拔网线 1 分钟	面板显示「连不上」，恢复后自动追平积压
暂停开关	后台置 <code>paused:true</code>	代理停止取单，恢复后自动继续
上报	面板点「发送测试上报」	接收端 200，看板状态实时更新

10.5 通过标准

- 10.1 的 22 条全部 PASS（SKIP 的必须手工补做）；
- 10.3 的 8 项走查全部通过；
- 10.4 的 8 项演练至少各做一次，行为与表中一致；
- 附录 B 逐项勾选并签回。

10.6 我方在空库环境里的实测记录

环境：2026-10-18，代理 v1.8.0，macOS；业务端为按本规范实现的**空库模拟服务**（`server.ver=1.7.0`）—— `0` 设备 / `0` 单据 / `0` 状态快照 / `bindMode:true` / `limit:1`，代理从全新配置（`machineId` 与 `deviceToken` 均为空）开始，出纸为演练模式（只生成 PDF，不真送打印机）。

项	结果
接口层 22 条用例	22 PASS / 0 FAIL / 0 SKIP
C01 未绑定自检	200, <code>authMode=license</code> 、 <code>needsBind=true</code> 、 <code>bound=0/1</code>
C02 无凭证自检	401 <code>no_token</code>
C04 绑定	200, <code>deviceId=DEV-xxxx</code> 、 <code>deviceToken</code> 64 位
C05 同机重绑	200, 令牌已轮换 (新旧不同), <code>bound</code> 仍 1/1, <code>bindCalls</code> 只 +1
C08 拉单	<code>jobId=oaf7fee93#1</code> 、 <code>attempt=1</code> , 字段齐备
C10 正常回写	<code>applied:true</code> , 该单离开队列
C11 重复回写同一 <code>jobId</code>	200 + <code>applied:false</code> 、 <code>stale:true</code> , 状态不变
C18 后台补打	<code>jobId</code> 由 #1 变 #2、 <code>attempt=2</code> , <code>orderId</code> 不变
C19 旧 <code>jobId</code> 回写	200 + <code>applied:false</code> 、 <code>stale:true</code> , 该单 仍在队列 (未被误标完成)
C20 新 <code>jobId</code> 回写	<code>applied:true</code> , 队列清空
代理补打全链路	同一 <code>orderId</code> 本地出现两条任务 (#1 / #2), 两次都出纸 , 两次回写均 <code>applied:true</code> ; 改前第 2 次会被本地去重吞掉
令牌轮换后继续拉单	面板「重新绑定」→ 新 token 立即落盘, <code>phase=polling</code> 、 <code>fatal=false</code> , 未重启、未丢单
解绑后自动重绑	8 秒内自动重绑, <code>bound</code> 回到 1/1, 恢复轮询, 全程无人工
绑定后 1 秒内被解绑 (S2)	首次自动重绑直接放行, <code>phase=polling</code> 、 <code>fatal=false</code> ; 改前会命中 30 秒保护而停机
<code>machineId</code>	随机 UUID v4 (如 <code>9f2c1e40-7b3a-4d55-9a11-2c6e8f0b1a77</code> , 36 字符); 重启不变, 删掉配置才变
<code>deviceToken</code> 落盘	<code>config.json</code> 里恒为空串; 令牌在 macOS 钥匙串 (<code>data/device.token</code> 只留 <code>{"v":1,"alg":"keychain"}</code> 标记, 权限 0600), 面板显示 <code>2da183...(64)</code>
令牌跨重启复用	杀进程重启后 <code>bindCalls</code> 不变 (仍是 2), <code>authMode=device</code> —— 从加密存储读回, 无需重绑
解绑清理	清 token 后 <code>config.json</code> 、加密存储、内存三处同时为空, 下次启动不会「复活」旧令牌
第二台电脑同码 (<code>limit=1</code>)	<code>fatal=device_mismatch</code> , 面板列出已绑设备, 未出纸; <code>bindCalls</code> 只 +1
业务单禁止面板重打	对 <code>source=biz</code> 的任务调 <code>/reprint</code> → 409 , 提示「请在后台点补打」, 不会产生绕过看板的第二次出纸

项	结果
空授权码点「验证授权」	不发任何请求，面板直接提示「还没填授权码」
状态上报	报文 950 B， <code>agent.version=1.8.0</code> ， <code>printer.state=printing</code> ，打完一单立即补推

以上是我方环境的结果，请贵方在自己的实现上原样跑一遍——10.1 的表就是给这个用的。

11 从 1 到 N

第一台跑通之后，剩下的事情都是同一条路径的重复，只有三处需要提前想清楚。

11.1 第二台电脑（同一客户）

- `device_limit=1` 且 `bindMode=true`：第二台绑定会收到 `409 device_mismatch`。这不是故障，是设计——办公室只该有一台在打。确实要换，先解绑旧机。
- `device_limit=N`：按 5.4 处理，`bindMode` 决定码值，与 N 无关。
- 要看「这台是不是重复装了两遍」：比对 `machineId`，不要比对主机名（可以重名）。

11.2 多客户

- 一客户一码，绝不复用。码是租户隔离的唯一依据。
- 队列查询必须带 `tenant_id`，不要出现「按状态全表捞」的写法——那会把 A 客户的单发给 B 客户的电脑。
- 每台电脑只属于一个租户；跨租户共用的打印机不存在于本方案。

11.3 容量假设

参数	单机量级	说明
空闲请求	6 次/分钟（10 秒一轮）	100 台电脑 ≈ 10 次/秒，属于轻负载
上报	1 次/分钟 + 每单一次	打完 10 单约 15 次/分钟
单轮响应	建议 < 500 ms	超过 1 秒会在积压时明显拖慢
队列单条	5–30 KB HTML	<code>limit=20</code> 时未压缩 220 KB 量级，gzip 后 25–30 KB

按这个量级，单台应用服务器 + 单库可以支撑数百台电脑；真正的瓶颈通常不是接口，而是渲染（4.3 要点 3）。

11.4 换机、重装、换码

见 5.6 的三行表。三条的共同点：**后台动作只有一个「解绑」**，现场动作只有一个「填码 + 验证授权」；不需要远程登录、不需要改配置文件、不需要重启服务。

12 常见问题

现象	原因	处理
绑定失败，提示没收到凭证	请求里没有任何可识别的凭证，或服务端只认其中一种而代理只给了另一种	代理两个都给；仍报则查服务端是否把 <code>body.code</code> 读成了字符串（C04）
第二台电脑提示「已绑其它电脑」	强制绑定下 <code>limit=1</code> 且已绑满	后台解绑旧机（C13）
看板一直是空的	上报没带凭证，或地址写错（租户站多加了 <code>/wb/</code> ）	面板点「发送测试上报」看 HTTP 码
看板有电脑但状态永远 <code>unknown</code>	状态采集没命中	打印原始 <code>PrinterStatus</code> （Win）或 <code>lpstat -p <name> -l</code> （mac）
解绑后不自动重绑	服务端把 <code>device_revoked</code> 回成了别的码，或重绑限流命中	应为 403 + <code>device_revoked</code> （C14）；限流命中时面板点「重新绑定」
拉单一直 403 <code>bind_required</code>	服务端已开强制绑定	正常：代理会立刻绑定再重试
面板提示「接口返回的不是 JSON」	地址命中网页/登录页，或异常时返回了 HTML	浏览器打开该地址应看到 <code>{"ok":...}</code>
一张单打了两遍	业务系统换了 <code>orderId</code> ，或补打时没加 <code>attempt</code> （代理按同一 <code>jobId</code> 又收了一次）	<code>orderId</code> 用主键；补打走 <code>attempt + 1</code> ，见 7.1
客户点了「重打」但不出纸	服务端只把状态改回 <code>pending</code> ， <code>jobId</code> 没变，代理当成已打过	<code>jobId</code> 必须随 <code>attempt</code> 变（C18）
补打后旧回写把新单标成已完成	回写没按 <code>jobId</code> 定位，只按 <code>orderId</code> 覆盖	按 <code>jobId</code> 回写，过期回写回 <code>applied:false</code> （C19）
回写返回 <code>applied:false</code>	重复上报或该次已被补打	正常，代理只记 <code>warn</code> 不停机；服务端别改成 4xx
多页只打了第一页	HTML 没按 <code>.paper</code> 分块，或块高超过 140 mm	一页一块，固定 140 mm 高（T5）
重绑后旧令牌立刻 403	服务端在重绑时轮换了 <code>deviceToken</code>	正常。代理会保存新令牌继续拉单（C05 / 10.4）
面板显示「业务系统 v1.7.0」	来自 <code>server.ver</code>	正常，联调时用它对齐版本；不显示也不影响出纸
套打位置偏移	驱动里纸型与实纸不一致	驱动建 241×140 mm 自定义纸型，再用预览校准

现象	原因	处理
拉单正常但不出纸	演练模式开着 / 打印机名不对 / 缺纸	关演练、核对打印机名、看 <code>printer.state</code>
单据里的中文变方块	PDF 渲染缺字体	HTML 用系统自带字体 (SimSun / Songti SC / PingFang SC)，不要外链字体

附录 A 单据字段速查（`print_queue` → `orders[]`）

字段	类型	必须	说明
<code>jobId</code>	string	是	本次出纸的标识， <code>orderId#attempt</code> ；补打后变化
<code>attempt</code>	number	否	第几次出纸，从 1 开始（面板显示用）
<code>orderId</code>	string	是	主键，稳定唯一
<code>orderNo</code>	string	否	单号，纸面与面板展示
<code>customer</code>	string	否	客户名称
<code>driver</code>	string	否	司机
<code>amount</code>	number	否	金额
<code>date</code>	string	否	<code>YYYY-MM-DD</code>
<code>html</code>	string	三选一	完整 HTML
<code>pdfUrl</code>	string	三选一	PDF 直链
<code>templateUrl</code>	string	三选一	模板地址
<code>data</code>	object	否	模板填充数据
<code>paper</code>	object	否	<code>{ width, height }</code> ，缺省 241×140 mm
<code>pages</code>	number	否	自报页数（参考值）

附录 B 请贵方确认的清单

请逐项勾选后签回；有不同意见请在备注里写明接口或字段。

#	确认项	是	备注
1	五个动作名与 <code>?a=</code> 入口与本文一致	☐	
2	<code>print_bind</code> 同时接受 <code>body.code</code> 与 <code>Authorization</code> 头，两者都给时以 <code>body.code</code> 为准	☐	代理两个都给
3	绑定越界只按 <code>bindMode</code> 分叉，与上限是 1 还是 N 无关	☐	
4	未绑定 <code>machineId</code> 的匿名上报返回 401 <code>no_token</code> ，且不新增设备	☐	
5	<code>orderId</code> 使用订单主键且稳定； <code>orderNo</code> 仅用于展示	☐	
6	<code>limit</code> 钳制在 1–50，队列接口支持 <code>gzip</code>	☐	
7	错误码通过 <code>body.code</code> 表达且码值稳定	☐	
8	分页以 HTML 的 <code>.paper</code> 块为准，不按行数推断	☐	
9	本项目从第一天起开启强制绑定（ <code>bindMode:true</code> ），无过渡期	☐	
10	第 10.1 节 22 条用例在本方环境全部 PASS	☐	
11	同机重绑允许轮换 <code>deviceToken</code> ， <code>bound</code> 不变；旧令牌立即失效（401 或 403 均可）	☐	C05 / C15
12	<code>print_license</code> / <code>print_bind</code> 响应里给 <code>server.ver</code> （可选，仅面板显示）	☐	缺失不报错
13	<code>print_queue</code> 每个 <code>orders[]</code> 都带 <code>jobId</code> （ <code>orderId#attempt</code> ）与 <code>attempt</code> ；补打时 <code>attempt+1</code>	☐	3.1 核心
14	<code>print_result</code> 以 <code>jobId</code> 为准；重复/过期回写返回 200 + <code>applied:false</code> 、 <code>stale:true</code> ，不返 4xx	☐	C11 / C19
15	<code>machineId</code> 按随机 UUID v4 字符串处理（长度 ≤ 80 、字符集 <code>A-Za-z0-9_-</code> ），服务端不假设它是硬件标识	☐	v5 R2
16	<code>job_id</code> 只递增不回收：后台重置 / 重新开单不要把 <code>attempt</code> 归零，否则代理会漏打	☐	7.4

签署：_____ 日期：_____

附录 C 版本

版本	日期	说明
规范 3.2	2026-10-18	按贵方《外部服务接入要求 v5》对齐： <code>machineId</code> 回到随机 UUID v4（不再硬件派生）； <code>deviceToken</code> 加密落盘（DPAPI / 钥匙串 / 0600）且 <code>config.json</code> 不再存明文； <code>jobId</code> 集合保留 30 天；业务单的补打只在业务后台做（面板不再提供「重打」）；带令牌仍被判 <code>no_token</code> 时清令牌重绑一次。接口与错误码无变化。对应代理 v1.8.0
规范 3.1	2026-10-18	补打闭环：新增 <code>jobId</code> （ <code>orderId#attempt</code> ）与 <code>attempt</code> ， <code>print_result</code> 按 <code>jobId</code> 幂等并回 <code>applied / stale</code> ； <code>machineId</code> 改硬件指纹派生；同机首次自动重绑不再触发限流停机。验收 22 条接口用例 + 8 项走查 + 8 项演练。对应代理 v1.7.0
规范 3.0	2026-10-18	零存量基线。按「0 客户 / 0 单据 / 0 设备」的起点重写：单一运行形态（强制绑定）、无过渡期、无历史兼容；新增空库最小数据结构（3.6）、可直接使用的 HTML 骨架（6.3）与从空库到第一张纸的施工顺序（第 9 章）。对应代理 v1.6.3

2.x 及更早为双方协商过程稿，已归档，仅作历史参考，不作为接入依据。