04 開發者 / 整合

MQTT 整合

狀態:✅ **IMPLEMENTED(v6.0.83,2026-08-28 更新 WI-211 QoS／節流／週期重發／payload 格式)** | 原草擬:2026-06-02

01

⓿ WI-155(v5.9.272–283)— 對外命名空間 & 身分更新(整合商先讀)

對外 topic 前綴 `<P>` 從「**強制 UID 前綴**」升級為「**可設定命名空間 + MAC**」,並保持**完全向後相容**:

模式	topic 前綴 `<P>`	何時
新(命名空間)	`{projectname}-{env}/gateway/{mac12hex}`	設定頁填了「專案名稱」+「環境」(兩者皆全小寫;mac=12 hex 無冒號)
舊(legacy,預設)	`mes/gateway/{uid}`	未設專案名稱/環境時(既有設備不變)

範例:`topwell-dev/gateway/a8610a508b61`、`kspicnc-staging/gateway/aabbcc445566`。

整合商重點:

1. **訂閱用萬用字元涵蓋兩種**:`+/gateway/+/info`、`+/gateway/+/status` …(`+` 第一段同時吃 `mes` 與 `{proj}-{env}`)。
2. **MAC = 對外身分**:`whoami` / `info` 回應**已加** `mac` **欄位**(12 hex 小寫,= 新格式 topic 的 mac 段、= MQTT clientId `OPTA-<mac>` 的 mac)。內部路由/授權仍以 **UID** 為鍵(UID↔MAC 由雲端心跳自動建表,WI-147 不破)。

3. **MAC 撞號自動解決(AC4)**:同 MAC 多台時,雲端令「後到者」自動降級回 legacy `mes/gateway/{uid}`(永遠唯一、不誤路由);撞號排除後自癒。管理端可查 `GET /api/devices/mac-collisions`。
4. `cmd/signal/<N>` **payload**:同時接受 `{"value":N}` **(新)** 與**裸數字(舊)**,平滑遷移。
5. **MAC override(進階)**:設定頁可指定軟體 MAC(不碰硬體 OTP);設了會失聯的 MAC → 試用期 90s 內無 IP 自動退回硬體 MAC(防救不回)。⚠ 改 MAC 後設備可能取得**新 IP**(到路由器查或看雲端 ip);失聯排查見 `../guides/guide-troubleshooting.md` 〈MAC 覆寫後找不到設備〉。一般無需使用。

> 下文 §0 起的 `<P>` = `mes/gateway/<UID>` 為 **legacy 表示法**;啟用命名空間時整段前綴改為 `{proj}-{env}/gateway/{mac}`,**其後的子主題(`/cmd/...`、`/info`、`/status`、CONVERTER 輸出…)結構完全不變**。

> 原始狀態:草擬:2026-06-02
>
> ⚠ **實作與本草案的差異**:因設備 MQTT buffer 限 512B,**通道清單改「分頁」**—— `info` 只回摘要(身分/net/counts),每個通道另發 `{prefix}/info/io/<id>`(見 §3 實際格式)。草案 §3 的「單一 info 內含 channels[]/commands[]」未採用。`commands[]`(cmd/signal 角色標註)**v1 未做**,夥伴端可由 `info/io/<id>` 的 in 通道 topic 推得:規則引擎以 `cmd/signal/<N>` 數值訊號為來源,訊號可為 latched(持續閘控)或 pulse(上升緣觸發),再對應到 TCP IO 通道。對象:**現場管理端整合方**(第三方軟體團隊)+ 本控制器韌體開發
>
> 本合約定義第三方現場管理軟體如何**只透過 MQTT**(不需逐台 IP、不需走雲端):①探索現場有哪些設備　②取得設備管理資訊　③知道能對哪些 topic 做遠端控制。
>
> 已實作的底層語意:`cmd/signal/<N>` 為規則引擎的數值訊號來源,可作 latched(持續閘控)或 pulse(上升緣觸發),並對應到 TCP IO 邏輯通道。

02

❶-2 WI-211（v6.0.83）— QoS、發送節流與 payload 格式（整合商必讀）

一、QoS 預設值改了

方向	預設	為什麼
設備發出（狀態、回報）	**1**（至少一次）	送出去的絕大多數是「**最後狀態**」——重複送達會被下一筆覆蓋，無害。需要的是「一定會到」，不是「恰好一次」。
設備發出（`<P>/rules/*/trigger` 純通知）	**0**（最多一次）	它本來就是**設計上可丟**的通知（設備已刻意合併中間值）。QoS 1 每則都要等 `PUBACK`，函式庫逾時 1000ms——broker 在網際網路上只要塞車超過一秒就判定失敗並斷線重連。遺失風險由**週期重發**兜底（見下）。
設備接收（`<P>/cmd/#` 命令）	**2**（恰好一次）	命令是控制，**不可重複執行**（例如「開燈」被送兩次）。

v6.0.82 之前**收發都寫死 QoS 2**，且無法設定。實測每則發送要 265~400ms 且期間設備主迴圈完全凍結—— 密集狀態變化時會把連線打掉。現在兩者都可在設備網頁（設定 → MQTT）調整。

> **對整合商的影響**：若你的訂閱端假設 QoS 2 的「恰好一次」語意，請改為**冪等處理**（同一個狀態值收到兩次，結果必須相同）。狀態類 topic 本來就該這樣處理。

二、⚠️ 不要假設「一個事件 = 一則訊息」

v6.0.82 起，設備對**狀態類 topic** 做兩件事：

1. **同 topic 合併**：同一個 topic 還在發送佇列裡尚未送出時，新的值**直接取代**舊的，不新增一則。
2. **每 topic 最小發送間隔 1 秒**：同一個 topic 在 1 秒內的連續變化，只會送出**最後一個值**。

單次變化不受影響（該 topic 若 1 秒內沒送過就立刻送）。只有**連續快速變化**會被壓。

> **這是刻意的。**設備是單執行緒工控閘道，每則 MQTT 發送都會阻塞現場控制。密集發送曾造成連線反覆斷開、甚至設備自我重開（WI-210）。**控制的即時性優先於通知的完整性** —— 中間被覆蓋的值本來就已經不是現況了。

若你需要完整的變化歷程，看下一節。

二-B、週期重發：過時上限 60 秒

設備每 **60 秒**會把每個狀態類 topic 的**現況原樣重送一次**（該 topic 若期間本來就有更新，則不重送）。

> **這是給你的保證**：即使某一則訊息在傳輸中遺失，**你的狀態最多過時 60 秒**，不會永遠停在錯的值。這也是 `rules/*/trigger` 敢用 QoS 0 的前提。
>
> 為什麼不是「同一則發兩次」：兩份相隔幾毫秒的複本會**一起死在同一條斷掉的連線上** —— 而連線斷掉正是 QoS 0 唯一會發生的遺失情境。重複發送擋不到它，週期重發可以。

對整合商的影響：你會週期性收到「內容沒變」的訊息。這是正常的，請以冪等方式處理（收到相同狀態值不應觸發重複動作）。已被週期推送的資料（如 RS485 量測值）不受影響 —— 它們本來就在更新，不會額外重發。

三、payload 夾帶歷史（選用，預設關閉）

設備網頁勾選「payload 夾帶歷史紀錄」後，**狀態類 topic 的 payload 會從裸值改成 JSON**：

```jsonc
// 關閉時（預設）── 格式與 v6.0.82 之前完全相同
12.345

// 開啟後
{"v":12.345,"t":1787839126,"h":[[354,12.1],[712,12.28]]}
```

欄位	意義
`v`	**最後狀態**（即時值）。只要即時狀態的介面**只讀這個欄位就夠了**。
`t`	本批的起點時間，**Unix epoch 秒**。`t=0` 代表設備時鐘尚未同步 → 只有相對順序可信。
`h`	上一次發送到這一次之間，被合併掉的每一筆：`[相對 t 的毫秒數, 當時的狀態]`。
`trunc`	只在歷史超過 8 筆被截斷時出現（`true`）。**代表這批不完整**，不要當成只發生了 8 次。

實際樣本（本站實測）：

```JSON
{"v":"A2-MQTT滅","t":1787839126,"h":[[354,"A2-MQTT滅"]]}
```

`v` 的型別跟隨原本的值：數字不加引號、字串加引號。

> ⚠️ **開啟這個選項會破壞既有的解析程式。**這是知情的選擇，不是升級後被動改變 —— 韌體升級到 v6.0.82 **不會**自動開啟它。要開之前，先確認所有訂閱端都改好了。
>
> 為什麼上限是 8 筆：單則 payload 的硬上限是 128 bytes。超過就標 `trunc`。

03

0. 完整系統 MQTT Topic 清單（權威）

`<P>` = topic 前綴：legacy `mes/gateway/<UID>`（24 字 UID）**或** WI-155 命名空間 `{projectname}-{env}/gateway/{mac12hex}`（見 § ❶）。以下子主題結構兩種模式通用。為**韌體實際 sub/pub 的系統 topic**（v5.9.283 核對 `src/MesMQTT.h` / `main.cpp`）。

探索 / 身分 / 狀態

Topic	方向	Retained	說明
`mes/gateway/whoami`	管理端 → 全設備（**免前綴廣播**）	–	探索全廠；各設備回 `<P>/info`
`<P>/cmd/info`	管理端 → 設備	–	指定查詢單台 → 回 `<P>/info`
`<P>/info`	設備 → 管理端	–	身分摘要（uid/name/fw/ip/online/uptime/counts）
`<P>/info/io/<id>`	設備 → 管理端	–	每通道控制/資料介面（分頁，避 512B 上限）
`<P>/status`	設備 → broker	✅	在線狀態 birth/LWT（`online`）

命令（管理端 → 設備，`<P>/cmd/#`）

Topic	Retained	說明
`<P>/cmd/signal/<N>`	❌ **切勿 retain**	MQTT 訊號值（規則引擎來源；觸發/閘控）。payload: `{"value":N}`（WI-155 新）**或**裸數字（舊）皆可
`<P>/cmd/do/<N>`	–	數位輸出 DO<N> 控制
`<P>/cmd/scale/active`	–	磅秤發重量門控（`1` 開 / `0` 停）
`<P>/cmd/scale/calibrate`	–	磅秤校正命令
`<P>/cmd/telemetry`	–	啟動聚合遙測 + keepalive（payload=間隔秒數，`0` 停；超時自停）
`<P>/cmd/_ping`	–	**內部** self-ping：設備每 45s 發給自己訂的此 topic，確認 MQTT subscribe 仍活著

回報 / 輸出（設備 → 外）

Topic	Retained	說明
`<P>/telemetry`	–	聚合遙測（`cmd/telemetry` keepalive 門控才發）
`<P>/cmd/scale/calibrate/ack`	–	磅秤校正命令 ack
`<P>/<自訂>`	–	**CONVERTER 通道輸出**，topic 由設定決定。慣例：`<P>/scale/weight`（磅秤）、`<P>/power`（用電量）、`<P>/count`（SF965 計米）；見 `../guides/guide-triggered-periodic-report.md`

授權（雲端 → 設備）

Topic	Retained	說明
`<P>/license/state`	✅	簽章 token 下發（base64；WI-145；唯一 publisher = 雲端；retained 重連即重投最新 token）

雲端中繼（cloud-bridge）

- 雲端 `cloud-bridge` 訂 **`+/gateway/#`**（WI-155:同時涵蓋 legacy `mes/gateway/#` 與命名空間 `{proj}-{env}/gateway/#`）做 telemetry fan-out（→ `wss://opta.smms.com.tw/ws/telemetry`）；遙測拆解子主題如 `<P>/rs485/<idx>/<field>`、`<P>/power/<idx>/<field>`。
- WI-155:命名空間 topic（`{proj}-{env}/gateway/{mac}/...`）由 cloud-bridge 經 **MAC↔UID 反查**改寫成 canonical `mes/gateway/{uid}/...` 後再處理 → 下游（PG/Things、WS fan-out）與儲存皆以 UID 為鍵，整合不受命名空間影響。

> **小結**：固定系統 topic 約 **14 條**（探索 5 + 命令 6 + 回報 2 + 授權 1），外加數量不定的 **CONVERTER 自訂輸出** topic。⚠️ `cmd/signal/*` 與所有 `cmd/*`：**MQTT 無 per-message 授權，控制權 = broker 連線權**（見 §6 安全）。`cmd/*` 一律別 retain（retained 重連會反覆灌回誤觸發）。

04

1. 為什麼需要這份合約

- 雲端 config server 有自己的設備管理；但**現場管理端是另一個團隊的獨立軟體**，不走雲端、不便逐台用 HTTP（要先知道每台 IP）。
- 控制器與現場管理端**共用現場 MQTT broker**。最自然的整合面就是 MQTT：pub/sub、免逐台 IP、可廣播。
- **前綴 `mes/gateway/<UID>` 是強制的、綁設備 UID**（韌體以設備 24 字 UID 組 topicPrefix，確保每台 topic 命名空間唯一、不撞台）。→ 管理端「要先知道 UID 才能跟設備對話」，因此需要**不需先知 UID 的探索機制**。

05

2. 介面總覽（三個能力）

能力	方向	Topic	機制
探索 (whoami)	管理端→全設備	`mes/gateway/whoami`（**免前綴廣播**）	廣播請求
資訊回報 (info)	設備→管理端	`<P>/info`（`<P>` = `mes/gateway/<UID>`）	回應 / 也可指定查 `<P>/cmd/info`
在線狀態 (status)	設備→管理端	`<P>/status`（retained）	既有，birth/LWT

> 探索流程：管理端訂 `mes/gateway/+/info`（萬用）→ 發一筆 `mes/gateway/whoami`（payload 空或 `1`）→ **每台設備各自回 `<P>/info`**（含自己的 UID）→ 管理端即取得全廠清單與每台的控制合約。指定查詢：已知 UID 時發 `<P>/cmd/info`（payload 空）→ 該台回 `<P>/info`。

06

3-IMPL. 實際回應格式（v5.9.197，分頁）★以此為準

收到 whoami / cmd/info 後，設備發**多筆**訊息（QoS0、非 retained）：

① **摘要 →** `<P>/info`（單筆，小）：

```JSON
{"schema":1,"uid":"00310044303351103432393 2","mac":"a8610a508b61","name":"OPTA-K(
 "ip":"192.168.72.77","online":true,"uptimeSec":46,
 "counts":{"signals":7,"rules":7,"actions":6,"tcpio":5}}
```

> WI-155(v5.9.279+):新增 `mac` 欄(12 hex 小寫)= 對外身分。整合端用它對應 MAC-based topic;內部仍以 `uid` 為鍵。② **每個啟用通道 →** `<P>/info/io/<channelId>`（控制/資料介面，分頁）：

```JSON
{"id":5,"name":"啟動磅秤","dir":0,"mode":0,"proto":0,"topic":"mes/gateway/<UID>/cm
{"id":22,"name":"發出重量","dir":1,"mode":2,"proto":0,"topic":"mes/gateway/<UID>/s
```

- `dir`：0=IN（管理端可發此 topic 控制）／1=OUT（設備發、管理端訂閱取資料）
- `mode`：0=Legacy／1=Parser／2=Converter　`proto`：0=MQTT／1=Modbus TCP
- 管理端流程：訂 `mes/gateway/+/info` + `mes/gateway/+/info/io/+` → 發 `mes/gateway/whoami` → 收齊全廠摘要＋各通道。

> 以下 §3 為**原始草案**（單一大 JSON），保留作設計脈絡；實際以上方 §3-IMPL 為準。

07

3. `info` 回應 payload（JSON）★合約核心〔草案，未採用〕

> 重點：`channels` **與** `commands` **就是遠端控制合約** —— 管理端據此知道「能對哪些 topic 發什麼、設備從哪些 topic 出資料」。

```jsonc
{
  "schema": 1,                          // 合約版本，破壞性變更才 +1
  "uid": "003100443033511034323932",
  "name": "OPTA-KC-A1",                 // 設備名稱（設定頁可改）
  "model": "MES-Gateway-Opta",
  "fw": "5.9.196",
  "net": { "ip": "192.168.72.77", "mac": "AA:BB:..", "online": true, "uptimeSec"
  "rs485": [                            // 掛載的 Modbus 設備
    { "name": "TDA08B-1", "slave": 3, "baud": 9600, "online": true }
  ],
  "counts": { "signals": 7, "rules": 7, "actions": 6, "tcpio": 5 },

  // ★ 遠端控制 / 資料介面（TCP IO 通道）— 管理端最需要的
  "channels": [
    { "name": "發出重量", "dir": "out", "proto": "mqtt",
      "topic": "mes/gateway/<UID>/scale/weight", "desc": "每2秒，門控開時。payload
    { "name": "啟動磅秤", "dir": "in",  "proto": "mqtt",
      "topic": "mes/gateway/<UID>/cmd/signal/7", "desc": "發重量開關" }
  ],

  // ★ 內建命令暫存器對照（cmd/signal/N，遠端控制用）
  "commands": [
    { "topic": "mes/gateway/<UID>/cmd/signal/5", "role": "啟動",       "kind": "p
    { "topic": "mes/gateway/<UID>/cmd/signal/6", "role": "校正參數",   "kind": "la
    { "topic": "mes/gateway/<UID>/cmd/signal/7", "role": "發重量開關", "kind": "la
  ]
}
```

欄位語意

- `kind`：`pulse`＝要送 `1` 再 `0`（邊緣觸發）；`latched`＝送一次保留（狀態）；`level`＝持續電平門控。管理端據此知道**怎麼驅動**每個控制 topic（pulse 須送 `1` 再送 `0` 才不會卡在觸發態，latched/level 則送一次即保留，故只要遵守 kind 語意就不會前後競態）。

- `dir`：`in`＝管理端可發佈來控制；`out`＝設備發佈、管理端訂閱取資料。
- `topic`：合約回**完整 topic**（已含前綴），管理端直接用，不必自己拼。

> **大小考量**：`channels` 多時 payload 會變大。若超過單筆 MQTT/RAM 上限，v1 可：① `info` 只回身分+net+counts，②另設 `<P>/cmd/io` → `<P>/io` 專回完整 channels 清單（分頁）。實作時定。

08

4. 安全性（實作前必須拍板）

- MQTT 沒有 per-message 授權 —— **任何能連上 broker 的人都能發 whoami / cmd/signal**。
  - `info` 是**唯讀管理資訊** → 一般可接受（不含密碼/token）。
  - 但 `cmd/signal/*`（遠端控制）同樣無授權 → **控制權＝broker 連線權**。現場 broker 的帳密/網段隔離就是安全邊界。
  - ⚠️ 若管理資訊含敏感欄位（MAC、內網拓樸）或要限制控制，需在 broker ACL（依 topic/帳號授權）層處理，韌體不做 per-client 授權。
- 廣播 whoami 要評估**多設備同時回報的訊量**（N 台 → N 筆 info）；可加隨機抖動避免同時湧出。

09

5. 與現有介面的關係

現有	提供什麼	為何仍需本合約
雲端 config server	雲端設備清單/部署	現場第三方不走雲端
HTTP `/api/system`、`/api/poll`、`/api/config`、`/api/tcpio`	同等資訊（含 deviceUid、IP、通道）	要先知道每台 IP；MQTT 免逐台 IP
`<P>/status`（retained `online`）	在線與否	只有 online 字串、無設備資訊、無 UID 以外內容

> 本合約等於把 HTTP 的 `/api/system` + `/api/tcpio` 精華，用 **MQTT 廣播探索 + 回應** 包成第三方好整合的形狀。

10

6. 實作備註（韌體，WI-131 待排）

- 新增 handler：訂 `mes/gateway/whoami`（免前綴，需在既有 `cmd/#` 之外多訂一條）+ `<P>/cmd/info`。
- 組 `info` JSON：重用既有 `/api/system`、`/api/tcpio` 的序列化邏輯（ArduinoJson）。
- **Flash 預算**：目前 91.8%（餘 ~64KB）。新 handler + JSON 組裝要量 size；channels 清單可能要分頁/精簡。
- birth 是否一併豐富化（`status` 改 JSON，於上線時即帶身分摘要）→ 可選。
- 合約版本 `schema` 欄位：破壞性變更才 +1，讓管理端能相容多版設備。

11

7. 待夥伴團隊確認

- [ ] `info` 要哪些欄位（本草案：身分/net/rs485/counts/channels/commands）是否齊全？
- [ ] channels 是否需要更細（每通道的 payload schema、QoS、retain 慣例）？
- [ ] 探索：廣播 whoami 足夠，還是也要定期主動 publish（管理端被動清點）？
- [ ] 安全邊界：broker ACL 由誰管？是否需要限制 `cmd/*` 控制權？

12

8. 相關文件

- 通用引擎 / API：`spec-mqtt-chain-workflow.md`
- HTTP API：`../api/api-firmware.md`