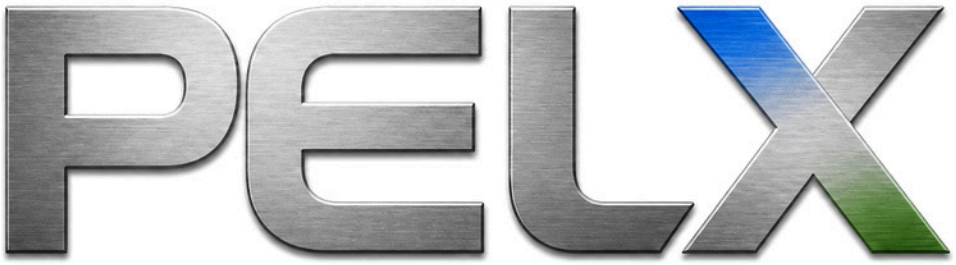


設備· 主機· 認證· 主動上報

自訂電子鎖整合

電子鎖主機整合指南

版本 1.0

目錄

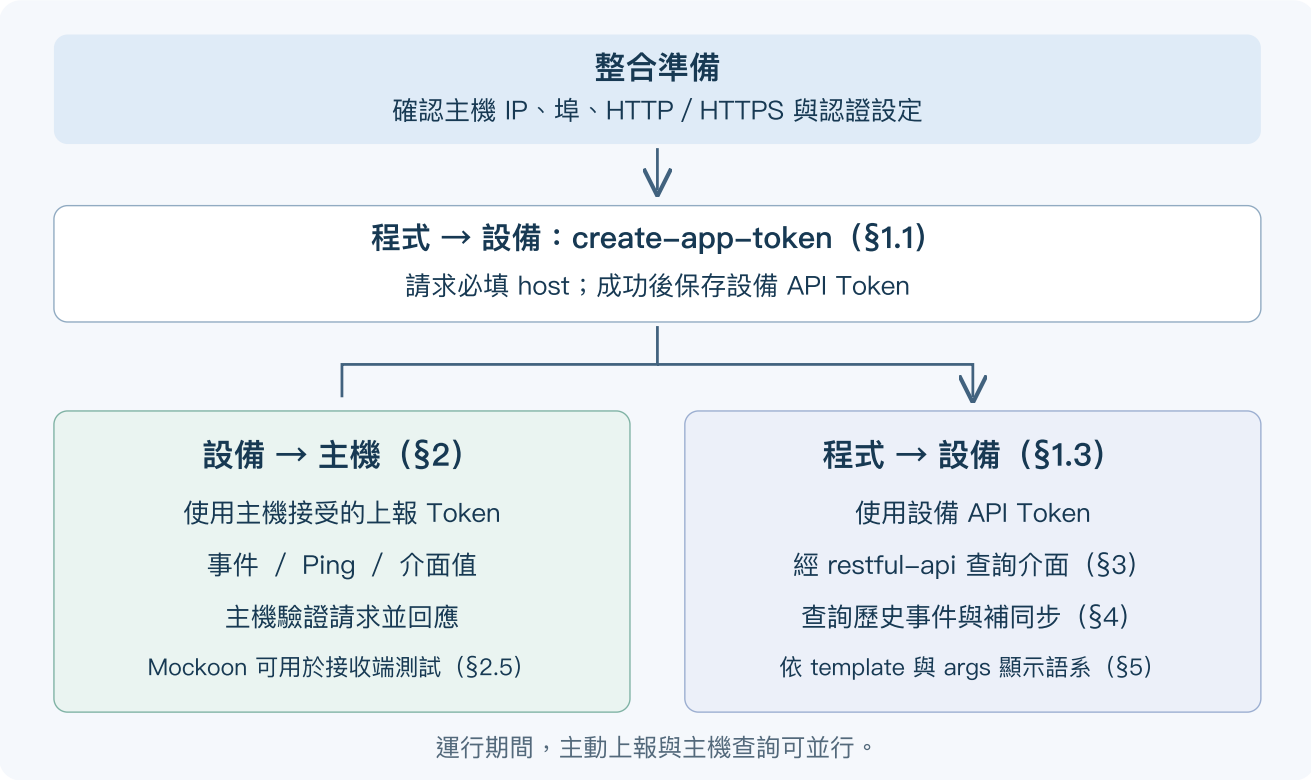
前言

本文件說明設備與自訂電子鎖主機整合時之常用項目;**主機上報 token** 與 **設備 API token** 用途不同,應分開管理(見 **§ 1.2**)。**向設備取得設備 API token** 見 **§ 1.1**。

1. **設備 > 主機** 運行期主動上報(事件、Ping、介面值)
2. **工程模式/週邊資訊**:介面位址 `addr` 與週邊對應畫面
3. **Mockoon**:本機模擬主機接收上報、Logs 除錯
4. **設備介面查詢**:經 **§ 1.3** 統一入口呼叫 **`hwctrl`** 的 **`GET /interfaces`**
5. **歷史紀錄**:經 **§ 1.3** 統一入口呼叫 **`history`** 的 **`GET /logs`**(時間查詢)、**`GET /logs/max-seq`**、**`GET /logs/sync`**(主機補足缺失, § 4.7~4.8)
6. **事件訊息語系**:模板與中英文對照見 **§ 5**

API 整合流程總覽

下列流程涵蓋「一次性設定」與「運行期持續上報」,並標示本文件各節之對應關係。



各步驟說明

步驟	誰發起	要做的事
1 主機連線資訊	開發商	確認主機 `ip`、`port`、HTTP/HTTPS 協定及憑證設定。埠號不會自動啟用 TLS;URL 為 `protocol://ip:port` 加上各上報路徑。
2 取得 app-token	第三方程式 → **設備**	向設備 POST `.../create_app_token`(帳密與必填 `host`),取得 `accessToken`、`urlToken`;詳見 **§1.1**。
3 保存 Token 與認證	第三方程式	**呼叫設備**時使用回應 body 之 `accessToken`(或 `appToken`)作為 `Authorization: Bearer ...`。**設備 > 主機**上報所用 Token 見 **§1.2**,兩者勿混用。
4 送事件/Ping/介面值	設備/程式 → **主機**	依時機或週期送 `eventUrl`、`pingUrl`、`ifaceUrl`;格式見 **§2**。
5 主機回應與驗證	主機/開發商	驗證 Token、回傳狀態碼與 JSON;本機可用 Mockoon 驗證,見 **§2.5**。
6 查詢介面與紀錄	第三方程式 → **設備**	經 **§1.3** 統一入口呼叫 `GET /interfaces`、`GET /logs`、`GET /logs/max_seq`、`GET /logs/sync` 等,見 **§3**、**§4**。

運行期: 4 與 5 可並行。**設備 > 主機**上報使用 **主機上報 token**(**§1.2**);**程式 > 設備**使用 **設備 API token**(**§1.3**),**兩者用途不同**。

範疇說明

類別	內容	本文件
Token 與主機導向	`create_app_token`、必填 `host`	**§1.1~1.2**
A. 主機接收設備資料	事件、Ping、介面值	**§2**
B. 查詢與介面	介面列表、歷史紀錄、主機補件(seq)	**§3**、**§4**
C. 訊息語系	事件模板與語系	**§5**

建議順序:**先完成 §1.1(`create_app_token`)與必填 `host`**,再完成 **A(上報鏈路)**,最後串 **B** 查詢與 **C** 顯示。

實機核對範圍

本次以 2026-09-17 的設備進行唯讀核對;`hwctrl` 與 `history` 回報版本 `0.0.1`、建置編號 `202609172144`。已確認一般登入、介面列表、事件查詢與補同步、通知語系查詢的回應。

核對先使用一般登入後的 `/api/hwctrl/...` 與 `/api/history/...` 路徑,再使用測試 `host` 成功呼叫 `create-app-token`,取得 `accessToken`。已用該 Token 經 `restful-api` 測通歷史查詢、最大 seq、補同步、介面列表、告警彙總/明細及通知語系讀取;均回傳 HTTP 200,且 `session` 一致。尚未測試開鎖、設定 PATCH、金鑰刷新或主動上報接收;取得 Token 不代表測試 host 已具備接收能力。

第 1 章 自訂電子鎖整合

本章說明設備與自訂電子鎖主機的整合方式,包含認證、統一入口、主動上報、介面值、事件記錄與語系設定。

1. 認證與統一入口

1.1 `create-app-token`(程式 → 設備)

第三方程式向**設備**呼叫 `create-app-token`,主要取得**呼叫設備 API**(如 **§1.3** `restful-api`)所用之 **`accessToken`**。設備向主機上報所用的憑證則設定於 **`host.accessToken`**,由接收主機的認證規格決定。兩者用途不同,請分開保存;不要假設設備核發的 Token 可直接通過主機認證。

Request Headers

Header	值
`Content-Type`	`application/json`

1.1.1 Request Body(必填與選填)

此版本必須包含 `host`。 在建置版本 `202609172144`,只傳帳密、`expiration` 與 `refreshKey` 會回傳 HTTP 400,訊息為 `Missing required fields: host`。因此不能用省略 `host` 的方式取得應用程式 Token。

> **警告**:此 API 的請求包含主機上報設定。正式呼叫前,先確認要使用的主機位址、上報路徑及憑證;不要將下列示例值直接套用到運行中的設備。此次已經使用經同意的測試 host 建立 Token;測試值僅供驗證,正式上報前須替換為實際主機設定。

完整請求範例(主機欄位依整合環境填入)

```
{
  "username": "user123",
  "password": "123456",
  "expiration": 0,
  "refreshKey": false,
  "host": {
    "name": "host1",
    "ip": "192.168.1.100",
    "port": 443,
    "deviceId": "device1",
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "eventUrl": "/api/history/events",
    "pingUrl": "/api/devices/ping",
    "ifaceUrl": "/api/device/interfaces/values",
    "pingInterval": 900,
    "ifaceInterval": 900,
    "ifaceErrorInterval": 30
  }
}
```

欄位	必填	說明
username	是	設備登入帳號
password	是	設備登入密碼
expiration	否	Token 有效期限設定;本文件以 0 為範例,其單位及 0 的實際含義須依目標韌體確認。
refreshKey	否	本文件以 false 為範例;設為 true 的作用及是否使既有 Token 失效,須依目標韌體確認。
host	是	此版本不可省略;內含主機位址與上報設定;此次以 name、ip、port、deviceId 四個欄位成功建立 Token。上報路徑須與 §2 接收端路由一致。

host 內的欄位說明

實測只傳 ip 時回傳 400,指出缺少 name、port、deviceId;補齊這三項後,以 expiration: 0、refreshKey: false 成功回傳 200。此結果只確認 Token 建立成功,不代表上報接收流程已通過。

欄位	說明
name	主機名稱
ip	設備可連線的主機 IP;跨設備測試時不可填 localhost 或 127.0.0.1。
port	主機埠(例如 443)

欄位	說明
`deviceId`	設備識別碼(與上報 JSON 之 `deviceId` 對應)
`accessToken`(僅 **`host`** 內)	**主機上報 Token**。由主機或專案提供可接受的憑證後寫入;設備上報使用的 `Authorization: Bearer ...` 須符合主機約定。此欄位與回應的設備 API `accessToken` 用途不同;是否採用 JWT、是否允許省略,須確認主機與韌體規格。
`eventUrl` / `pingUrl` / `ifaceUrl`	三條上報路徑,須與主機或 Mockoon Route **完全一致**(含前綴、`ifaceUrl` 是否含 `/values`)
`pingInterval`	Ping 週期(秒),本文件範例為 `900`
`ifaceInterval`	介面值定時上報週期(秒),本文件範例為 `900`
`ifaceErrorInterval`	介面值上報失敗後重試間隔(秒),本文件範例為 `30`

- 使用 **Mockoon** 本機測試時,將 `host.ip`、`host.port` 設為 Mockoon 監聽位址與埠即可。

1.1.2 Response Body(成功時)

已實測 HTTP 200,回應包含 `accessToken`、`urlToken`、`username`、`fullName`、`role`。下列範例僅列整合使用的兩個 Token 欄位。

```
{
  "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "urlToken": "rH3Zkrpzr_Jb134hvz2q5FIwYg0f9QqNV1mfAvdvPu7..."
}
```

欄位	說明
`accessToken`	供 **程式 > 設備** 呼叫 `restful-api` 等 API 時,置於 `Authorization: Bearer ...`(此版本已實測欄位名為 `accessToken`;範例程式另容許舊版 `appToken`)。**不作為**設備 > 主機上報憑證;上報請依 **§1.2** 使用 `host.accessToken` 或韌體約定。
`urlToken`	中控網頁/URL Token 等流程使用(與本文件 **§3、§4** 無直接關聯時可僅保存)

200	成功(已實測取得 accessToken 與 urlToken)
400	請求缺少必填欄位;此版本已確認省略 host 會得到此回應。

1.1.3 curl 範例:取得 Token 並設為環境變數

下列與後續 curl 範例中的 `-k` 僅用於測試設備的自簽憑證;正式使用請配置信任憑證後移除,或改用 `--cacert <CA 檔案>`。範例帳密僅供示意,請換成實際帳號。後續 §3、§4 範例須在同一個 shell 中執行,沿用 `DEVICE_IP` 與 `ACCESS_TOKEN`。

Bash(macOS / Linux / Git Bash / WSL)

```
# host.json 為已核定的 host 物件，不含外層 host 鍵。
# 請勿直接使用手冊示例位址覆蓋現有主機設定。
DEVICE_IP="<設備-ip>"
USERNAME="<設備帳號>"
PASSWORD="<設備密碼>"

REQUEST_JSON=$(jq -en \
  --arg username "$USERNAME" --arg password "$PASSWORD" \
  --slurpfile host host.json \
  'if ($host | length) == 1 and ($host[0] | type) == "object"
   then {username:$username, password:$password,
         expiration:0, refreshKey:false, host:$host[0]}
   else error("host.json 必須包含一個 host 物件") end') || exit 1
TOKEN_JSON=$(curl -k -f -sS -X POST \
  "https://${DEVICE_IP}/app-token/create-app-token" \
  -H "Content-Type: application/json" \
  --data-binary "$REQUEST_JSON") || exit 1

ACCESS_TOKEN=$(printf '%s' "$TOKEN_JSON" | jq -er \
  '(.accessToken // .appToken) | select(type == "string" and length > 0)') || exit
        1
URL_TOKEN=$(printf '%s' "$TOKEN_JSON" | jq -r '.urlToken // empty')
AUTH_HEADER="Authorization: Bearer ${ACCESS_TOKEN}"
```

PowerShell(Windows)

```powershell
$DEVICE_IP = "<設備-ip>"
$USERNAME = "<設備帳號>"
$PASSWORD = "<設備密碼>"
$HOST_CONFIG = Get-Content -Raw -Encoding UTF8 .\host.json | ConvertFrom-Json

$BODY = @{
  username = $USERNAME
  password = $PASSWORD
  expiration = 0
  refreshKey = $false
  host = $HOST_CONFIG
} | ConvertTo-Json -Depth 10

$TOKEN_JSON = curl.exe -k -f -sS -X POST "https://$DEVICE_IP/app-token/create-app-
        token" `
  -H "Content-Type: application/json" `
  -d $BODY

if ($LASTEXITCODE -ne 0) { throw "取得設備 Token 失敗" }
$TOKEN_OBJ = $TOKEN_JSON | ConvertFrom-Json
$ACCESS_TOKEN = if ($TOKEN_OBJ.accessToken) { $TOKEN_OBJ.accessToken } else {
        $TOKEN_OBJ.appToken }
if (-not $ACCESS_TOKEN) { throw "回應未包含設備 API Token" }
$URL_TOKEN = $TOKEN_OBJ.urlToken

$AUTH_HEADER = "Authorization: Bearer $ACCESS_TOKEN"
```

取得 AUTH_HEADER(內含 **設備 API 用** accessToken)後,用於呼叫 **§ 1.3** restful-api。**設備 > 主機**上報之 Bearer 請依 **§ 1.2** 另備(多為 **host.accessToken**),勿逕自沿用 ACCESS_TOKEN。

1.2 請求情境與 Bearer 用途(摘要)

兩種請求方向有不同的驗證對象:設備驗證 **設備 API Token**,接收主機驗證 **主機上報 Token**。請使用不同變數保存,不要自行將一方的憑證用於另一方。

憑證用途對照

情境	Authorization 建議寫法	說明
程式 > 設備(restful-api 等)	`Bearer <deviceApiToken>`	**設備 API token**:即 `create-app-token` 成功回應 body 之 **accessToken**(舊版 **appToken**)。**僅**用於呼叫設備;**不是**主機上報 token。
設備 > 主機(上報)	`Bearer <hostReportToken>`	**主機上報 token**:主機要求設備在上報 URL 出示的憑證。須由主機或專案核發後寫入 **`host.accessToken`**。主機是否接受某一憑證,取決於主機的驗證設定;未配置時請確認主機/韌體規格,不要直接以 body 的 `accessToken` 代替。

1.3 統一入口 `restful-api`(摘要)

項目	說明
URL	`POST https://<設備-ip>/app-token/restful-api`
Header	`Content-Type: application/json`、`Authorization: Bearer <deviceApiToken>`(即 **§ 1.2** 之 **accessToken/appToken**)

Request Body 共通欄位

欄位	說明
`session`	追蹤識別字串
`serviceName`	`hwctrl` 或 `history`
`url`	目標服務路徑(GET 可含 query string)
`method`	GET / POST / PATCH / DELETE
`body`	目標服務的請求內容;GET 範例省略整個 body 欄位。

Response 共通格式

```
{
  "session": "<與請求相同>",
  "body": {}
}
```

HTTP 狀態碼	200 正常、400 參數錯誤、401 未授權、500 伺服器錯誤。
實測結果	使用 `create-app-token` 核發的 `accessToken`,以下 GET 查詢均經本統一入口回傳 HTTP 200,且回應 `session` 與請求一致:

- `history`:`/logs?offset=0&limit=2`、`/logs/max-seq`、`/logs/sync`(帶 seq 區間及 `lang=zh-hant`)、`/config/notify-lang`。
- `hwctrl`:`/interfaces`、`/interfaces/alarms`、`/interfaces/alarms/warnings`、`/interfaces/alarms/critical`。

查詢資料位於回應的 `body`;`/interfaces` 的 `body` 是陣列,其餘上述查詢為物件。一般網頁登入 `/api/auth/login` 的 Token 用於此入口則回傳 401,不能取代應用程式 Token。時間篩選與時間單位的細部測試先前使用 `/api/history/...` 路徑,本次統一入口歷史查詢驗證分頁取回 2 筆,補同步取回 1 筆。未測試設定寫入、開鎖或主動上報接收。

1.4 上線前需確認的版本差異

本文件的 `create-app-token`、`host` 與統一入口範例須配合目標韌體驗證。現有參考資料未提供下列行為的完整定義:

- `expiration` 的單位、`0` 的意義及 Token 到期後的處理方式。
- `refreshKey=true` 的作用與既有 Token 是否失效。
- 上報採 HTTP 或 HTTPS 的選擇方式、TLS 憑證驗證規則。不能僅依 `port` 推定協定。
- `host` 各欄位省略時的預設值,以及上報失敗的重送與保留策略。
- 統一入口遇到認證失敗或下游錯誤時,HTTP 狀態碼與回應格式的對應。

請以目標韌體的介面規格或實機測試確認上述項目,再作為正式整合契約。

2. 主動上報

上報路徑範例(實際以 `host` 設定為準):

用途	方法	路徑
事件	POST	`/api/history/events`
存活 Ping	POST	`/api/devices/ping`
介面值	POST	`/api/device/interfaces/values`

2.1 事件(`eventUrl`)

- **時機**:告警、狀態變化等(非固定週期)。

Request Body 範例

```
{
  "deviceId": "device1",
  "seq": 1001,
  "time": 1700050000,
  "level": 3,
  "msg": "[門禁1] 卡號 [1234567890] 開鎖",
  "kind": 1,
  "template": "elockCardUnlock",
  "args": {
    "pName": "門禁1",
    "iValue": "1234567890"
  }
}
```

欄位	必填	說明
deviceId	是	設備識別碼
seq	是	事件流水號(與 GET /logs/sync 之 **seq** 一致)
time	是	Unix 時間戳(秒)
level	是	3 資訊、4 正常、5 警告、6 危急
msg	是	人類可讀事件訊息
kind	否	週邊類型
template	否	事件模板鍵(與 **§ 5** 語系表一致時便於還原多語訊息)
args	否	模板變數物件(見 **§ 5**)
alarmSeries	否	告警趨勢;有告警趨勢資料時一併上送,格式與 **§ 4.1**/**§ 4.8** 回傳相同

主機建議回應

```
{ "success": true }
```

200	成功
400	格式錯誤
401	認證失敗
500	伺服器錯誤

2.2 存活 Ping(pingUrl)

- **時機**:每 pingInterval 秒(本文件範例為 900)。

```
{ "deviceId": "device1" }
```

回應同 2.1:{"success": true} 與上表狀態碼。

2.3 介面值(ifaceUrl)

- **定時全量**:每 ifaceInterval 秒,使用 values。
- **異動加送**:使用 updates(含 status / value)。

方式 1:values

```
{
  "deviceId": "device1",
  "time": 1753843649,
  "values": {
    "AI00E101": 1,
    "DI00E101": false
  }
}
```

方式 2:updates

```
{
  "deviceId": "device1",
  "time": 1753843649,
  "updates": {
    "AI00E101": { "status": "ok", "value": 1 }
  }
}
```

- 鍵為介面位址 addr,與 **GET /interfaces** 回傳之 addr 一致;與週邊、通道的對應請於設備上依 **§2.4** 畫面查閱。

2.4 工程模式/週邊資訊:位址對應週邊

介面值上報(values / updates)的**鍵名**必須為系統定義之介面位址。請於設備選單進入 **工程模式** > **週邊資訊**(實際選單名稱以韌體為準),畫面上會列出週邊與介面位址對照,例如 AI00101、DI00101、DO00101 或門禁/電子鎖相關位址。

將此處查到的 **addr** 作為 JSON 內的鍵,即可與 **GET /interfaces** 及主機端介面資料對齊。

2.5 使用 Mockoon 模擬主機(本機接收)

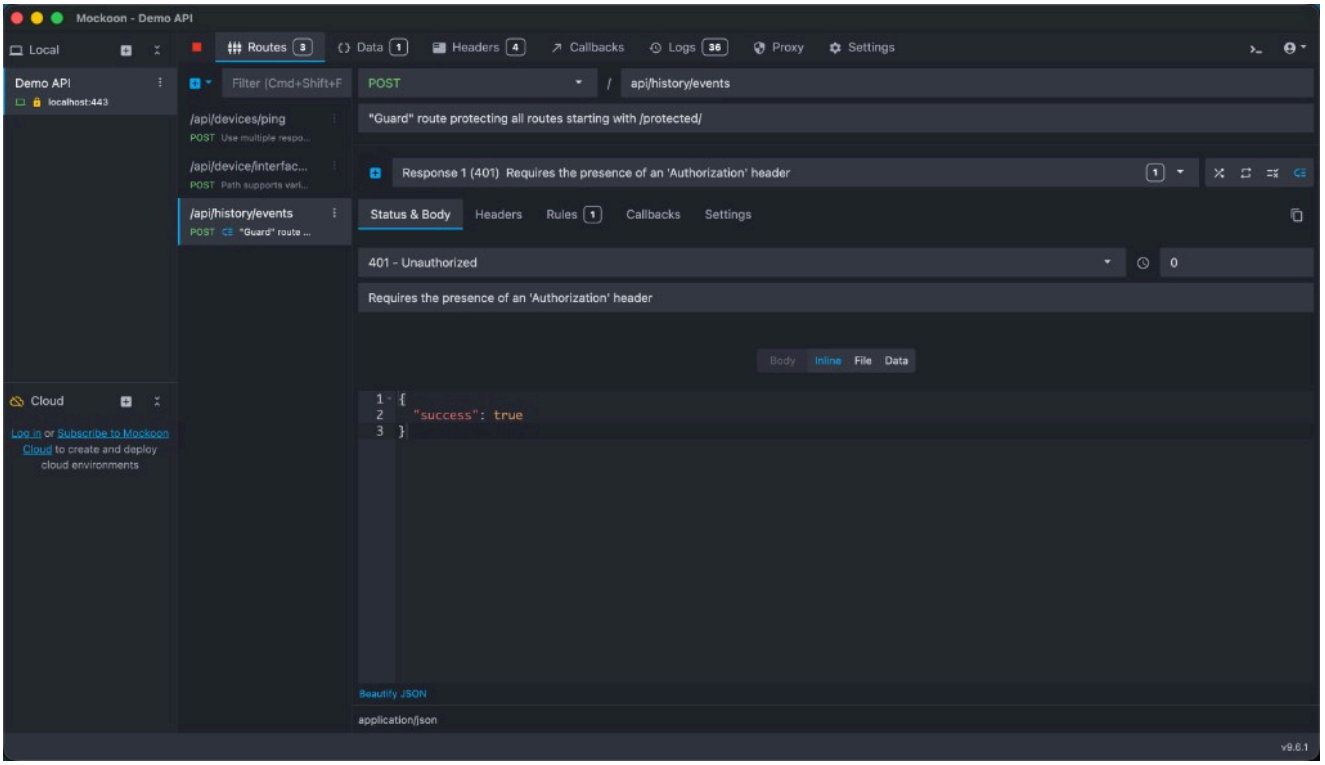
在尚無實體主機或需本機驗證時,可用 Mockoon 在本機模擬接收端,驗證**路徑、HTTP 方法、Request Body** 與 **Logs**。本節僅設定固定回應,不驗證 JWT 簽章或有效期限;可另外用 Rules 模擬 401,但這不等於完整的認證驗證。

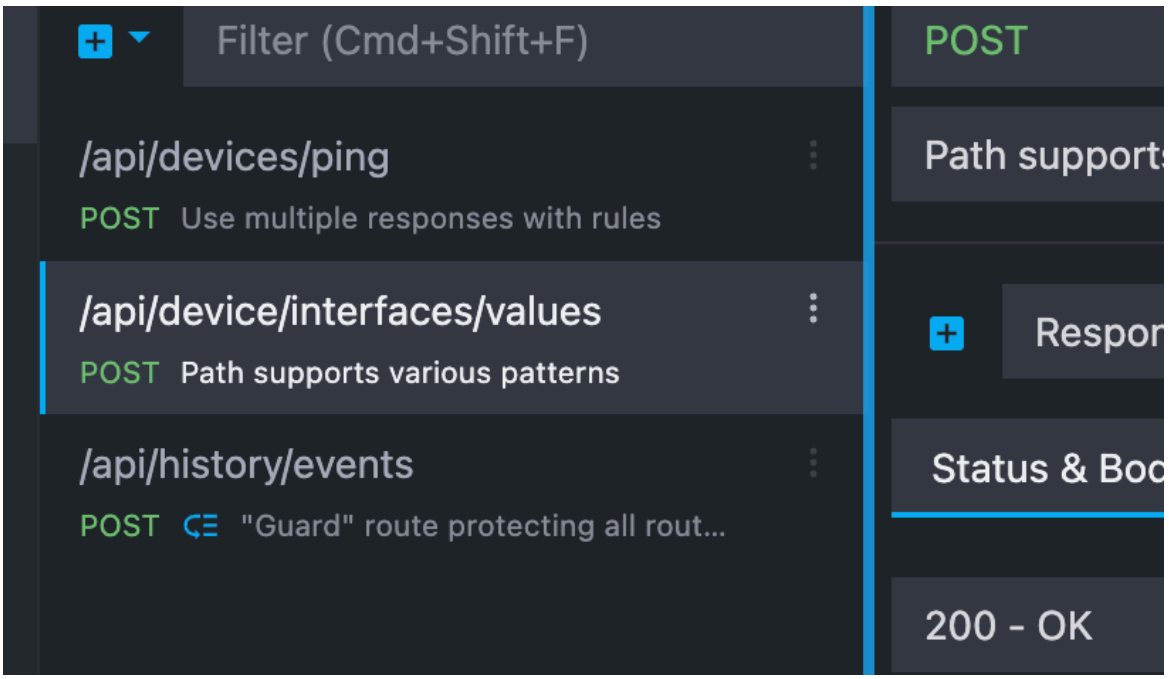
2.5.1 操作步驟摘要

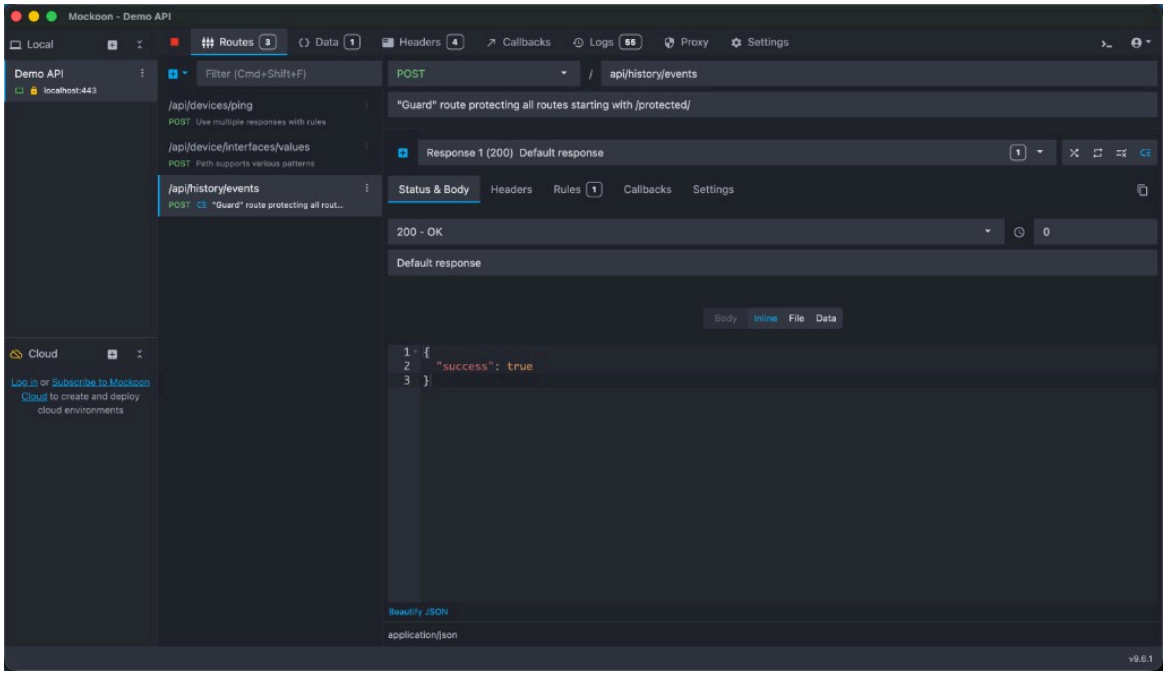
1. 安裝 Mockoon,新增 Environment(例:**Demo API**)。
2. 讓 Mockoon 監聽設備可連線的網路介面,例如 `0.0.0.0:3000`,並允許測試網路連入。設備的 `host.ip` 填該電腦的區域網路 IP(例如 `192.168.1.100`),`host.port` 填 `3000`;不可填 `0.0.0.0` 或 `localhost`。上報協定須與接收端一致;若韌體使用 HTTPS,需啟用 Mockoon 的 HTTPS 並確認設備接受其憑證,僅改成埠 443 不會啟用 TLS。
3. 新增三條 **POST** Route,路徑須與設備 `host` 內設定的 `eventUrl`、`pingUrl`、`ifaceUrl` **完全一致**(本文件範例如下表)。
4. 各 Route 的 Response 可先設 **HTTP 200**,Body 建議使用本文件 **§ 2.1~2.3** 所列主機建議格式:`{"success": true}`。
5. 將設備或測試程式的 `host.ip`、`host.port` 指向 Mockoon;送出請求後於 Mockoon **Logs** 檢查是否 **200** 與 Body 內容。

用途	POST 路徑(範例)
事件	`/api/history/events`
Ping	`/api/devices/ping`
介面值	`/api/device/interfaces/values`

注意: Environment **Settings** 若設定了 **Route prefix**,實際 URL = `prefix` + 路徑。請勿讓 prefix 與 `host` 內路徑**重複拼接**導致 **404**;請核對路徑是否與上表及設備設定完全一致。

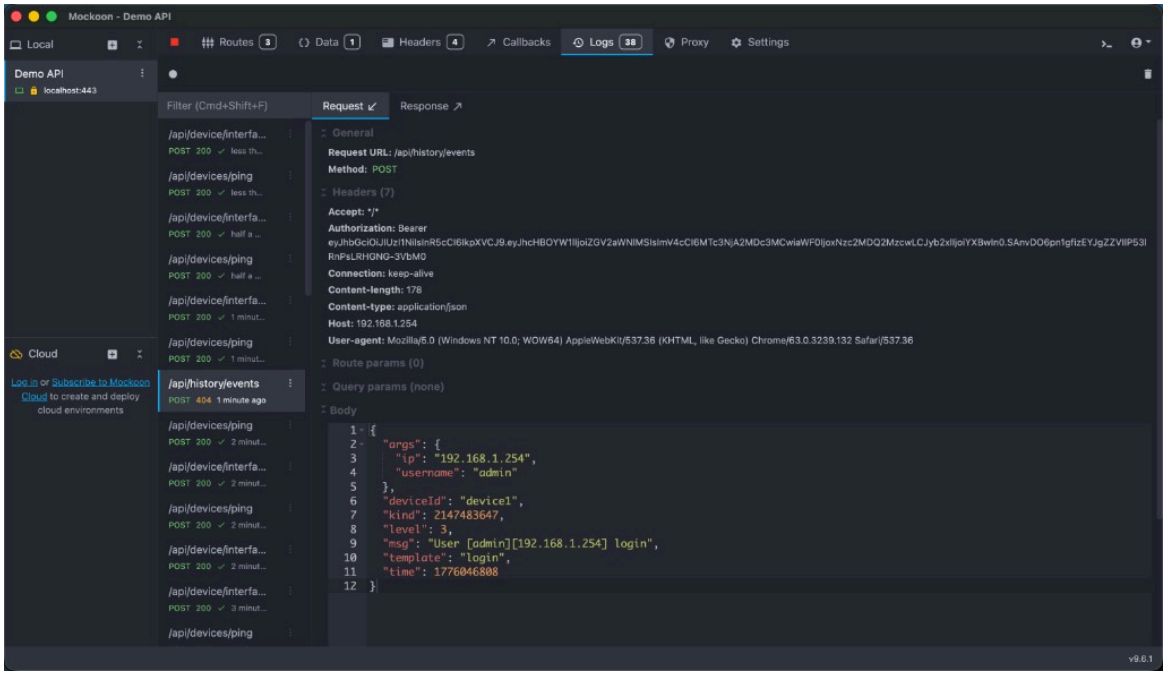




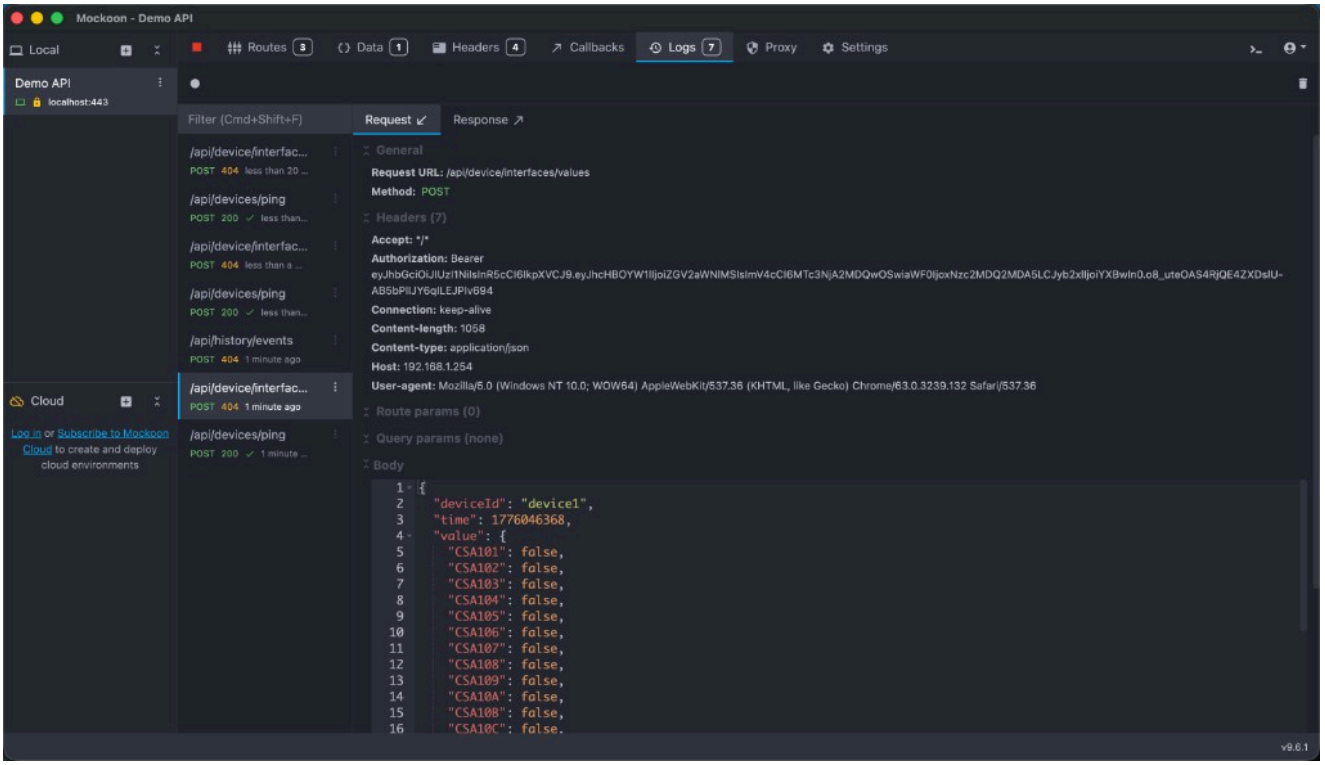


2.5.2 Logs 除錯

於 Mockoon 開啟 **Logs**,可檢視每次 POST 的 URL、標頭與 JSON。



若回應 **404**,請依序核對:是否為 **POST**、路徑是否與設備設定相同、`ifaceUrl` 是否含 **`/values`** 等尾段、prefix 是否重複。



截圖中的 IP、埠及路徑僅為示例;實際連線以本節步驟及設備的 `host` 設定為準。

3. GET /interfaces(hwctrl)

3.1 用途

取得設備上**所有介面**之即時列表:位址、狀態、目前值、閾值相關參數(如 AI)。整合時可用來對照主動上報 `values` / `updates` 的鍵名,或顯示相關介面狀態。

3.2 經統一入口呼叫

轉發 JSON(Body)

```
{
  "session": "trace-001",
  "serviceName": "hwctrl",
  "url": "/interfaces",
  "method": "GET"
}
```

回傳 body 結構(陣列;常見欄位如下)

實際欄位依機種/設定可能增減,常見如下:

```
[
  {
    "addr": "AI009C1",
    "status": "ok",
    "value": 220.5,
    "params": {
      "hc": 260,
      "hw": 250,
      "lc": 180,
      "lw": 190
    }
  }
]
```

欄位	說明
addr	介面位址(與 iface 上報等共用)
status	介面狀態字串(如 ok、hw、hc 等;詳見產品介面狀態說明)
value	目前值(可為數值、布林或字串,依介面型別)
params	介面參數;AI 的 hc/hw 為過高異常/警告值,lc/lw 為過低異常/警告值。範例為類比量測,不代表電子鎖的固定參數。

3.3 curl 範例(Bash)

```
curl -k -sS -X POST "https://${DEVICE_IP}/app-token/restful-api" \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${ACCESS_TOKEN}" \
  -d '{
    "session":"trace-001",
    "serviceName":"hwctrl",
    "url":"/interfaces",
    "method":"GET"
  }'
```

3.4 相關 API(選用)

路徑	說明
GET /interfaces/alarms	告警筆數彙總(warnings / critical / total)
GET /interfaces/alarms/{kind}	kind = warnings 或 critical 之明細列表

統一入口實測:上述介面列表、告警彙總、warnings 及 critical 明細,已使用應用程式 accessToken 經 restful-api 回傳 HTTP 200。介面列表的 body 為陣列;告警明細的 body 包含 count 與 items。

4. 事件記錄(history / logs)

4.1 GET /logs — 依時間查詢

依時間區間與篩選條件查詢**歷史事件/紀錄**(含先前上報並由設備保存之事件)。搭配 template、args 與 **§5** 語系表,可組出與畫面一致的中英文訊息。

4.2 完整 URL(query)

```
/logs?startTime={startTime}&endTime={endTime}&level={level}&offset={offset}&limit={limit}&name={name}
```

4.3 查詢參數

參數	說明
`startTime`	起始時間,Unix 秒。此版本使用完整參數名,不是 `start`。
`endTime`	結束時間,Unix 秒。此版本使用完整參數名,不是 end;邊界是否含當秒仍需確認。
`level`	紀錄等級篩選,例如 3;查詢全部等級時省略。`0` 不代表全部等級。
`offset`	分頁偏移
`limit`	每頁筆數
`name`	iName 過濾(選用)

4.4 經統一入口呼叫

轉發 JSON(Body):請將查詢時間替換成實際區間。建置版本 202609172144 的 Web 代理查詢已確認使用 `startTime`/`endTime`(Unix 秒);原先的 `start`/`end` 未生效,會得到未依指定時間篩選的資料。下例省略 `level` 與 `name`,表示不指定這兩項篩選。

```
{
  "session": "trace-002",
  "serviceName": "history",
  "url": "/logs?startTime=1700000000&endTime=1700086400&offset=0&limit=50",
  "method": "GET"
}
```

回傳 body 範例

```
{
  "total": 1,
  "offset": 0,
  "info": 1,
  "ok": 0,
  "warn": 0,
  "critical": 0,
  "columns": ["seq", "time", "level", "username", "template", "args",
        "alarmSeries"],
  "data": [
    {
      "seq": 1001,
      "time": 1700050000,
      "level": 3,
      "username": "system",
      "template": "elockCardUnlock",
      "args": {"pName": "門禁1", "iValue": "1234567890"}
    }
  ]
}
```

欄位	說明
`total`	符合條件之總筆數
`offset`	本次偏移
`info` / `ok` / `warn` / `critical`	各等級筆數統計(依服務實作)
`columns`	可用資料欄位清單;即使包含 `alarmSeries`,無趨勢資料的事件列仍可能省略該欄位。
`data`	紀錄列;含 **seq**、**username**、選用 **alarmSeries**(與 **§ 2.1** 主動上送相同)

欄位	說明
`data[].time`	事件時間,Unix 秒;此版本 `/logs/sync` 的事件時間則為毫秒,兩個 API 不可直接共用單位假設。
`data[].args`	此版本 `/logs` 回傳 JSON 物件;`/logs/sync` 回傳 JSON 字串,需要先解析。
`alarmSeries`	選用的告警趨勢陣列,元素含 `time`、`value`;本節卡號開鎖範例不含此欄位。時間單位須依目標韌體確認。

4.5 curl 範例(Bash)

```
START=1700000000
END=1700086400
curl -k -sS -X POST "https://${DEVICE_IP}/app-token/restful-api" \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer ${ACCESS_TOKEN}" \
  -d "{
    \"session\":\"trace-002\",
    \"serviceName\":\"history\",
    \"url\":\"/logs?startTime=${START}&endTime=${END}&offset=0&limit=20\",
    \"method\":\"GET\"
  }"
```

4.6 與主動事件的關係

- 主動 **POST 事件**(§2.1)與 **GET /logs**/**GET /logs/sync** 應保留一致的事件模板與參數內容;此版本 `/logs` 與 `/logs/sync` 的時間單位及 `args` 格式不同,主機需先正規化。若有 `alarmSeries`,也應保存其內容並確認時間單位。
- **GET /logs/sync** 另回傳依 **lang** 組好之 **msg**(§4.8)。
- 顯示語系可搭配 **GET/PATCH `/config/notify-lang`** 取得或設定通知語系。已實測一般登入後 `GET /api/history/config/notify-lang` 回傳 HTTP 200,內容為 `{"lang":"zh-TW"}`;`/api/sysctrl/config/notify-lang` 回傳 404。因此此設備使用 `history` 服務。此次未測試 PATCH,亦未變更語系。本節補同步請求使用 `lang=zh-hant` 已確認可回應 200。

4.7 GET /logs/max-seq — 設備最大 seq(主機補件)

用途:主機得知設備事件庫目前**最後一筆 seq**,以決定 `GET /logs/sync` 的 `toSeq` 或判斷是否尚有未同步紀錄。

`/logs/max-seq`

經統一入口

```
{
  "session": "trace-003",
  "serviceName": "history",
  "url": "/logs/max-seq",
  "method": "GET"
}
```

回傳 body 範例

```
{
  "maxSeq": 5230
}
```

欄位	說明
maxSeq	設備最新事件 seq;無事件時為 **0**

4.8 GET /logs/sync — 依 seq 區間補足缺失(主機)

用途:主要提供給主機補足缺失的事件記錄。主機保存已連續同步之 syncSeq,先 **§4.7** 取得 maxSeq,再依區間分批拉回(單次最多 **100** 筆)。中間 seq 不連續時,對缺口另行指定 fromSeq/toSeq。

`/logs/sync?fromSeq={fromSeq}&toSeq={toSeq}&lang={lang}`

參數	必填	說明
fromSeq	是	起始 seq(含)
toSeq	是	結束 seq(含);toSeq >= fromSeq
lang	是	zh-hant 或 en-us

經統一入口範例

```
{
  "session": "trace-004",
  "serviceName": "history",
  "url": "/logs/sync?fromSeq=1001&toSeq=1001&lang=zh-hant",
  "method": "GET"
}
```

回傳 body 範例

此版本的 data[].time 為 Unix 毫秒,data[].args 為 JSON 字串。以下 1700050000000 毫秒對應 §4.4 的 1700050000 秒,均為示例時間。

```
{
  "fromSeq": 1001,
  "toSeq": 1001,
  "count": 1,
  "maxCount": 100,
  "deviceId": "device1",
  "columns": ["seq", "time", "level", "username", "template", "args",
        "alarmSeries", "msg"],
  "data": [
    {
      "seq": 1001,
      "time": 1700050000000,
      "level": 3,
      "username": "system",
      "template": "elockCardUnlock",
      "args": "{\"pName\":\"門禁1\",\"iValue\":\"1234567890\"}",
      "msg": "[門禁1] 卡號 [1234567890] 開鎖"
    }
  ]
}
```

建議流程:syncSeq → GET /logs/max-seq → 若 syncSeq < maxSeq 則分批 GET /logs/sync(例:fromSeq=syncSeq+1,toSeq=min(syncSeq+100, maxSeq),lang=zh-hant)→ 寫入主機並更新 syncSeq。請以 count 與 data 判斷實際收到的筆數;上例完整示範 1 筆事件,maxCount=100 表示單次上限。

4.9 補同步與資料保留注意事項

- 主機依設備識別碼與 seq 去除重複事件,避免主動上報與補同步重複入庫。
- syncSeq 表示已連續同步完成的位置,不可直接以主機資料庫的最大 seq 取代;有缺口時需另行補查。
- 設備歷史事件有容量限制;已被覆寫的事件無法經 /logs/sync 取回。區間回傳空資料時,記錄缺口並查明保留狀態,避免無限重試,也不要宣告缺口已補齊。
- 若設備事件庫被清除或更換設備,且 maxSeq 小於原有游標,應重新確認設備識別與同步起點。

5. 事件模板與語系對應

以下為相關事件之模板鍵、說明、等級與訊息句式(含門禁/電子鎖等)。${xxx} 表示變數名稱;實際 JSON args 內為鍵值,請與下表變數對應。

5.1 英文語系(English)

template	說明	level	value(模板)
elockCardUnlock	電子鎖卡號開鎖	3	[${pName}] card [${iValue}] unlocked
elockCardSuccess	電子鎖卡號驗證開鎖	3	[${pName}] card [${iValue}] verified, user: [${iAValue}]
elockCardNotFound	電子鎖卡號不存在	6	[${pName}] card [${iValue}] not found
elockCardExpired	電子鎖卡號過期	6	[${pName}] card [${iValue}] expired
elockCardNotInPeriod	電子鎖卡號不在時段內	6	[${pName}] card [${iValue}] not in period
elockUnlockByHost	主機開鎖	3	[${pName}] unlocked by host, controlled by user [${username}] [${ip}] via web
elockUnlockByKey	鑰匙開鎖	3	[${pName}] key unlocked
elockSetOpenTime	設定開啟時間	3	[${pName}] set open time, source: [${source}][${name}]
elockAddCard	新增卡號	3	[${pName}] add card [${iValue}], source: [${source}][${name}]
elockDeleteCard	刪除卡號	3	[${pName}] delete card [${iValue}], source: [${source}][${name}]

5.2 中文語系(繁體)

template	說明	level	value(模板)
elockCardUnlock	電子鎖卡號開鎖	3	[${pName}] 卡號 [${iValue}] 開鎖
elockCardSuccess	電子鎖卡號驗證開鎖	3	[${pName}] 卡號 [${iValue}] 驗證成功, 使用者: [${iAValue}]
elockCardNotFound	電子鎖卡號不存在	6	[${pName}] 卡號 [${iValue}] 不存在
elockCardExpired	電子鎖卡號過期	6	[${pName}] 卡號 [${iValue}] 過期
elockCardNotInPeriod	電子鎖卡號不在時段內	6	[${pName}] 卡號 [${iValue}] 不在時段內
elockUnlockByHost	主機開鎖	3	[${pName}] 主機開鎖,使用者 [${username}][${ip}] 由網頁操作
elockUnlockByKey	鑰匙開鎖	3	[${pName}] 鑰匙開鎖
elockSetOpenTime	設定開啟時間	3	[${pName}] 設定開啟時間, 來源: [${source}][${name}]
elockAddCardId	新增卡號	3	[${pName}] 新增卡號: [${iValue}], 來源: [${source}][${name}]
elockDeleteCardId	刪除卡號	3	[${pName}] 刪除卡號: [${iValue}], 來源: [${source}][${name}]

說明： 英文表使用 `elockAddCard` / `elockDeleteCard`, 中文表使用 `elockAddCardId` / `elockDeleteCardId`;這個差異也存在於目前的事件參考文件,尚無可核對的韌體實作。不要自行改名或假設為同一鍵;請以實機送出的 `template` 確認映射,遇到未知鍵先保留原始事件。