01 使用者

# 磅秤校正操作指南

> 2026-10-06：6.0.134 已完成 `.46` 實秤三段 5→25→45 kg 與 finish，降載 25／5／0 kg 時面板及系統分別為 25.01／5.01／0.01 kg。雲端發布至 beta，變體 `calibration-candidate`，Build ID `35B4EEB5DA9C49EC9B363566A7EFE4B7`。操作工具預設穩定等待 5 分鐘；精度允收、校正後斷電保存、五段實秤、業主 Edge 與長時仍待驗，1.0 未放行。

本文件操作對應 135 測試候選（v2）；134 beta的實秤結果是前版證據，沿用 126 起的原重量 payload 整合；124 的舊獨立 topic 測試不能代替本輪驗證。123 不支援這個新命令。一般 `opta` 建置不含本功能，工程建置使用 `opta-calibration`。雲端可下載／指定部署，不代表整機已完成交機驗收。

**開發者入口：** 請讀Edge 開發者指南，內含完整會話流程、ping保活格式與回覆、每種命令、完整payload欄位、ACK／重送與逾時處理。這些由Edge自動實作，操作員不應手動維持心跳；135的穩定憑證可供MQTTX逐步手動測試。

---

01

## 實機操作前確認

先依 現場驗收計畫 核對候選版本、備份與本輪允許操作範圍。確認 TDA08B 通道 1、單位 kg、正確站號，並由業主確認安全秤量至少涵蓋本次累加總量、允許誤差與原曲線恢復方式；未確認不寫入校正。停用既有訊號觸發校正規則，避免與新會話混用。

### 開始前必讀：取消不會恢復校正前讀值

**目前沒有校正曲線Rollback。** 只有begin時還沒寫校正；zero成功後零點已改，每個point成功後也已提交該點，並不是等finish才開始生效。中途按取消、斷線或逾時，都不會自動還原這些變更；結果unknown時也不能假設沒寫入。

中止後標記「需要重新校正」，重新清空秤盤、完整做完所有段數，再依允許誤差驗重量。OPTA設定備份、回復舊韌體，以及checkbox恢復原本勾選，都不能還原控制器曲線。開發者的取消確認與處理方式見中止與Rollback限制。

---

02

# 先用設備網頁完成設定與正常發送

**本節 A～E 在 OPTA 自己的網頁操作，不是在文件中心。**新版多段命令由業主 Edge 頁面或下方操作工具發出；OPTA 網頁目前沒有「開始三段／下一段／完成」表單。網頁顯示重量或「發送中」也不能代替校正 ACK。

## A. 登入與核對本輪設備

1. 瀏覽器開啟現場 OPTA 的 IP，以具設定權限的帳號登入；不要把實驗室 `.46` 當成所有現場設備的固定 IP。
2. 依 驗收計畫的前置核對，由工程人員確認完整版本、Build ID、UID 與備份。畫面未顯示的資訊用安裝紀錄核對，不自行認定相同版號就是同一個包。
3. 點左側「設備」，找到「RS485 設備」。已有該磅秤時編輯原設備，不要再新增相同 Slave ID。

**通過畫面：**開啟的是本輪設備，既有磅秤項目與現場站號一致。未完成備份先停在這裡。

## B. 設定 RS485 通訊與磅秤

1. 在「設備」→「RS485 設備」按「通訊埠設定」。核對 Baud Rate、Parity 與磅秤面板；這是匯流排共用設定，其他接在同一線路的设备也必須一致。需要修改時才按「儲存所有通訊埠設定」。
2. 按「＋」新增，或打開原設備的編輯視窗。「設備類型」選 **TDA08B 磅秤**。掃描顯示「未知設備」只表示未辨認型號；須先確認真的是 TDA08B，再指定樣板。
3. 檢查「設備名稱」：選型號會帶入預設名稱，已手動命名者保留。`Slave ID` 填控制器通訊位址；「實體連接埠 (Bus)」選實際接線埠，勿把 IP 或 UID 填進 Slave ID。
4. 「輪詢間隔 (ms)」先保留畫面預設值，確認設備為啟用。使用 TDA08B 預設暫存器；已有自訂內容時先核對，別為了驗收覆蓋它。
5. 按「儲存」，依下一節處理設定同步。

*新增磅秤畫面示意：使用模擬 API 拍攝，站號 3 為示例；不是本輪實秤驗收證據*

## C. 儲存時核對控制器設定

| 按儲存後看到什麼 | 你要做什麼 | 這一步算完成嗎 |
|---|---|---|
| 設定一致，直接儲存成功 | 重開編輯視窗確認內容仍正確 | 設定儲存完成，仍需核對實際重量 |
| 「磅秤設定與目前表單不同」 | 閱讀差異、關閉提示、檢查更新後的倍率，再按一次「儲存」 | 第一次只是更新表單；第二次成功才算已存 |
| 「尚未儲存設備」或讀取失敗 | 檢查供電、站號、配線及共用通訊參數，恢復後重試 | 未完成；不要猜倍率繞過同步 |

每次編輯儲存都會讀取控制器設定，並非只在新增時執行。這是顯示倍率／分度設定同步，**不會改寫控制器校正曲線**。

回「總覽」→「RS485・匯流排設備」，選磅秤，與面板對照重量、正負號、小數位及單位。放上／移除已知重量，確認畫面跟著變。若面板 `-0.03`、畫面 `-0.3`，或數值不動／離線，先停止並處理通訊與倍率；不要靠多段校正掩蓋設定錯誤。

## D. 建立重量發送流程

**已經有正常重量流程時，保留原流程，直接進 E 核對。** 不要為了補時間戳重新套用整套範本覆蓋現場設定。

1. 新設備先儲存，再重新開啟其編輯視窗。
2. 找到「⚖️ 套用『磅秤發重量』流程範本」，按下後閱讀確認視窗，再按「套用流程」。未儲存的新設備不會顯示此按鈕。
3. 應看到「磅秤發重量流程已建立」，記下視窗中的實際重量 topic。它會建立「直接發」及「校正後發」兩條流程；後者是**先歸零再發**，不是新版多段校正。
4. 這輪使用「磅秤直接發閘／磅秤發重量-直接」。由工程人員確認舊訊號校正規則已停用；不要操作「磅秤校正流程」的執行按鈕。

**通過畫面**：流程存在、指向正確磅秤與 topic。提示成功只代表建立完成，還要確認實際發送。

## E. 在「工作流程」開啟直接發送

1. 到「設定」確認 MQTT 已連線，再點左側「工作流程」。
2. 找到來源為「磅秤直接發閘」、輸出為「磅秤發重量-直接」的卡片，確認對應這台磅秤。改過名稱時依來源、輸出及 topic 辨認。
3. 將這張卡的「發送開關」切為 ON。應顯示「🟢 發送中」。若顯示送出失敗，先檢查 MQTT 連線，不要連按其他流程。
4. 在業主 Edge 的接收畫面，核對原重量 topic 確實收到新資料、重量與面板一致。沒有 Edge 接收畫面時，由工程人員用 MQTT 接收工具保存 payload；OPTA 開關燈號不能代替接收證據。
5. 校正期間保持直接發送開啟，**不用手動取消「數值變動才送／變更時才發」checkbox**。系統保留原設定，校正中暫時讓相同重量的新樣本也發送；`finish` 成功後恢復原規則：原本勾選就只在資料變動時送，原本未勾選就按原間隔送，原門控也恢復。v2同一reading的重複包不得算兩筆資料；sample相同、reading增加是正常持續量測。
6. 收尾後先確認匹配本輪 `boot/sid/seq` 的 `applied` 與 `state=complete`，再逐步取下砝碼，核對面板／總覽／MQTT。checkbox 前後應一致；勾選時，重量及模板內其他數值保持不變應停止重複發送，時間戳更新本身不算數值變動。若 `unknown`，先停止並查原因，明確取消後才回一般流程；不要把錯誤狀態當成校正完成。

> 工具在 finish 後剛好還送出一筆 ping 時，可能看到 `state=complete` 搭配 `result=rejected`、`reason=lease_not_active`、`requestSeq=0`。這是「已結束的會話不能再續期」，不是 finish 失敗。finish 的完成證據仍以對應 seq 的 `applied` 為準；Edge 應在結束後停止 ping。

開關用於開始／停止，並不是校正按鈕，也不是自由輸入發送間隔的表單。需要特定秒數時由工程人員設定並核對實際收到的間隔；重啟後應再次確認是否已開始發送。

## F. 網頁設定完成後，交給 Edge 控制校正

保持設備「總覽」方便對照面板。業主 Edge 頁面需提供開始、空秤確認、每段新增公斤數、完成／取消與狀態；實際按鈕名稱由業主介面決定，OPTA 網頁沒有這些按鈕。

- Edge 只訂閱原重量 topic，從同包 `calibration` 判斷狀態及匹配命令 ACK；單純重量到達不能進下一段。

- 砝碼依序新增 **5、20、20 kg**，保留前面砝碼；送往控制器的總量是 **5、25、45 kg**。操作人員不用去小數點。
- 若現場沒有可用的 Edge 頁面，由工程人員使用下一節工具帶流程；操作員仍用面板與 OPTA 總覽核對重量。不要把這種測試簽成「業主 Edge 網頁已驗收」。
- 時間戳的網頁查看步驟請依 量測時間指南：網頁核對，結果回填既有 TS 項目。

---

03

# 用 MQTT 訊息操作三段校正：照表執行

本節供現場操作人員使用 MQTT 客戶端核對訊息，也供 Edge 頁面開發者實作流程。已有 Edge ／下節工具控制校正時，不要再由另一個 MQTT 客戶端同時送校正命令。

範例使用站號 **3**、安全秤量 **200 kg**、三顆砝碼 **5、20、20 kg**。安全秤量必須經業主確認，不能因為範例寫 200 就照填。秤盤先清空，正常重量上報須已啟動，並確認面板與上報重量一致。

## 1. 設定 topic，先查詢目前狀態

| 用途 | Topic／設定 |
| --- | --- |
| 發送命令 | `mes/gateway/<設備UID>/cmd/calibration` |
| 接收重量與回覆 | 使用原重量 topic，標準為 `mes/gateway/<設備UID>/scale/weight`；不必另訂閱 ACK topic |
| 發送設定 | **QoS 1，Retain 關閉**；不要把校正命令設為保留訊息 |

`<設備UID>` 必須換成這台 OPTA 的 UID，不是 IP 或磅秤站號。先確認訂閱成功，再發送：

```json
{
  "op": "query"
}
```

query 只查詢，不標零、不改校正，也不延長會話。從回覆的 `calibration` 讀取 `boot`、`sid`、`seq`、`state`。不要使用 retained 舊訊息作本輪依據。

## 2. BOOT／SID／SEQ 怎麼填

| 欄位(JSON 用小寫) | 白話意思 | 正確填法 |
| --- | --- | --- |
| `boot` | 哪一次 OPTA 開機 | 複製最新 query 回覆的 `calibration.boot`，包含大小寫、完整字串；不是 bootCount、韌體版本，也不是自己編的值 |
| `sid` | 哪一輪校正 | 確認沒有需處理的既有會話後，新一輪用設備目前 `sid + 1`；這輪從開始到收尾都不變 |
| `seq` | 這輪的第幾個動作 | begin 從 **1** 開始；收到匹配的 applied 後，下一個動作才加 1。ping／query 不增加 seq |

**Edge 怎麼取得 boot？不需要登入 OPTA，也不需要人工抄寫。**

| Edge 的步驟 | 發送／接收位置 | 能取得什麼 |
| --- | --- | --- |
| 探索設備 | 先訂閱 `mes/gateway/+/info`，向 `mes/gateway/whoami` 發送 `1`（Retain 關閉） | 各台回 `<P>/info`：UID、名稱、版本、IP 等；**6.0.128 這份回覆沒有校正 boot** |
| 找到重量 topic | 檢查既有設定，或探索時一起訂閱 `mes/gateway/+/info/io/+` 取得啟用通道資訊 | 選出這台磅秤實際使用的重量 topic，不能只假設一定是標準名稱 |
| 取得最新校正狀態 | 先訂閱重量 topic 並確認成功，再向 `<P>/cmd/calibration` 發 `{"op":"query"}` | 從非 retained 回覆的 `calibration` 取得 boot／sid／seq／state |
| 自動帶入命令 | Edge 暫存這台設備的校正狀態 | begin／zero／point／finish／ping 使用同一個 boot；發現 boot 改變即停止舊流程、重新 query |

`<P>` 是該台的完整 topic 前綴，例如 `mes/gateway/<UID>`。已持續接收重量時，payload 的 `calibration.boot` 本來就可讀取；開始前仍 query 一次核對最新狀態。這個 16 位十六進位字串不是 Build ID，也不是 bootCount，三者不可互換。不要把文件或昨天的 boot 寫死在 Edge 裡。

例如 query 回覆 `sid:0`、`seq:0`、`state:"recovery_required"`，新一輪才用 `sid:1`、`seq:1`。如果回覆 sid 已是 7，新一輪應用 8，不能永遠填 1。查到 `await_zero`、`collecting` 或 `unknown` 時，先查明／明確取消既有會話，不另開新 sid 強行接管。取消不回滾控制器已生效的點。

**OPTA 重開後 boot 會改變。**停止舊流程，重新 query 並完整重做；不能只替換 boot 後續做舊段數。頁面重開也應先 query，不自動接管。

## 3. 開始三段校正，啟動心跳

以下是**填寫範本**：將 `填入最新query回覆的boot` 換成剛收到的字串，sid 依上表計算；這個佔位字不能直接發送。

```JSON
{
  "op": "begin",
  "boot": "填入最新query回覆的boot",
  "sid": 1,
  "seq": 1,
  "slave": 3,
  "steps": 3,
  "unit": "kg",
  "model": "tda08b",
  "capacityKg": "200",
  "toleranceKg": "0.1"
}
```

`capacityKg`／後續 `totalKg` 必須是**帶引號的十進位字串**；`sid`／`seq`／`slave`／`steps` 是整數。begin 先讀取控制器設定、建立會話，**不會標零**。

本例應收到 `boot` 相同、`requestSid:1`、`requestSeq:1`、`result:"applied"`、`state:"await_zero"`。`accepted` 只是處理中，不能當完成。

從 begin 成功開始，Edge／操作工具應**每 10 秒**向命令 topic 發送心跳，直到完成／取消；同樣關閉 Retain：

```JSON
{
  "op": "ping",
  "boot": "填入本輪boot",
  "sid": 1
}
```

超過 60 秒沒有有效心跳，會話會逾時。**不要只靠在聊天中往返貼訊息維持會話**；下節操作工具每 10 秒自動送 ping，query 不能替代 ping。134 起，原重量 payload 的 `calibration.reason=lease_renewed`、`requestSid` 等於本輪 sid、`requestSeq=0` 才表示 OPTA 已續期；`lease_not_active` 表示本輪未續期，先 query。MQTT PUBACK 或工具顯示「已送出」都不能代替這份設備回覆。

**搬放砝碼可以慢慢操作，不要求在 60 秒內完成。**工具等你按 Enter 期間會持續保活；請確保工具與連線保持運作。放好並按 Enter 後，工具預設最多等 **5 分鐘**取得至少5秒穩定量測及ready；需要更久時，在操作命令加 `--stable-timeout 600` 可改為 10 分鐘。重量仍變動時只等待、不寫入；到時仍未穩定會停止並提示本步尚未送出，先 query 檢查，不自行跳到下一段。

這與通訊失聯不同：**6.0.133 起**，短暫採樣失敗會將當筆重量／量測時間標為 null，ready=false；135恢復後重新等待至少5秒穩定量測。連續達 10 秒沒有有效量測，或偵測感測器故障／設定改變，才鎖為 unknown，需明確取消並重做。持續收到有效但不穩定的重量，不會觸發這個 10 秒門檻。132 及以前仍是單次採樣失敗即中止；請先核對版本。

## 4. 三段校正順序表

下表以本輪 sid=1、每步皆成功為例；boot 全程沿用同一個最新開機字串。

| 步驟／命令 | 現場動作與累計重量 | `sid` | `seq` | applied 回覆的狀態 |
|---|---|---|---|---|
| 開始 `begin` | 清空秤盤；安全秤量 `capacityKg:"200"` | 1 | 1 | await_zero |
| 標零 `zero` | 保持空秤，不放砝碼 | 1 | 2 | collecting，step=0 |
| 第一段 `point` | 放 5 kg；總量 **5 kg**，`totalKg:"5"` | 1 | 3 | collecting，step=1 |
| 第二段 `point` | 保留 5 kg，再加 20 kg；總量 **25 kg**，`totalKg:"25"` | 1 | 4 | collecting，step=2 |
| 第三段 `point` | 保留 25 kg，再加 20 kg；總量 **45 kg**，`totalKg:"45"` | 1 | 5 | collecting，step=3 |
| 收尾 `finish` | 保持最後重量，等新穩定樣本 | 1 | 6 | complete |

每筆 zero／point／finish 仍需帶 `boot`、`sid`、`seq`、`slave:3`、`steps:3`。**totalKg 是已知砝碼累計總量，不是本次新增量，也不是目前面板讀值。**Edge 頁面可讓人輸入 5、20、20，但應累加成 5、25、45 再送。

## 5. 135：sample 是可持續使用的穩定憑證

放妥砝碼後保持不動。OPTA 持續取得有效且控制器stable的重量，至少 **5秒** 全區段最高－最低差≤**0.1 kg**，就會回報 `ready:true`。begin 可加 `"toleranceKg":"0.1"` 調整最大差；不是 ±0.1 kg，也不是校正精度標準。

看到同一boot/sid/revision、`valid:true`、`stable:true`、`ready:true` 後，複製 `sample` 與 `revision` 到 zero／point／finish。**sample不再每秒加1，可以手動填寫；每次新採樣改加 reading。**送出前確認sample仍相同，且同一筆 `calibration.ready`、`calibration.valid`、`calibration.stable` 仍全部為布林值 `true`；ping必須持續。重量超出全區段範圍、失穩、讀取失敗或校正步驟改變後，舊sample失效，重新穩定才可用新sample。不能拿partial狀態的數字續做。

OPTA 寫入前仍重讀重量並檢查範圍；最近成功量測須≤2500ms，並未放寬成接受斷線舊值。工具以reading去重，人工確認後再接收三筆新reading及ready；MQTTX核對同一筆

`calibration.ready=true`、`calibration.valid=true`、`calibration.stable=true`，以及目前會話與步驟。**134以前v1的sample仍每次採樣遞增，不適用本節。**

若收到 `rejected/not_fresh_stable`，未執行寫入；保留相同總量與seq，重新確認後換用新憑證。ACK沒收到則只能原封不動重送一次，仍無回覆就停止query；unknown停止，不重播寫入。

## 完整六步 JSON 範例：begin → zero → point → point → point → finish

**第一段、第二段、第三段的 `op` 都填 `point`**，不是 first／second／third，也不是 begin。控制器依本輪進度決定段號，命令不另填 step；`totalKg` 分別是 `"5"`、`"25"`、`"45"`。

| 動作 | `sid` | `seq` |
|---|---|---|
| 開始 `begin` | 1 | 1 |
| 標零 `zero` | 1 | 2 |
| 第一段 `point`，累計 5 kg | 1 | 3 |
| 第二段 `point`，累計 25 kg | 1 | 4 |
| 第三段 `point`，累計 45 kg | 1 | 5 |
| 收尾 `finish` | 1 | 6 |

六筆都發到 `mes/gateway/<設備UID>/cmd/calibration`，QoS 1、Retain 關閉；只用原重量 topic 接收 `calibration` 回覆。每一步完成後，才進下一步；期間每 10 秒的 ping 照常送，ping 不占 seq。

**下列是完整格式示範，不是可預排的一串寫入指令。** `BOOT_FROM_QUERY` 一律替換成最新 query 的 boot；sid=1 僅適用設備目前 sid=0、沒有進行中會話。sample 的 1／3／5／7／8 與 revision 的 1～5 都是示意數字，正式發送時必須從目前ready回報複製實際整數，不能照抄。不同 revision 或已失效 sample 會被拒絕。

**每一步都看兩種訊息：先核對成功 ACK，再看下一步的穩定資料。** 以下完整回覆範例由135實際程式的主機測試產生，boot改為占位字串；屬格式範例，不是本次現場讀值或精度證據。回覆都在原重量 topic `mes/gateway/<設備UID>/scale/weight`，不發到命令topic，也不必另訂閱 status。

| 看到的回覆 | 代表什麼 | 此時做什麼 |
| --- | --- | --- |
| `result=accepted`、`state=processing` | 已受理，控制器交易還沒完成 | 留在本步等待，不重按或進下一段 |
| 匹配 `boot/slave/requestSid/requestSeq` 的 `result=applied`，且state/step符合本步 | 本次指令成功；ACK通常 `ready=false`，因下一步穩定資格已重置 | 保存ACK，再依步驟確認秤盤／放置砝碼；不能拿ACK內舊sample直接做下一步 |
| `result=state`、`reason=sample`、`ready/valid/stable=true` | 新的穩定重量與可用憑證；不是指令完成ACK | 保持砝碼不動，從這筆複製sample/revision送下一步；不能用它代簽前一步 |
| `rejected`、`unknown`、`partial` 或ACK逾時 | 未完成，或結果／會話需處理 | 停止前進，依上節拒絕／重送規則處理 |

**一直等不到ready時，先看state與boot：** `recovery_required/sid=0` 表示目前開機尚未建立會話；`partial` 表示會話已結束，繼續等重量穩定或補ping都不會恢復。若boot與begin不同，先查重啟原因，停止使用舊boot；查明後重新query，確認沒有進行中會話，再用最新boot與大於目前sid的新編號begin。begin成功後每10秒ping，並核對 `reason=lease_renewed`；只有發出ping或收到一般重量，不能證明已續期。query不會續期。

下列以「量測時間」範本為例，成功ACK可見 `weight/stable/timestamp=null`，因它是狀態事件；這不表示校正失敗。若範本使用 `${sys.timestamp}`，ACK的timestamp通常仍是訊息時間字串。後續有效量測會另送重量。`revision/sample/reading` 均依實際回覆，不由範例推算；同一份ACK也可能由兩個Converter重複送達，只算一次。

### 第一步：開始 begin（seq=1）

先清空秤盤，確認安全秤量 200 kg，送：

```JSON
{
  "op": "begin",
  "boot": "BOOT_FROM_QUERY",
  "sid": 1,
  "seq": 1,
  "slave": 3,
  "steps": 3,
  "unit": "kg",
  "model": "tda08b",
  "capacityKg": "200",
  "toleranceKg": "0.1"
}
```

等匹配 boot、requestSid=1、requestSeq=1，且 `result:"applied"`、`state:"await_zero"`。begin 只建立會話／讀取設定，還沒有標零。

**預期收到：成功 ACK（完整重量 payload 範例）**

```JSON
{
  "weight": null,
  "stable": null,
  "timestamp": null,
  "calibration": {
    "v": 2,
    "boot": "BOOT_FROM_QUERY",
    "slave": 3,
    "sid": 1,
    "seq": 1,
    "revision": 1,
    "state": "await_zero",
    "step": 0,
    "steps": 3,
    "sample": 0,
    "reading": 0,
    "toleranceRaw": 10,
    "stableMs": 5000,
    "sampleEvent": false,
    "ready": false,
    "valid": false,
    "stable": false,
    "weightRaw": null,
    "decimals": 2,
    "result": "applied",
    "reason": "controller_ack",
    "requestSid": 1,
    "requestSeq": 1
  }
}
```


**核對與下一步：**這筆確認的是begin：`requestSid=1/requestSeq=1`、`applied`、`await_zero`、`step=0`。還沒標零，接著等下方三個旗標全部為true的重量回報；這筆ACK本身

不要求ready／valid／stable為true。

**接著預期收到：空秤已穩定、可以標零的重量回報**

**送zero前，以下條件必須同時成立：**

| 核對同一筆最新回報的欄位 | 必須符合 |
| --- | --- |
| `calibration.boot`、`sid`、`slave` | 與目前成功建立的會話一致；本例sid=1、slave=3，現場用實際值 |
| `calibration.state`、`step` | `state="await_zero"`、`step=0` |
| `calibration.ready` | 布林值 `true`：已滿足本步的穩定時間與新鮮度條件 |
| `calibration.valid` | 布林值 `true`：這筆校正量測有效且未過期 |
| `calibration.stable` | 布林值 `true`：控制器回報重量穩定 |

**三個旗標要在同一筆回報中全部為true，不能分別從不同訊息湊齊。**檢查的是 `calibration` 內的欄位，外層 `stable:1` 不能代替 `calibration.stable:true`。任一旗標為false就不要送zero：await_zero時繼續保活、保持空秤並等新有效穩定回報；partial／unknown／recovery_required則依前述中止處理，不能繼續等或寫入。

此例空秤0.01kg、`sample=1/revision=1`；新reading會繼續增加，同一穩定區段sample保持不變。全部條件符合後，從**同一筆**回報複製sample和revision到zero，送出前再確認仍符合；不能抄範例數字或之前的成功ACK。

```JSON
{
  "weight": 0.01,
  "stable": 1,
  "timestamp": "2026-10-05T08:00:00.237Z",
  "calibration": {
    "v": 2,
    "boot": "BOOT_FROM_QUERY",
    "slave": 3,
    "sid": 1,
    "seq": 1,
    "revision": 1,
    "state": "await_zero",
    "step": 0,
    "steps": 3,
    "sample": 1,
    "reading": 7,
    "toleranceRaw": 10,
    "stableMs": 5000,
    "sampleEvent": true,
    "ready": true,
    "valid": true,
    "stable": true,
    "weightRaw": 1,
    "decimals": 2,
    "result": "state",
    "reason": "sample",
    "requestSid": 0,
    "requestSeq": 0
  }
}
```

這筆 `result=state/requestSeq=0` 只提供穩定憑證，不能當成zero成功。它對應下方zero指令的sample/revision；現場以最新回覆取代範例。

### 第二步：空秤標零 zero（seq=2）

保持空秤，等至少5秒有效穩定量測；確認同一筆目前會話的 `calibration.state="await_zero"`、`step=0`，且 **`ready=true`**、**`valid=true`**、**`stable=true`** **全部成立**。假設從這筆回報複製的sample=1、revision=1，格式如下：

```JSON
{
  "op": "zero",
  "boot": "BOOT_FROM_QUERY",
  "sid": 1,
  "seq": 2,
  "slave": 3,
  "steps": 3,
  "sample": 1,
  "revision": 1
}
```

等 requestSeq=2 的 applied，state=collecting、step=0。再確認新重量讀值；此時才準備放第一顆砝碼。

**預期收到：成功 ACK（完整重量 payload 範例）**

```JSON
{
  "weight": null,
  "stable": null,
  "timestamp": null,
  "calibration": {
    "v": 2,
    "boot": "BOOT_FROM_QUERY",
    "slave": 3,
    "sid": 1,
    "seq": 2,
    "revision": 2,
    "state": "collecting",
    "step": 0,
    "steps": 3,
    "sample": 1,
    "reading": 7,
    "toleranceRaw": 10,
    "stableMs": 5000,
    "sampleEvent": false,
    "ready": false,
    "valid": false,
    "stable": false,
    "weightRaw": null,
    "decimals": 2,
    "result": "applied",
    "reason": "controller_ack",
    "requestSid": 1,
    "requestSeq": 2
  }
}
```

**核對與下一步：**這筆確認的是zero：`requestSeq=2`、`applied`、`collecting/step=0`。`revision=2`是本例的新狀態；再查看空秤重量是否合理，才放第一顆5kg。下一步憑證須在砝

碼放妥後重新取得。

### 第三步：第一段 point，累計 5 kg（seq=3）

放上 **5 kg**，放妥後等至少5秒穩定量測。確認同一筆目前會話的 `calibration.state="collecting"`、`step=0`，且 **`ready=true`**、**`valid=true`**、**`stable=true`** **全部成立**，再複製該筆sample/revision；假設為sample=3、revision=2：

```JSON
{
  "op": "point",
  "boot": "BOOT_FROM_QUERY",
  "sid": 1,
  "seq": 3,
  "slave": 3,
  "steps": 3,
  "totalKg": "5",
  "sample": 3,
  "revision": 2
}
```

等 requestSeq=3 的 applied，state=collecting、step=1。

**預期收到：成功 ACK（完整重量 payload 範例）**

```JSON
{
  "weight": null,
  "stable": null,
  "timestamp": null,
  "calibration": {
    "v": 2,
    "boot": "BOOT_FROM_QUERY",
    "slave": 3,
    "sid": 1,
    "seq": 3,
    "revision": 3,
    "state": "collecting",
    "step": 1,
    "steps": 3,
    "sample": 3,
    "reading": 19,
    "toleranceRaw": 10,
    "stableMs": 5000,
    "sampleEvent": false,
    "ready": false,
    "valid": false,
    "stable": false,
    "weightRaw": null,
    "decimals": 2,
    "result": "applied",
    "reason": "controller_ack",
    "requestSid": 1,
    "requestSeq": 3
  }
}
```

**核對與下一步：**這筆確認的是第一段：`requestSeq=3`、`applied`、`collecting/step=1`。接著核對實際重量，再保留5kg並加上20kg；放妥後等ready，複製新的sample/revision。

### 第四步：第二段 point，累計 25 kg（seq=4）

**保留原有 5 kg**，再加一顆 20 kg。放妥後等至少5秒穩定量測。確認同一筆目前會話的 `calibration.state="collecting"`、`step=1`，且 **`ready=true`**、**`valid=true`**、**`stable=true`** **全部成立**，再複製該筆sample/revision；假設為sample=5、revision=3：

```JSON
{
  "op": "point",
  "boot": "BOOT_FROM_QUERY",
  "sid": 1,
  "seq": 4,
  "slave": 3,
  "steps": 3,
  "totalKg": "25",
  "sample": 5,
  "revision": 3
}
```

等 requestSeq=4 的 applied，state=collecting、step=2。這筆 totalKg 不能填 `"20"`，因為秤上累計是 25 kg。

**預期收到：成功 ACK（完整重量 payload 範例）**

```JSON
{
  "weight": null,
  "stable": null,
  "timestamp": null,
  "calibration": {
    "v": 2,
    "boot": "BOOT_FROM_QUERY",
    "slave": 3,
    "sid": 1,
    "seq": 4,
    "revision": 4,
    "state": "collecting",
    "step": 2,
    "steps": 3,
    "sample": 5,
    "reading": 31,
    "toleranceRaw": 10,
    "stableMs": 5000,
    "sampleEvent": false,
    "ready": false,
    "valid": false,
    "stable": false,
    "weightRaw": null,
    "decimals": 2,
    "result": "applied",
    "reason": "controller_ack",
    "requestSid": 1,
    "requestSeq": 4
  }
}
```

**核對與下一步：**這筆確認的是第二段：`requestSeq=4`、`applied`、`collecting/step=2`。接著核對實際重量，再加最後一顆20kg至總量45kg；放妥後等ready及新憑證。

### 第五步：第三段 point，累計 45 kg（seq=5）

**保留原有 25 kg**，再加另一顆 20 kg。放妥後等至少5秒穩定量測。確認同一筆目前會話的 `calibration.state="collecting"`、`step=2`，且 **`ready=true`**、**`valid=true`**、**`stable=true`** **全部成立**，再複製該筆sample/revision；假設為sample=7、revision=4：

```JSON
{
  "op": "point",
  "boot": "BOOT_FROM_QUERY",
  "sid": 1,
  "seq": 5,
  "slave": 3,
  "steps": 3,
  "totalKg": "45",
  "sample": 7,
  "revision": 4
}
```

等 requestSeq=5 的 applied，state=collecting、step=3。三段已完成，但還不能跳過 finish。

**預期收到：成功 ACK（完整重量 payload 範例）**

```JSON
{
  "weight": null,
  "stable": null,
  "timestamp": null,
  "calibration": {
    "v": 2,
    "boot": "BOOT_FROM_QUERY",
    "slave": 3,
    "sid": 1,
    "seq": 5,
    "revision": 5,
    "state": "collecting",
    "step": 3,
    "steps": 3,
    "sample": 7,
    "reading": 43,
    "toleranceRaw": 10,
    "stableMs": 5000,
    "sampleEvent": false,
    "ready": false,
    "valid": false,
    "stable": false,
    "weightRaw": null,
    "decimals": 2,
    "result": "applied",
    "reason": "controller_ack",
    "requestSid": 1,
    "requestSeq": 5
  }
}
```

**核對與下一步：**這筆確認的是第三段：`requestSeq=5`、`applied`、`collecting/step=3`。此時仍未收尾；保持45kg不動，再等至少5秒穩定及ready，才送finish。

### 第六步：收尾 finish（seq=6）

保持最後砝碼不動，等上一段完成後的至少5秒穩定量測。確認同一筆目前會話的 `calibration.state="collecting"`、`step=3`，且 **`ready=true`**、**`valid=true`**、**`stable=true`** **全部成立**，再複製該筆sample/revision；假設為sample=8、revision=5：

```JSON
{
  "op": "finish",
  "boot": "BOOT_FROM_QUERY",
  "sid": 1,
  "seq": 6,
  "slave": 3,
  "steps": 3,
  "sample": 8,
  "revision": 5
}
```

等 requestSeq=6 的 applied，`state:"complete"`、step=3、steps=3，才結束會話與心跳。finish 不需要 totalKg，也不是再發一筆 `point` 並填重量 0；收尾所需的控制器寫入由韌體處理。之後另驗實際精度及保存。

**預期收到：成功 ACK（完整重量 payload 範例）**

```JSON
{
  "weight": null,
  "stable": null,
  "timestamp": null,
  "calibration": {
    "v": 2,
    "boot": "BOOT_FROM_QUERY",
    "slave": 3,
    "sid": 1,
    "seq": 6,
    "revision": 6,
    "state": "complete",
    "step": 3,
    "steps": 3,
    "sample": 8,
    "reading": 49,
    "toleranceRaw": 10,
    "stableMs": 5000,
    "sampleEvent": false,
    "ready": false,
    "valid": false,
    "stable": false,
    "weightRaw": null,
    "decimals": 2,
    "result": "applied",
    "reason": "controller_ack",
    "requestSid": 1,
    "requestSeq": 6
  }
}
```

**核對與下一步：**這筆確認的是finish：`requestSeq=6`、`applied`、`complete/step=3/steps=3`。此時停止ping；`ready=false`正常，不必等它重新變true。之後

才做降載、精度與斷電保存驗收。

任一步 rejected／unknown／ACK 逾時，都先依上文處理，不繼續照序送下一筆。以上 sample 間隔只是範例，現場可能因等待更久而跳到更大的數字。

### 6. 正確判讀回覆，再驗實際重量

只有 `boot`、`requestSid`、`requestSeq` 都對上送出的命令，且 `result:"applied"`、state／step 符合上表，才算這一步完成。`result:"state"`／`reason:"snapshot"` 是狀態快照；仍有重量、MQTT 發送成功或 accepted 都不是完成 ACK。applied 代表控制器交易已回覆，不等於秤重精度或斷電保存驗收通過。

**尚未開始時的常見訊息**：外層 weight=49.70、stable=1，但內層 calibration 是 recovery_required、sid=0、sample=0、valid=false、weightRaw=null。這表示一般輪詢有重量，校正流程尚無有效樣本；不代表已證明磅秤故障。內層 decimals=0 也不能當成控制器實際小數位。一般重量品質不要用 calibration.valid 判斷；進入校正後則須遵守上節校正樣本條件。

看到 complete 後，再依業主允收誤差核對空秤、5／25／45 kg 及點間重量，保存面板值與 MQTT 原始資料。斷電保存另外驗收，不把 complete 當成保存證據。

---

04

## 重量 payload：完整範例與每個欄位

以下是現場「尚未開始本輪校正」的實際訊息。Topic 是 `mes/gateway/003100443033511034323932/scale/weight`，接收 QoS 為 0；Topic／QoS 是 MQTT 訊息資訊，不是 JSON 欄位。這是說明用紀錄，時間與 boot 不可當成未來命令的固定值。

JSON


```
{
  "weight": -0.28,
  "stable": 1.0,
  "timestamp": "2026-10-06T06:10:53.047Z",
  "calibration": {
    "v": 2,
    "boot": "d9bb1ae411a03682",
    "slave": 3,
    "sid": 0,
    "seq": 0,
    "revision": 0,
    "state": "recovery_required",
    "step": 0,
    "steps": 0,
    "sample": 0,
    "reading": 0,
    "toleranceRaw": 0,
    "stableMs": 5000,
    "sampleEvent": false,
    "ready": false,
    "valid": false,
    "stable": false,
    "weightRaw": null,
    "decimals": 0,
    "result": "state",
    "reason": "snapshot",
    "requestSid": 0,
    "requestSeq": 0
  }
}
```

## 外層：重量與時間

| 欄位 | 型別／範例 | 意義與 Edge 用法 |
| --- | --- | --- |
| `weight` | 數字或 null；-0.28 | 重量。此現場已確認單位為 kg；JSON 本身沒有 unit 欄位，不可憑欄名推論其他設備的單位。平時取一般輪詢值；校正採樣事件使用該次直接採樣，無效時為 null，不用 0 代替。 |
| `stable` | 數字或 null；1.00 | 外層穩定值；本樣板 1 表示穩定、0 表示不穩定，校正無效樣本可為 null。1.00 與 1 是同一個 JSON 數字，不是布林值。這一個欄位不足以判斷可以執行下一段。 |
| `timestamp` | UTC 字串或 null | **取決於範本所用變數**：`${sys.timestamp}` 是訊息組合時間；`${rs485.3.實時重量.timestamp}` 是重量讀取時間。6.0.128 起如 `.047Z` 有三位毫秒，Z 表示 UTC；本次開機未對時為 null。量測變數亦在無有效樣本時為 null。訊息時間已知不代表重量有效，毫秒解析度不保證 UTC 毫秒準確度。 |
| `calibration` | JSON 物件 | 校正會話、樣本狀態與命令回覆。未進入校正時，其 valid=false 不否定外層一般輪詢重量。 |

需要同時保存兩種時間時，用不同欄位，例如 `"timestamp":${sys.timestamp},"measured_at":${rs485.3.實時重量.timestamp}`。不要讓同一 topic 的兩個範本使用同名 timestamp 卻代表不同時間，令接收端無法區分。

## calibration：身分與進度

以下表格的欄位全部位於 `calibration` 物件內。

| 欄位 | 型別／本例 | 意義與 Edge 用法 |
| --- | --- | --- |
| `v` | 整數；2 | 校正 payload 的協定版本，不是韌體版本。接收端應先確認支援此結構。 |
| `boot` | 字串；16 位十六進位 | OPTA 本次啟動的校正識別碼。從 query／非 retained 重量回覆取得，原樣帶回命令。改變即停止舊會話；不能以 UID、Build ID 或 bootCount 代替。 |
| `slave` | 整數；3 | 這筆重量所屬的 RS485 站號。Edge 必須核對目標磅秤，不能只看 sid。 |
| `sid` | 整數；0 | 目前／最近一次校正會話編號；本例 0 表示本次開機尚未建立。新會話用較大的 sid，進行中不得自動接管。 |
| `seq` | 整數；0 | 目前會話最後已消耗的命令序號。applied 或結果 unknown 都可能消耗序號；不是收到訊息的累計次數。不能只見 seq 增加就判斷成功。 |
| `revision` | 整數；0 | 校正狀態修訂號。動作生效／結果未知、失效或逾時等轉變可能增加；Edge 以它排除前一階段的樣本，送 zero／point／finish 時帶回最新值。不是韌體版本，也不是 seq。 |
| `state` | 字串；recovery_required | 校正流程所在狀態，見下表。 |
| `step` | 整數；0 | 已成功套用的正重量校正點數；標零後仍為 0，完成第一個重量點才是 1。 |
| `steps` | 整數；0 | 本輪要求的正重量點數：開始後為 3 或 5；本例尚未建立所以是 0。不包含標零／收尾。 |

## calibration：樣本與可操作條件

| 欄位 | 型別／本例 | 意義與 Edge 用法 |
| --- | --- | --- |
| `sample` | 整數；0 | v2穩定區段編號，同區段保持不變；失效後新區段遞增。只有ready才可作為校正憑證，不能自行填步數。 |
| `reading` | 整數 | v2每次採樣嘗試流水號，失敗也增加；多路Converter用reading去重。 |
| `toleranceRaw` | 整數 | 本輪最大－最小容許差，除以10^decimals為kg；不是精度驗收標準。 |
| `stableMs` | 整數；5000 | 穩定觀察至少5秒。 |
| `sampleEvent` | 布林；false | true 表示這筆是一次校正採樣事件；false 是狀態／命令回覆或一般快照。同一 boot／sid／revision／reading 重複抵達只算一次。 |
| `ready` | 布林；false | 目前 active 會話已累積足夠連續穩定且新鮮的校正樣本，可供下一步檢查。仍需操作者確認空秤／砝碼放妥、命令順序正確；不是自動寫入授權。 |
| `valid` | 布林；false | 校正樣本有效且仍在新鮮度期限內。不是一般 RS485 online，也不是外層重量的通用有效旗標。本例尚無校正樣本，所以為 false。 |
| `stable` | 布林；false | 校正樣本有效、新鮮且穩定。與外層數字 stable 來源／生命週期不同；單獨 true 不代表已有 3 筆穩定新樣本。 |
| `weightRaw` | 有號整數或 null；null | 校正流程直接讀到的原始重量整數，並非要提交的砝碼總量。有效時重量 kg = weightRaw ÷ 10^decimals；例如 4970、decimals=2 對應 49.70 kg，負值亦可。null 表示無有效新鮮校正樣本，不能當 0。 |
| `decimals` | 整數；0 | 本輪從控制器讀取的小數位數，用來換算 weightRaw 與命令公斤值。尚未 begin 時的 0 是初始值，不能推論磅秤真的是零位小數；有效會話可能合法使用 0。 |

## calibration：命令結果

| 欄位 | 型別／本例 | 意義與 Edge 用法 |
| --- | --- | --- |
| `result` | 字串；state | `state`=狀態回報；`accepted`=處理中；`applied`=動作已取得所需回覆；`rejected`=拒絕；`unknown`=控制器結果不確定。begin／cancel 不寫校正點，不能把其 applied 當成曲線已寫入。 |
| `reason` | 字串；snapshot | 補充原因碼。snapshot=狀態快照、sample=採樣、processing=處理中、controller_ack=已完成協定動作；拒絕／中斷可能是 boot_mismatch、not_fresh_stable、weight_order_or_range、lease_expired、measurement_invalid（舊版）等。133 起新增 read_timeout（未收到回應）、read_frame（長度／標頭錯誤）、read_crc（校驗錯誤）、read_unavailable（讀取介面不可用）；短暫失敗不消耗 seq。measurement_gap=有效讀取中斷達 10 秒、settings_changed=倍率／分度值改變、config_changed=設備／映射／匯流排設定不符、sensor_fault=感測器故障，這些會停止會話。保留完整原因碼供診斷，不只顯示「失敗」。 |
| `requestSid` | 整數；0 | 此回覆所對應請求的 sid。校正 ACK 應與待處理命令匹配；一般採樣／query通常為0；134起ping續期回覆為請求的sid，requestSeq固定0，不能當作「會話已清除」。 |
| `requestSeq` | 整數；0 | 此回覆所對應請求的 seq，與 requestSid、boot 一起核對。部分早期格式／boot 檢查失敗的 rejected 也可能沒有帶回請求編號；不得誤配到待處理命令。 |

收到新快照可能覆蓋 result/reason 顯示，因此 Edge 必須處理訊息事件、記錄已匹配的 ACK，不能只看畫面最後一筆文字。QoS0 可能遺失回覆；MQTT QoS1 也只保證傳輸語意，不能取代這組應用層 ACK。

### state：顯示文字與下一步

| state | 建議顯示 | 該怎麼做 |
|---|---|---|
| `recovery_required` | 尚未建立本輪校正／需重新確認 | 開機預設狀態，無法證明以前曲線是否完整；query 後從清空秤盤完整開始。不能單憑此值判定硬體故障。 |
| `await_zero` | 等待空秤標零 | 確認空秤，等 3 筆穩定新樣本，再送 zero。 |
| `collecting` | 校正中，已完成 step／steps 段 | 依三段表加入砝碼；做滿指定段數並等新穩定樣本後才能 finish。 |
| `processing` | 控制器處理中 | 等匹配的最終結果，不再送下一個動作。 |
| `complete` | 本輪命令流程完成 | 再驗實際精度與保存；不是已完成所有出廠驗收。 |
| `partial` | 流程中斷／已取消 | 不能假設已寫入的點被回滾；確認後完整重做。 |
| `unknown` | 寫入結果不確定，或量測／設定異常使流程停止 | 停止加砝碼及下一段命令，query、明確 cancel 後重做；不要直接重送新序號。 |
| `idle` | 此重量來源不屬於目前校正會話 | 多磅秤時核對 slave；不能套用其他磅秤的 sid／樣本進行校正。 |

**這筆範例的結論**：一般重量是 -0.28、穩定值為 1；本次開機尚未建立校正會話，所以 calibration 沒有有效樣本、也沒有命令 ACK。下一步是 query 核對後 begin，並非直接 point 或 finish。

05

## 現場操作工具：依作業系統下載與安裝

使用業主 Edge 網頁者不必安裝本工具。**沒有 Edge 頁面、由電腦發 MQTT 命令時**，請依下列步驟準備。電腦經網路連到 MQTT broker；USB 線不是本工具的通訊路徑，也不需要 USB-RS485 轉接器。OPTA 到磅秤的 RS485 接線仍須正常。

需下載兩個檔案，放在同一個 `scale-calibration` 資料夾：

- scale_edge_sim.py：現場 MQTT 操作程式
- calibration-requirements.txt：Python 相依套件清單

下面指令會直接下載這兩個檔案。若改用瀏覽器且點開只顯示文字，請用「另存連結為」存檔；確認名稱不是 `.py.txt`，也不是登入頁面的 HTML。**現場連線只需這兩個檔案，不必下載完整專案、PlatformIO 或 C++ 編譯器。**

## 安裝 Python 與下載工具

頁籤會依瀏覽器回報的作業系統，自動選擇 **macOS / Linux** 或 **Windows**；也可手動切換。遠端操作另一種系統時，請手動選擇目標電腦的頁籤。PDF 會同時列出兩種指令。

先安裝 Python 3（建議 3.10 以上）：

- **Windows**：由 Python 官方下載頁 安裝，重新開啟 PowerShell，以 `py -3 --version` 確認。找不到 `py` 時先修復安裝。
- **macOS**：在「終端機」執行 `python3 --version`；未安裝時由同一官方下載頁取得 macOS 安裝程式。
- **Linux**：需 Python 3、venv 與 curl。Ubuntu／Debian 可由管理者執行 `sudo apt-get install python3 python3-venv python3-pip curl`；其他發行版使用對應套件。

依頁籤執行下列指令，建立工作目錄、下載兩個檔案及安裝相依套件：

```bash
# macOS / Linux
mkdir -p "$HOME/scale-calibration"
cd "$HOME/scale-calibration"
curl --fail --location https://opta.smms.com.tw/docs/assets/scale_edge_sim.py
--output scale_edge_sim.py
curl --fail --location https://opta.smms.com.tw/docs/assets/calibration-
requirements.txt --output calibration-requirements.txt
python3 -m venv .venv
./.venv/bin/python -m pip install -r calibration-requirements.txt
./.venv/bin/python ./scale_edge_sim.py --help
```

```powershell
# Windows PowerShell
New-Item -ItemType Directory -Force "$HOME/scale-calibration" | Out-Null
Set-Location "$HOME/scale-calibration"
Invoke-WebRequest 'https://opta.smms.com.tw/docs/assets/scale_edge_sim.py' -
OutFile 'scale_edge_sim.py'
Invoke-WebRequest 'https://opta.smms.com.tw/docs/assets/calibration-
requirements.txt' -OutFile 'calibration-requirements.txt'
py -3 -m venv .venv
.\.venv\Scripts\python.exe -m pip install -r calibration-requirements.txt
.\.venv\Scripts\python.exe ./scale_edge_sim.py --help
```

看到含 `--host`、`--prefix`、`--run` 的說明即完成。本指南直接使用虛擬環境內的 Python，**不用執行 Activate.ps1，也不用修改 PowerShell 執行原則**。

## 設定本輪連線資料

macOS／Linux 頁籤的密碼輸入與陣列使用 Bash，請先依區塊第一行執行 `bash`。

以下範例使用 **TLS 8883**。請向現場管理者取得 broker、帳號、密碼、實際 topic 前綴及重量 topic；不要把 OPTA IP 當成 broker。`network.topicPrefix` 可由 `/api/system` 核對，重量 topic 依步驟 D 記錄。範例文字須換成現場值。

```bash
# macOS / Linux (Bash)
bash
cd "$HOME/scale-calibration"
BROKER='填入現場 MQTT 主機名稱'
PREFIX='mes/gateway/填入設備UID'
WEIGHT_TOPIC="$PREFIX/scale/weight"
SLAVE=3
read -r -p 'MQTT 帳號: ' MQTT_USER
read -r -s -p 'MQTT 密碼: ' MQTT_PASSWORD
printf '\n'
export MQTT_USER MQTT_PASSWORD
MQTT_ARGS=(--host "$BROKER" --prefix "$PREFIX" --weight-topic "$WEIGHT_TOPIC"
--slave "$SLAVE" --tls --port 8883)
```

```powershell
# Windows PowerShell
$Broker = '填入現場 MQTT 主機名稱'
$Prefix = 'mes/gateway/填入設備UID'
$WeightTopic = "$Prefix/scale/weight"
$Slave = 3
$env:MQTT_USER = Read-Host 'MQTT 帳號'
$Secret = Read-Host 'MQTT 密碼' -AsSecureString
$env:MQTT_PASSWORD = [System.Net.NetworkCredential]::new('', $Secret).Password
$MqttArgs = @('--host', $Broker, '--prefix', $Prefix, '--weight-topic',
$WeightTopic, '--slave', "$Slave", '--tls', '--port', '8883')
```

若重量 topic 曾修改，將 `$WeightTopic` ／ `WEIGHT_TOPIC` 改成真正的 topic，再建立參數陣列。站號不是 3 時也須修改。只有現場明確使用非 TLS 1883 時，才移除陣列中的 `--tls` 並將 `8883` 改為 `1883`；不要用關閉 TLS 解決憑證錯誤。匿名 broker 可略過帳密輸入，並先清除舊的 MQTT 環境變數。

帳密只保留在目前終端機環境，不填進文件或指令歷史。後續 query、校正、取消請在**同一個終端機與工作目錄**執行；重開視窗需重新設定以上變數。

06

# 接實機：先查、不校正

先使用上節設定好的參數查詢；這個命令不開始校正。

```bash
# macOS / Linux (Bash)
./.venv/bin/python ./scale_edge_sim.py "${MQTT_ARGS[@]}"
```

```powershell
# Windows PowerShell
.\.venv\Scripts\python.exe ./scale_edge_sim.py @MqttArgs
```

預期從原重量 payload 的 calibration 收到 boot/sid/seq/state；兩條流程若送出同一 reading，須去重，不能算兩筆穩定樣本。逾時時先查韌體是否支援、MQTT 連線與主題權限；不要反覆發校正命令。

## 開始三段校正（會寫入控制器）

先確認本輪允許寫入、秤盤可清空，且業主確認的安全秤量涵蓋 **45 kg**。以下由使用者輸入安全秤量，不提供可直接誤用的預設值。日誌存於工作目錄，檔名含時間，避免覆蓋前輪紀錄。

```bash
# macOS / Linux (Bash)
read -r -p '業主確認的安全秤量 (kg) : ' CAPACITY
LOG="scale-live-$(date +%Y%m%d-%H%M%S).jsonl"
./.venv/bin/python ./scale_edge_sim.py "${MQTT_ARGS[@]}" --run --capacity
"$CAPACITY" --increments 5,20,20 --log "$LOG"
```

```powershell
# Windows PowerShell
$Capacity = Read-Host '業主確認的安全秤量 (kg) '
$Log = 'scale-live-' + (Get-Date -Format 'yyyyMMdd-HHmmss') + '.jsonl'
.\.venv\Scripts\python.exe ./scale_edge_sim.py @MqttArgs --run --capacity
"$Capacity" --increments 5,20,20 --log "$Log"
```

五段時，將 `--increments` 後的值換成現場五次新增砝碼重量，例如 `5,5,10,10,15`（總量 45 kg）；依實際砝碼操作，不照抄不存在的砝碼。

1. 清空秤盤，按 Enter；程式以reading去重，等OPTA至少5秒穩定、ready才標零。
2. 放 5 kg，按 Enter；系統提交總量 5 kg。
3. **保留 5 kg**，再放 20 kg，按 Enter；提交總量 25 kg。
4. **保留原有 25 kg**，再放 20 kg，按 Enter；提交總量 45 kg。
5. 程式等新穩定樣本再收尾，看到 complete 後，用業主允許誤差逐點驗重量，保存面板值、上報值及實測誤差。真正斷電保存另依 OC.11，在取得本輪授權後執行，不以軟體重啟代替。

公斤到控制器整數的換算由 OPTA 讀取小數位後處理，操作人員不用去小數點。工具的 `--increments` 填每次新增砝碼重量；工具累加後才把已知總量送出，不抄尚未校準的面板數字。

---

07

## 出錯時停在哪裡

| 畫面結果 | 下一步 |
| --- | --- |
| rejected | 此請求未進行校正寫入；核對 reason。not_fresh_stable 會有限次重新等樣本，其他錯誤停下 |
| unknown | 控制器可能已生效；停止加砝碼，不自行換新 seq 重送 |
| ACK timeout | 程式只同內容重送一次；仍失敗即停止，先 query |
| partial／recovery_required | 上次可能只完成部分校正；清空後完整重做，不能視為原曲線已恢復 |
| valid=false／無新 reading | 不可進下一段；檢查 reason。133 起短暫唯讀通訊失敗可等恢復，135再重新累積至少5秒穩定量測；unknown 仍須取消重做 |
| 超過穩定等待時間 | 本步未送出，不增加 seq；先 query。搬放或振動需要更久，可用 --stable-timeout 調整下次操作工具的等待時間 |

如 query 顯示需要取消的會話 **sid=1**，先確認真的是該會話，再執行以下對應指令；其他 sid 必須替換，不固定填 1。

```bash
# macOS / Linux (Bash)
./.venv/bin/python ./scale_edge_sim.py "${MQTT_ARGS[@]}" --cancel-session 1
```

```powershell
# Windows PowerShell
.\.venv\Scripts\python.exe ./scale_edge_sim.py @MqttArgs --cancel-session 1
```

取消僅停止流程，**不回滾已寫入的校正點**。重新開始時程式會使用較大 sid。不要以「仍收到重量」判定上一個命令成功；只認匹配 requestSid/requestSeq/boot 的 applied。

整合 Edge 頁面時，只訂閱既有重量 topic，讀取同包 `calibration`；會話中將重量標示為校正用途，隔離生產統計。協定細節見 設計文件。

## 操作結束與安裝問題

保存工作目錄中的 `scale-live-日期時間.jsonl`，連同版本、UID、砝碼順序與實測誤差交給驗收人員。關閉視窗前清除帳密：

```bash
# macOS / Linux (Bash)
unset MQTT_USER MQTT_PASSWORD
```

```powershell
# Windows PowerShell
Remove-Item Env:MQTT_USER, Env:MQTT_PASSWORD -ErrorAction SilentlyContinue
Remove-Variable Secret -ErrorAction SilentlyContinue
```

| 遇到的問題 | 處理方式 |
| --- | --- |
| 找不到 Python／py | 完成對應系統的 Python 安裝後重新開終端機 |
| 找不到 scale_edge_sim.py／requirements | 回到 `scale-calibration` 資料夾，確認兩個下載檔案名稱與內容正確 |
| No module named paho | 使用上節同一個 `.venv` 的 Python 重跑 pip 安裝，不要改用系統 Python |
| Linux 無法建立 venv | 安裝發行版對應的 python3-venv 套件，再重試 |
| 下載或套件安裝失敗 | 停在此步，檢查網路／代理設定；不要把錯誤網頁當成腳本執行 |
| TLS 憑證錯誤／MQTT 連線失敗 | 請管理者核對主機名稱、電腦時間、信任憑證、帳密、port 與 topic 權限；不要跳過驗證 |

08

## 交付前要完成的驗收

依 現場 OC.1～OC.12 與 工站 Phase 5C 留逐項證據。工程附錄的兩輪 --simulate 通過只完成主機流程；實機、精度、真正斷電、業主 Edge 與長時結果仍須另驗。

09

## 工程附錄：電腦模擬三段／五段

**本節供已有完整專案的開發者使用，不是現場下載工具的安裝步驟。單檔下載不支援 `--simulate`；目前以下為 macOS／Linux 開發環境流程，Windows 原生模擬未驗證。**

電腦需有 Python、C++ 編譯器、PlatformIO 已下載的 ArduinoJson、`paho-mqtt` 與 `mosquitto`。在專案根目錄執行：

```sh
python3 scripts/scale_edge_sim.py --simulate --capacity 50 --increments
5,20,20 --log /tmp/scale-three.jsonl
```

預期依序看到零點 ACK、三段「總重量 5／25／45 kg」ACK，最後 state=`complete`。每段即使重量相同仍應有不同 sample 編號。模擬會自己開暫時的 localhost MQTT broker、關閉並清理，不連現場設備。這是流程驗證，不是磅秤精度驗證。

五段範例（輸入的是每次**新增**重量）：

```sh
python3 scripts/scale_edge_sim.py --simulate --capacity 50 --increments
5,5,10,10,15 --log /tmp/scale-five.jsonl
```

預期總量為 5／10／20／30／45 kg。`--capacity` 必須填業主確認的安全秤量，不能為了讓命令通過自行放大。

---

10

## 舊版參考

只有維護舊設備或既有流程時才使用以下文件；本次 126 線上會話請依本指南操作，不混用命令。

- 舊訊號觸發多段校正：既有 cmd/signal 流程。
- 舊版兩點校正精靈：兩點校正工具。

驗收與簽核仍統一依 現場驗收計畫。

---

11

## 2026-10-06 現場勘誤：TDA-08B 測量狀態

實測控制器版本為 **260521**（不是 OPTA 的 6.0.x）。FC03 讀位址 7、數量 2，高字先合併；`.46` Slave 3 回覆 `00 03 F9 A9`，即 260521。

| 感測器狀態 | 暫存器 200 | bit 4 | bit 3 |
|---|---|---|---|
| 正常且穩定 | `0x0018` | 1＝正常 | 1＝穩定 |
| 拆線、面板 ERR02 | `0x0008` | 0＝故障 | 仍可能為 1，不能單靠穩定判定有效 |

證據為 KC 提供的正常／拆線對照；KC 確認是在現場同一台控制器上實測。圖片的250829版本標示有誤；先前我們也把原始版本值算錯，130部署後重新讀取及換算確認260521。此勘誤不是原廠公告；實測證據只涵蓋現場控制器。KC 決定採用 TDA-08B 型號統一的正常高位策略，不以韌體版本白名單限制；這不表示已實測所有版本。原手冊表格保留作歷史對照。

OPTA **6.0.132 候選**對 TDA-08B 統一以 bit 4＝1 正常／0 故障檢查開始、寫入前、提交前與連續採樣，不讀版本來阻擋 begin。感測器故障的 begin 回覆 `sensor_fault`。拒絕 begin 不消耗 sid／seq、不寫入校正。通訊、倍率、穩定、新鮮樣本與交易檢查維持。129／130白名單誤設250829；131限制260521；132依KC要求移除版本限制。

儲存设备設定只同步小數位倍率；控制器2位、網頁0.01時直接儲存，不修正重量曲線。三段校正使用已知砝碼累計5／25／45 kg，不使用校正前錯誤示值。132安裝與實秤精度仍須各自驗收。