EPORDO Connect 接口说明书（中文）

项	内容
文档版本	**1.1.5**
发布日期	2026-11-11
读者	接入方开发者、中间商 / 集成商、项目交付人员
产品	**EPORDO Connect**（第三方设备接入云平台）
交付形式	本 Markdown 源稿 + 门户可下载的 **PDF**（帮助 → 下载接口说明书）

> 本文描述**租户侧 REST API**（路径前缀 `/connect/v1`）。设备与云端之间的通讯由 Connect 完成，接入方**无需**解析或实现设备原始二进制 / 文本协议。

> 文中示例默认使用生产风格的 Base URL；联调时请换成交付环境给出的地址。

> **约定：** 下文路径均相对于 Base URL。JSON 字段名以本文为准（例如事件业务体字段为 `payload_json`，需自行 `JSON.parse`）。

修订记录

版本	日期	说明
1.1.5	2026-11-11	§5.3.1 五语补齐 `door_mode`、`door_opened`；`photo_status` 改为 `absent` / `stored` / `omitted_too_large` / `omitted_invalid`
1.1.4	2026-11-10	§5.3 现行类型为 `door.remote.recorded`、`device.presence`；补充在线 10 分钟窗口、离线约 5 分钟扫描，以及旧类型名只匹配对应边沿
1.1.3	2026-11-08	完善：§5.3 表头改为 `event_type` 并补充 `door.open.recorded`；附录 D 命令速查改为 `GET /commands/{id}` + `POST .../cancel`
1.1.2	2026-11-08	纠错：事件字段为 `event_type`（非 `type`）；抓拍接口返回 JSON/ `data_base64`（非直接图片流）；无 `GET /webhooks/{id}`；无 `DELETE /device-groups/{id}`（用 `status=DISABLED`）；留存天数说明对齐 3–14 天
1.1.1	2026-11-08	对照现网核对：补 API Key / 分组路径与示例；Webhook 验签与 outbox；事件 `payload_json` 样例与留存说明；fan-out 下发人员示例；目录
1.1.0	2026-11-08	增加系统总览与通讯关系说明；各章补充 cURL 示例与场景举例；补充设备解绑；增加修订记录；交付 PDF
1.0.0	2026-11-07	首版公开说明书：认领、事件、Webhook/MQTT、标准命令、fan-out、错误码摘要

后续若仅修正笔误，修订记录会增加补丁版本；若接口行为变更，会升次版本并在对应章节注明。

目录

1. 系统总览
2. 环境与鉴权（含 API Key 管理）
3. 统一错误格式
4. 设备
5. 事件
6. Webhook / MQTT
7. 标准命令
8. 按序列号批量 fan-out
9. 设备分组与 Scoped Key
10. 限流与日配额
11. 推荐接入流程

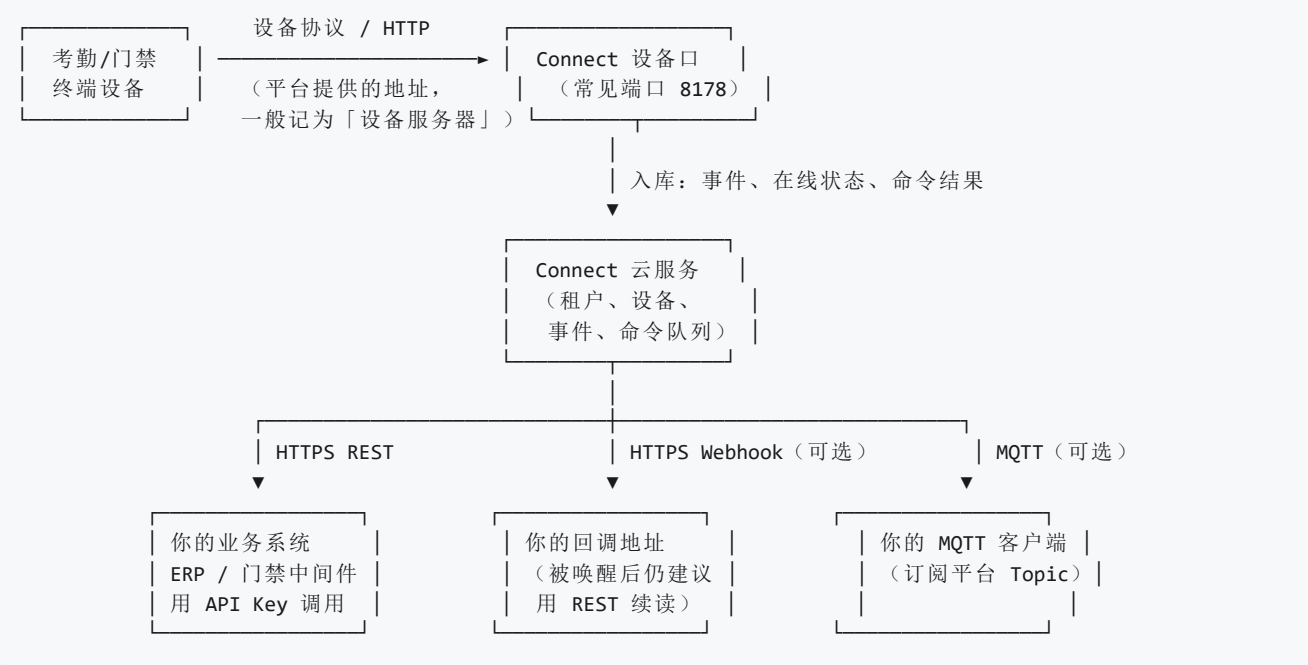
- 附录 A 错误码 · 附录 B 凭证 kind · 附录 C 支持 · 附录 D 接口速查

---

1. 系统总览（先读本节）

1.1 Connect 在整体架构中的位置

可以把它理解成「设备和你的业务系统之间的云端适配层」：



两条线不要混：

线路	谁连谁	你需要做什么
设备线	终端 → Connect 设备服务器（示例：`http://api.bnetlink.com:8178/`）	在设备菜单里填平台给出的地址；**不用**在你的代码里拼设备报文
业务线	你的服务器 → Connect REST（示例：`https://api.bnetlink.com/connect/v1`）	用 **Bearer API Key** 调本文各接口

门户（浏览器）给业务人员做认领、看事件、试命令；**正式对接请用你自己的后端 + API Key**，不要把 Key 写进公开网页。

1.2 设备通讯协议（接入方视角）

终端机使用厂商约定的设备协议（如动态机 BS、普通机 FK 等）与 Connect 的 **设备服务器** 通讯。Connect 负责：

1. 识别序列号、归属租户；
2. 把打卡 / 登记等上报整理成统一 **事件 JSON**；
3. 把你通过 REST 下发的**标准命令**翻译成设备能执行的报文，并排队、回写结果。

因此接入文档里出现的是标准名，例如 `set-user`、`delete-user`，而**不是**设备内部的命令码字符串。你只需要：

- 认领序列号 `sn`；
- 拉事件 / 收 Webhook 或 MQTT；
- 调标准命令并轮询 `command_id` 状态。

1.3 典型生命周期（举例）

举例：门店新装一台人脸机，接到你们的考勤中间件。

1. 厂商出厂时已把该序列号写入「可售库存」，并绑定到你的租户。
2. 你（或门户）调用 `POST /devices`，`sn=ABC123` → **认领成功。**
3. 设备配置里填写设备服务器地址并联网 → 开始心跳 / 上报。
4. 员工刷脸 → Connect 产生 `punch.recorded` 事件 → 你可用 `GET /events` 拉到，或先被 Webhook/MQTT 叫醒再拉。
5. 你从 ERP 把员工「1001」下发：`POST /devices/ABC123/commands/set-user`。
6. 轮询 `GET /commands/{command_id}`，直到 `SUCCEEDED` 或 `FAILED`。
7. 若门店撤机：`DELETE /devices/ABC123` **解绑**，序列号退回未开封，以后可再认领；历史打卡事件仍保留。

1.4 Connect 做什么 / 不做什么

做：

- 认领 / 停用 / 启用 / 解绑设备；
- 接收打卡、登记变更等事件；
- 下发标准命令（对时、开门、人员、通行时段等）；
- 可选 Webhook、可选 MQTT 推送。

不做：

- 排班、算薪、考勤规则引擎（那是业务系统或其它产品的事）；
- 替你保存「读到哪一条事件」的游标（请存在你自己的库）；
- 跨固件自动转换人脸 / 指纹模板格式。

1.5 接入方日常只需关心的三块

能力	说明
设备	认领 `sn`、列表、停用 / 启用、解绑
事件	拉取；可选 Webhook 和/或 MQTT 作唤醒或旁路
命令	对时、查状态、开门、人员、时段；可按序列号列表批量（fan-out）

---

2. 环境与鉴权

2.1 Base URL

环境	REST 前缀（示例）
生产	由合同 / 交付提供，形如 `https://api.bnetlink.com/connect/v1`
本地联调（若开通）	`http://127.0.0.1:8080/connect/v1`

下文路径均相对此前缀。例如 `GET /me` = `GET {base}/me`。

设备服务器（给终端填）示例：`http://api.bnetlink.com:8178/`（以交付为准）。

2.2 鉴权

除 `GET /health` 外，请求头必须带：

```
Authorization: Bearer {API_KEY}
Content-Type: application/json
Accept: application/json
```

说明：

- 一把 API Key 对应一个**租户**，只能访问本租户数据。
- Key 在门户「密钥」页创建，或由平台开户时发放；**明文只出现一次**，请妥善保存。
- **Webhook 签名密钥**、**MQTT 订阅密码**都与 API Key **不是同一串**。
- 门户登录账号（登录名 + 密码）只用于浏览器操作，**不能**代替 API Key 调 REST。

2.3 健康检查（示例）

```
curl -s "https://api.bnetlink.com/connect/v1/health"
```

成功：

```
{ "status": "ok" }
```

2.4 当前租户（示例）

```
curl -s "https://api.bnetlink.com/connect/v1/me" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

成功时响应中含 `tenant_id`、套餐、Key 作用域等（字段以实际为准）。

举例： 若返回的 `scope_type` 为分组范围（`GROUPS`），则部分账号级接口（如认领、解绑、管理 Webhook / 密钥 / 分组）会返回 `403` `forbidden_scope`。

2.5 API Key 管理（REST）

门户「密钥」页底层即下列接口。创建时明文 **只返回一次。**

```
# 列表（无明文）
curl -s "https://api.bnetlink.com/connect/v1/credentials" \
  -H "Authorization: Bearer YOUR_API_KEY"

# 创建整户 Key
curl -s -X POST "https://api.bnetlink.com/connect/v1/credentials" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"name\":\"erp-prod\",\"scope_type\":\"ACCOUNT\"}"

# 创建限定分组的 Key（group_ids 须为本租户 ACTIVE 分组）
curl -s -X POST "https://api.bnetlink.com/connect/v1/credentials" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"name\":\"store-a\",\"scope_type\":\"GROUPS\",\"group_ids\":[1,2]}"

# 停用 / 重新启用（status: REVOKED | ACTIVE）
curl -s -X PUT "https://api.bnetlink.com/connect/v1/credentials/2" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"status\":\"REVOKED\"}"
```

创建成功示例（请立即保存 `api_key`）：

```
{
  "api_key": "epc_live_…",
  "key_prefix": "epc_live_…",
  "scope_type": "ACCOUNT",
  "warning": "Store api_key now; it will not be shown again."
}
```

注意：不能停用租户下**最后一把** `ACTIVE` Key（`409` `last_active_api_key`）。分组 Key **不能**登录门户。

---

3. 统一错误格式

```
{
  "error": {
    "code": "device_not_found",
    "message_key": "connect.error.device_not_found",
    "message": "device_not_found"
  }
}
```

HTTP	含义（摘要）
400	参数不合法
401	未带 Key / Key 无效或已停用
403	权限不足（如分组 Key 调了账号级接口）
404	资源不存在或不属于本租户
409	冲突（重复认领、设备已停用、幂等冲突等）
429	分钟限流或日命令配额触顶

举例： 对已停用设备下发开门 → HTTP `409`，`code` = `device_disabled`。

常用 `code` 见 附录 A。

---

4. 设备

4.1 认领设备

```
POST /devices
```

```
{ "sn": "TEST-SN-001", "alias": "前台闸机" }
```

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/devices" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"sn\":\"TEST-SN-001\",\"alias\":\"前台闸机\"}"
```

成功示例：

```
{
  "sn": "TEST-SN-001",
  "connect_device_id": 12,
  "register_status": "ACTIVE",
  "protocol_track": "BS",
  "model_code": "P5",
  "firmware_version": "1.02"
}
```

说明：

- 只能认领**厂商库存中已绑定本租户、且尚未开封**的序列号。
- 协议 / 型号 / 固件来自厂商预置，请求体**不能**伪造。型号是短名 `P5` / `P7`。设备上报的 `dev_model` 不改这个值。
- 未知序列号 → `404` `device_not_in_stock`。
- 已分配给其它租户 → `403` `device_not_for_tenant`。
- 本租户已认领 → `409` `device_already_registered`。
- 对外字段名一律用 `sn`。

举例： 包装盒标签是 `B515…`，但厂商尚未把该号写入你的库存 → 认领会 `device_not_in_stock`，需联系厂商预置后再试。

4.2 设备列表 / 详情

```
curl -s "https://api.bnetlink.com/connect/v1/devices?limit=50" \
  -H "Authorization: Bearer YOUR_API_KEY"

curl -s "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

列表支持游标： `next_cursor` 不透明，下一页原样放进 `cursor=`。

4.3 停用 / 启用

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001/disable" \
  -H "Authorization: Bearer YOUR_API_KEY"

curl -s -X POST "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001/enable" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

- **停用**：仍属于本账号，但不能再下发命令（`409` `device_disabled`）。设备上报事件通常仍可入库（以 registry 为准）。
- **启用**：恢复可下发命令。
- 停用 ≠ 解绑。

4.4 解绑设备

```
curl -s -X DELETE "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

成功示例：

```
{ "sn": "TEST-SN-001", "claim_status": "UNCLAIMED" }
```

效果：

- 设备从本账号移除；厂商库存回到「未开封」，**可再次认领**；
- 平台侧该机人员登记清除；排队中的命令会取消；
- **历史打卡事件保留**；
- 分组范围的子 Key **不能**调用（`403`）。

4.5 设备上的人员与凭证（平台侧当前登记）

方法	路径	说明	
GET	`/devices/{sn}/users`	人员列表；可选 `scope=device\`	compatible、q`
GET	`/devices/{sn}/users/{userId}`	人员详情与凭证元数据	
GET	`/devices/{sn}/users/{userId}/credentials/{kind}/{slot}`	凭证本体（`data_base64`）	

```
curl -s "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001/users?limit=50" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

`kind` 仅允许：`photo`、`face`、`fingerprint`、`palm`、`card`、`password`（见附录 B）。

---

5. 事件

Connect **不保存**你的读游标。请在自己的库记录「已处理到的 `event_id` / 时间」。

5.1 拉取事件

```
curl -s "https://api.bnetlink.com/connect/v1/events?limit=50" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

按入库时间拉最近 1 天（示例，请换成真实 ISO-8601）：

```
curl -s "https://api.bnetlink.com/connect/v1/events?received_from=2026-11-07T00:00:00Z&received_to=2026-11-08T
  -H "Authorization: Bearer YOUR_API_KEY"
```

规则摘要：

- 时间参数为 ISO-8601；起止成对时跨度不超过 **7 天**。
- `from` / `to` = 业务发生时间；`received_from` / `received_to` = 平台入库时间。
- 列表项含 **`payload_json` 字符串**（业务 JSON 的文本形式），**不含**抓拍大图。
- 平台按租户配置保留热事件：默认约 **7 天**，可配置范围约 **3–14 天**（以合同 / 平台配置为准）；时钟按 **`received_at`**。过期后 `GET /events/{id}` 可能变为找不到。请及时落库。
- 列表查询里「每一对时间窗最长 7 天」是**查询窗口**限制，与磁盘留存天数是两件事。

列表单项形状（示意，字段名与现网一致）：

```
{
  "event_id": "evt_a1b2c3d4e5f6478990abcdef12345678",
  "event_type": "punch.recorded",
  "occurred_at": "2026-10-29T14:30:00",
  "received_at": "2026-10-29T14:30:01.123+00:00",
  "connect_device_id": 12,
  "sn": "TEST-SN-001",
  "payload_json": "{\"event_id\":\"evt_a1b2…\",\"sn\":\"TEST-SN-001\",\"user_id\":\"1001\",\"occurred_at\":176
}
```

请对 `payload_json` 做一次 `JSON.parse`。注意外层是 **`event_type`**，不是 `type`。Webhook / MQTT 推送的 HTTP body 是 **`payload_json` 解析后的业务 JSON 本身**（不是整行列表包装）。

举例： Webhook 半夜失败漏了 20 条。天亮后用 `received_from` 覆盖昨晚时段分页拉取，按 `event_id` 去重写入业务库即可补齐。

5.2 单条事件与抓拍

```
curl -s "https://api.bnetlink.com/connect/v1/events/{eventId}" \
  -H "Authorization: Bearer YOUR_API_KEY"

curl -s "https://api.bnetlink.com/connect/v1/events/{eventId}/photo" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

仅当打卡已入库抓拍（常见 `photo_status=stored`）时照片接口可用。响应是 **JSON**，不是直接的图片字节流，例如：

```
{
  "event_id": "evt_…",
  "content_type": "image/jpeg",
  "byte_length": 12345,
  "sha256": "…",
  "data_base64": "/9j/4AAQ…"
}
```

请把 `data_base64` 解码后落盘。无照片 → `404` `photo_not_found`；事件不存在 → `404` `event_not_found`。列表 / 详情本身不带图片字节。

5.3 常见事件类型

event_type	含义
`punch.recorded`	打卡
`enroll.data.received`	人员登记 / 凭证变更摘要
`command.completed` / `command.failed`	命令终态
`door.state.changed`	门磁状态变化（设备主动上报）
`door.remote.recorded`	远程开关门（`mode`：pulse/hold/close；非打卡）
`device.presence`	设备上线/离线边沿（`online`：1/0）

新写入只用上表类型。`GET /devices` 的 `online`：`last_seen_at` 在 **10 分钟**内为 `1`，否则 `0`。`device.presence` 只在边沿各写一条：上线由设备上行触发；离线由后台约每 **5 分钟**扫描，断线后大约 **10 ~ 15 分钟**出现。该事件的 `occurred_at` 是最后一次上行，`received_at` 是记下边沿的时刻。旧名 `device.online` / `device.offline`、`door.open.recorded` / `door.close.recorded` 仍可筛选或订阅，但只匹配对应边沿。

5.3.1 打卡 `payload_json` 字段（解析后）

字段	类型	说明
`event_id`	string	事件 id，去重主键
`sn`	string	设备序列号
`user_id`	string	人员编号
`occurred_at`	number	业务发生时间，**毫秒**时间戳
`device_time`	string	设备本地时间串（常见 `yyyyMMddHHmmss`）
`verify_mode`	string	验证方式描述（如 Face / Card）
`door_mode`	string	可选。设备开门方式原文（如 `hand_open` / `open`）
`door_opened`	boolean	**P5 / BS**：设备上报了非空 `doorMode` 时为 `true`（已驱动锁），缺失或空白为 `false`。只给结果，不含拒绝原因。`remote_open` / `remote_close` 不是打卡
`has_log_photo`	boolean	是否声明有抓拍
`photo_status`	string	`absent`、`stored`、`omitted_too_large` 或 `omitted_invalid`

5.4 最新事件摘要（可选）

```
curl -s "https://api.bnetlink.com/connect/v1/sync/status" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

用于粗看是否有新事件；**不是**水位游标，不能替代你自己的续读位点。

---

6. Webhook（可选）

注册 HTTPS 回调后，有事件时 Connect 会向你 POST（带签名）。

推荐： Webhook 只作**唤醒**，权威数据仍以 `GET /events` 续读为准。

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/webhooks" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"url\":\"https://example.com/hooks/connect\",\"secret\":\"at-least-16-chars!\",\"event_types\":[]}"
```

其它：`GET /webhooks`（列表）、`PUT /webhooks/{id}`、`POST /webhooks/{id}/test`。

没有单独的 `GET /webhooks/{id}`：请从列表结果中按 `id` 取详情字段。

`PUT` 可改 `url` / `event_types` / `status`（`ACTIVE` | `DISABLED`）；改 `secret` 时填写新值，**省略则保留原 secret。** `event_types` 为空表示订阅全部类型。

要点：

- 生产公网 URL 须 **HTTPS**，不可指向私网 / 回环。
- `secret` 16–128 位，自拟；**不要**把 API Key 当作 Webhook secret。
- 列表 / 详情**不再返回** secret 明文，创建时请自行保存。
- 回调 HTTP body = 事件业务 JSON（与 `payload_json` 解析后相同），**不是**照片。

验签（必做）

Connect 使用 HMAC-SHA256：

```
签名字符串 = timestamp + "." + raw_body
X-Epordo-Signature = hex_lower( HMAC_SHA256(secret, 签名字符串) )
```

请求头还包括：`X-Epordo-Timestamp`、`X-Epordo-Event-Id`、`X-Epordo-Event-Type`。

接收端应用**原始 body 字节**验签，返回任意 **2xx** 表示成功；失败会重试，最终可能变为 `DEAD`。

举例（伪代码）：

```
expected = hmac_sha256_hex(secret, header["X-Epordo-Timestamp"] + "." + rawBody)
if expected != header["X-Epordo-Signature"]: reject
else: upsert_by_event_id(JSON.parse(rawBody)); return 200
```

投递队列（排障）

方法	路径	说明
GET	`/webhooks/outbox`	队列行：`PENDING` / `FAILED` / `SENT` / `DEAD`
GET	`/webhooks/deliveries`	每次尝试的审计（状态码、耗时、错误摘要）
POST	`/webhooks/outbox/{id}/replay`	仅 `FAILED` / `DEAD` 可重放

举例： 收到 Webhook 后只记录「有新数据」，再立刻用上次保存的时间窗调 `GET /events`；这样即使某次回调丢失，也不会丢业务数据。

6.1 MQTT（可选，可与 Webhook 同时开）

Connect 可把与 Webhook **相同**的事件 JSON 发到固定 Topic。可与 Webhook **并行**；请用 `event_id` 去重。

项	说明
Topic	`epc/{tenant_id}/events/{event_type}`（平台固定）

载荷	与事件 / Webhook 相同的业务 JSON
QoS	0 或 1（默认 1）
Broker	由平台提供；在门户 MQTT 页**签发订阅账号**获取用户名 / 密码
连接	服务端常用 TCP；浏览器常用 WSS（具体地址见门户提示）

```
# 登记要推送的事件（每租户一条）
curl -s -X POST "https://api.bnetlink.com/connect/v1/mqtt" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"event_types\":[],\"qos\":1}"

# 签发订阅账号（密码只显示一次）
curl -s -X POST "https://api.bnetlink.com/connect/v1/mqtt/credentials" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

说明：订阅账号**不是** API Key，也**不是** Webhook secret。设备通讯仍走设备服务器，**不会**改成设备直连 MQTT。

---

7. 标准命令

7.1 创建与查询

```
# 下发（以对时为例）
curl -s -X POST "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001/commands/set-time" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{}"

# 查询状态
curl -s "https://api.bnetlink.com/connect/v1/commands/{command_id}" \
  -H "Authorization: Bearer YOUR_API_KEY"

# 仅 QUEUED 可取消
curl -s -X POST "https://api.bnetlink.com/connect/v1/commands/{command_id}/cancel" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

创建成功时常见字段：

```
{
  "command_id": "cmd_a1b2c3d4e5f6",
  "sn": "TEST-SN-001",
  "command": "set-time",
  "state": "QUEUED"
}
```

字段	说明
`command_id`	如 `cmd_` + 十六进制
`state`	`QUEUED` → `SENT` → `SUCCEEDED` / `FAILED` / `CANCELLED`
`result`	终态时可能有结果对象（视命令而定）

- 可选头 `X-Idempotency-Key`（≤128）：相同 Key + 相同体 → 返回原命令；相同 Key + 不同体 → `409` `idempotency_conflict`。

- 设备在线时：普通命令约 3 分钟无进展会失败；分包回读包间空闲约 10 分钟。同机同时最多一条 `SENT`。
- 设备离线时命令会留在 `QUEUED`，待上线后下发（仍受超时与配额约束）。

举例： 网络抖动导致你重试了两次开门。两次使用同一个 `X-Idempotency-Key` 和同一 JSON，则只会真正入队一次，避免门开两次。

7.2 命令一览

标准名	路径后缀	作用
`set-time`	`set-time`	对时
`query-status`	`query-status`	查设备状态
`unlock`	`unlock`	远程开门 / 常开 / 关门
`set-user`	`set-user`	下发一人登记（含可选模板）
`delete-user`	`delete-user`	删除一人或多人
`list-user-ids`	`list-user-ids`	回读人员编号列表
`get-user`	`get-user`	回读人员并落平台登记
`set-access-schedule`	`set-access-schedule`	通行时段
`clear-managers`	`clear-managers`	清管理员标志（不删人）
`pull-logs`	`pull-logs`	按时间窗补拉打卡

当前**不提供**：远程注册界面、响铃设置等。请勿对接。

7.3 `set-time`

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001/commands/set-time" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"time\":\"20261108120000\"}"
```

`time` 可选，格式 `yyyyMMddHHmmss`；省略则用服务端取令时的本地时钟。

7.4 `query-status`

无正文。成功时 `result.model` 是这次通讯的 Header `dev_model`（P5 与 P7 都是 `P5`），不是设备上的销售型号 `model_code`。另有固件、容量等（设备不支持的项可能为 `null`）。

7.5 `unlock`

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001/commands/unlock" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"mode\":\"pulse\"}"
```

mode	含义
`pulse`（默认）	开一下再按设备延时关闭
`hold`	保持开门

`close`	关门

`SUCCEEDED` 只表示设备确认执行，**不等于**门磁已开。

7.6 `set-user`（单人 · 单机）

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001/commands/set-user" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"user_id\":\"1001\",\"name\":\"张三\",\"role\":\"user\",\"card\":\"123456\"}"
```

完整字段示例：

```
{
  "user_id": "1001",
  "name": "张三",
  "role": "user",
  "card": "123456",
  "password": "666",
  "valid_from": "20260101",
  "valid_to": "20261231",
  "replace": false,
  "face": "<Base64>",
  "photo": "<Base64>",
  "fingerprint": ["<Base64>"],
  "palm": ["<Base64>"]
}
```

- `user_id` 必填；`fingerprint` / `palm` 最多 10 槽，**空数组表示清空该模态。**
- Connect **不做**模板格式转换。
- 通行时段请用 `set-access-schedule`，不要塞进本命令。

7.7 `delete-user`（可多人 · 单机）

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001/commands/delete-user" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"user_ids\":[\"1001\",\"1002\"]}"
```

- 一条命令可删多人（最多 100）；不要同时传 `user_id` 与 `user_ids`。
- **不做**「清空整机全部人员」。

7.8 `list-user-ids` / 7.9 `get-user`

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001/commands/list-user-ids" \
  -H "Authorization: Bearer YOUR_API_KEY" -H "Content-Type: application/json" -d "{}"

curl -s -X POST "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001/commands/get-user" \
  -H "Authorization: Bearer YOUR_API_KEY" -H "Content-Type: application/json" \
  -d "{\"user_ids\":[\"1001\"]}"
```

- `get-user` 也可 `{"all":true}`（静默落库，避免事件风暴）。
- 指定 id 落库后通常会有 `enroll.data.received`。

7.10 `set-access-schedule`

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/devices/TEST-SN-001/commands/set-access-schedule" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"time_groups\":{\"mon\":[\"09001200\",\"14001800\"],\"tue\":[\"09001800\"]},\"user_ids\":[\"1001\",\"1
```

字段	说明
`time_groups`	必填。键 `sun` … `sat`；值是 `HHmmHHmm` 字符串数组，**每天 1–6 段**
`user_ids`	可选；省略 / 空 = 设备上全部人员

时段只存在设备上。核对可再发 `get-user` 看 `access_schedules`。

举例： 周一上午 09:00–12:00、下午 14:00–18:00 → `"mon": ["09001200", "14001800"]`。

7.11 `clear-managers` / `pull-logs`

- `clear-managers`：无正文；清管理员标志。
- `pull-logs`：`{"from":"…","to":"…"}`（8 或 14 位，跨度 ≤31 天）。

---

8. 按序列号列表批量（fan-out）

```
POST /commands/fan-out/set-user
POST /commands/fan-out/delete-user
POST /commands/fan-out/set-access-schedule
```

公共字段：`sns`（非空字符串数组）。

8.1 原则

同一台设备上，相同操作尽量打进**一条**命令（与设备能力一致），而不是「每人一条 HTTP」。

接口	推荐写法	队列行为
下发人员	`users:[{user_id,…}]`	**每 SN 一条**
删除人员	`user_ids:[…]`	**每 SN 一条**
通行时段	共享 `time_groups` + 可选 `user_ids`	**每 SN 一条**

8.2 上限（默认）

项	默认
不同 `sn` 数	60（大模板时可能降到 30）
人员 id 数	100
产生的队列命令数	200

超限 → `400` `fan_out_too_large`。

8.3 多机下发人员（示例）

推荐用 `users[]`（每人可有独立姓名 / 卡号 / 模板），**每个 SN 只入队一条**命令：

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/commands/fan-out/set-user" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"sns\":[\"SN-A\",\"SN-B\"],\"users\":[{\"user_id\":\"1001\",\"name\":\"张三\",\"card\":\"123\"},{\"use
```

人脸等大模板可能超过单机缓冲，请分页或缩小模板后再发。

8.4 删除多人多机（示例）

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/commands/fan-out/delete-user" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d "{\"sns\":[\"SN-A\",\"SN-B\"],\"user_ids\":[\"1001\",\"1002\"]}"
```

响应形状：

```
{
  "command": "delete-user",
  "queued": [
    { "sn": "SN-A", "user_ids": ["1001", "1002"], "command_id": "cmd_…" }
  ],
  "skipped": [
    { "sn": "SN-X", "reason": "device_disabled" }
  ]
}
```

请分别轮询每个 `command_id`。`skipped.reason` 常见：`device_not_found`、`device_disabled`、配额相关。

8.5 多机同一套通行时段（示例）

```
{
  "sns": ["SN-001", "SN-002"],
  "time_groups": {
    "mon": ["09001200", "14001800"],
    "tue": ["09001800"]
  },
  "user_ids": ["1001", "1002", "1003"]
}
```

---

9. 设备分组与 Scoped Key

路径前缀：`/device-groups`。创建 / 改名 / 启停分组需要整户（ACCOUNT）Key。

没有 `DELETE /device-groups/{id}`：停用请 `PUT` 将 `status` 设为 `DISABLED`。

方法	路径	说明
GET	`/device-groups`	列表
POST	`/device-groups`	创建：`{"code":"store-a","name":"A店"}`

GET	`/device-groups/{id}`	详情 + 成员设备	
PUT	`/device-groups/{id}`	改 `name` / `status`（`ACTIVE` \	`DISABLED`；不可改 `code`）
PUT	`/device-groups/{id}/devices/{sn}`	加入已认领设备	
DELETE	`/device-groups/{id}/devices/{sn}`	移出成员	
POST	`/device-groups/{id}/commands/set-time`	组内批量对时	
POST	`/device-groups/{id}/commands/query-status`	组内批量查状态	
POST	`/device-groups/{id}/commands/unlock`	组内批量开门（`mode` 同单机）	

```
curl -s -X POST "https://api.bnetlink.com/connect/v1/device-groups" \
  -H "Authorization: Bearer YOUR_API_KEY" -H "Content-Type: application/json" \
  -d "{\"code\":\"store-a\",\"name\":\"A店\"}"

curl -s -X PUT "https://api.bnetlink.com/connect/v1/device-groups/1/devices/TEST-SN-001" \
  -H "Authorization: Bearer YOUR_API_KEY"

curl -s -X POST "https://api.bnetlink.com/connect/v1/device-groups/1/commands/query-status" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

组级批量响应恒为 **200**（允许部分成功），形状类似 fan-out：`queued[]` / `skipped[]`。

说明：

- 人员 / 删除 / 通行时段 **没有**组级批量，请用 §8 fan-out（显式 `sns`）。
- 可为分组签发 `scope_type=GROUPS` 的 API Key（见 §2.5）。
- 分组 Key **不能**登录门户，也不能认领 / 解绑 / 管理 Webhook、MQTT、密钥、分组本身。
- 组状态为 `DISABLED` 时，组级下发可能 `409` `group_disabled`。

---

10. 限流与日配额（摘要）

- **分钟级**：按 Key 请求速率，触顶 `429` `rate_limited`（响应可能带 `Retry-After`）。
- **日配额（UTC 日）**：开门、全量回读、补拉、下发 / 删人、时段等「贵命令」；异常用量可能导致 Key 自动停用（`quota_exceeded_key_revoked`）。
- 幂等重放通常**不再扣**配额。具体数字以合同 / 平台配置为准。

---

11. 推荐接入流程

11.1 最小闭环（打卡）

1. 取得 API Key，`GET /me` 确认。
2. `POST /devices` 认领；设备填写设备服务器地址并上线。
3. （可选）注册 Webhook 或 MQTT。
4. 周期或被唤醒后 `GET /events` 续读；自行存进度。
5. 需要抓拍再 `GET /events/{id}/photo`。

11.2 人员下发

1. 业务侧维护人员与模板。
2. 单机 `set-user`；多机 `fan-out/set-user`。
3. 同一套通行规则：`set-access-schedule` 或 fan-out 共享 `time_groups`。
4. 核对：`list-user-ids` / `get-user`。

11.3 同系列机群

- 销售型号（P5、P7）仍然分开。
- 不要对每台都做全量 `get-user`。
- 以 ERP 或一台代表机为权威；其它机 `list-user-ids` 对账后补洞。
- 兼容列表 `GET /devices/{sn}/users?scope=compatible`：厂商在库存里写成同一个模板族，并且固件相同、非空时才会跨机列出。设备上报的 `dev_model` 不改变销售型号。固件为空则不跨机。

附录 A. 常用错误码

code	典型场景
`missing_bearer` / `empty_key` / `invalid_key`	鉴权失败
`missing_sn`	认领未带序列号
`device_not_in_stock`	序列号不在厂商库存
`device_not_for_tenant`	库存未分配给本账号
`device_not_found`	设备不存在或不在权限内
`device_already_registered` / `sn_already_registered`	重复登记
`device_disabled`	设备已停用
`invalid_set_user` / `invalid_delete_user` / `invalid_access_schedule`	命令体不合法
`invalid_fan_out` / `fan_out_too_large`	批量参数错误或超限
`command_not_found` / `command_not_cancelable`	命令查询 / 取消
`idempotency_conflict`	幂等键冲突
`rate_limited` / `quota_exceeded` / `quota_exceeded_key_revoked`	限流 / 配额
`invalid_webhook_url` / `invalid_webhook_secret`	Webhook 参数
`forbidden_scope`	分组 Key 调用了账号级接口
`last_active_api_key`	试图停用最后一把有效 Key
`api_key_not_found`	密钥 id 不存在
`mqtt_already_exists`	租户已登记 MQTT 出站，勿重复 POST
`outbox_not_replayable`	投递队列状态不可重放
`group_not_found` / `group_disabled`	分组不存在或已停用

附录 B. 凭证 kind

kind	槽位	说明
`photo`	0	可见光照片
`face`	0	人脸模板
`fingerprint`	0～9	指纹
`palm`	0～9	掌纹 / 掌静脉
`card`	0	卡号（UTF-8）
`password`	0	设备密码（UTF-8）

单块解码后超过约 **1MB** 会被拒绝。平台不做跨固件模板转换。

附录 C. 文档与支持

资料	说明
本文 / PDF	对外中文接口说明书（门户「帮助」可下载 PDF）
门户	浏览器认领、命令试跑、签发 MQTT 订阅账号等
ApiPost / Postman	由项目交付的导入集合（技术附件）
问题反馈	商务 / 技术支持渠道

附录 D. 接口速查

分类	方法与路径			
探活 / 身份	`GET /health` · `GET /me` · `GET	POST /credentials · PUT /credentials/{id}`		
设备	`POST	GET /devices · GET	DELETE /devices/{sn} · POST .../disable	enable · GET .../users...`
事件	`GET /events` · `GET /events/{id}` · `GET /events/{id}/photo` · `GET /sync/status`			
Webhook	`GET	POST /webhooks · PUT /webhooks/{id} · POST .../test · GET .../outbox	deliveries · POST .../outbox/{id}/replay`（无单独 GET by id）	

MQTT	`GET	POST /mqtt · PUT	DELETE /mqtt/{id} · POST .../test · GET	POST	DELETE /mqtt/credentials` · outbox/deliveries/replay
命令	`POST /devices/{sn}/commands/{name}` · `GET /commands/{id}` · `POST /commands/{id}/cancel`				
批量	`POST /commands/fan-out/set-user	delete-user	set-access-schedule`		
分组	`GET	POST /device-groups · GET	PUT /device-groups/{id} · 成员 PUT/DELETE · 组级 set-time / query-status / unlock`（无 DELETE 分组本身）		

标准命令 `{name}`：`set-time`、`query-status`、`unlock`、`set-user`、`delete-user`、`list-user-ids`、`get-user`、`set-access-schedule`、`clear-managers`、`pull-logs`。

文档结束 · EPORDO Connect 接口说明书 v1.1.5