04 開發者 / 整合

CLI 指令參考

本指南說明如何透過終端機 (Terminal/命令提示字元) 對 MES Gateway 設備執行 OTA (Over-The-Air) 韌體更新。這在無法使用 Web UI，或是在自動化流水線中進行批次部署時非常有用。

情境說明：

- 如果您是**終端客戶或運保人員**，且手上只有一個 `.bin` 韌體檔案，請直接參考 **策略一 (cURL)**。這不需要安裝任何額外環境。
- 如果您是**內部開發人員**，且擁有完整原始碼專案，請參考 **策略二 (PlatformIO)**。

01

策略一：直接推送韌體二進位檔 (客戶最推薦)

如果您手上只有編譯好的 `.bin` 檔案 (例如 `mes-gateway-x.x.x.bin`)，最簡單的方式是使用 `curl` 指令直接推送給設備。現代操作系統 (Windows 10+, macOS, Linux) 都已經內建了 `curl`，**您完全不需要安裝 PlatformIO 或撰寫腳本**。

1. 準備前置資訊

- 開啟終端機 (Windows 請開 CMD 或 PowerShell，macOS 請開 Terminal)。
- 確認您的設備目前的 IP 位址 (例如 `192.168.51.32`)。
- 確認設備的 API 授權 Token (預設為 `admin-token`)。

2. 執行更新

執行以下 `POST` 請求到 `/api/ota/upload`：

```bash
# 請將 IP、Token 與 "@檔案路徑" 替換成您的實際參數
curl -X POST "http://192.168.51.32/api/ota/upload" \
  -H "Authorization: Bearer admin-token" \
  -H "Content-Type: application/octet-stream" \
  -H "X-Firmware-Version: 5.9.9" \
  --data-binary "@firmware.bin" \
  --connect-timeout 5 \
  --max-time 120
```

- **`--data-binary "@firmware.bin"`**：請修改 `@` 後方的路徑指向您的 `.bin` 檔案，若檔案與終端機不在同一個資料夾，請使用絕對路徑 (例如 `@C:/Users/abc/Desktop/firmware.bin`)。
- **`--max-time 120`**：上傳與寫入 Flash 時間較長，建議給予至少 60-120 秒以免連線提早中斷。

3. 預期行為

- 寫入成功後，設備會回傳 HTTP 200 OK，並在 3 秒內自動閃爍燈號並重啟。
- 若您的指令因為 Timeout 斷線 (`curl: (28) Operation timed out`)，但設備隨後有成功重啟亮燈，請不用擔心，這通常代表設備忙於重啟而未關閉連線，實際上已經寫入成功。

02

策略二：使用 Python 腳本推送 (附帶進度與防呆)

如果您覺得 `curl` 指令太長、容易打錯，開發團隊也可以將專案內的 `scripts/pio_ota.py` 與 `.bin` 一起打包交給客戶 (客戶電腦需安裝 Python 3)。

這支腳本內建了完整的容錯機制與中文進度提示。

執行更新

在終端機中，切換到檔案所在目錄，並執行以下指令：

```bash
# 用法：python3 pio_ota.py <韌體檔案> <設備IP>
python3 pio_ota.py mes-gateway-v5.9.9.bin 192.168.51.32

# 如果設備自訂了 admin token（與預設 'admin-token' 不同），用 OTA_TOKEN env：
OTA_TOKEN=smmsadmin python3 pio_ota.py mes-gateway-v5.9.120.bin 192.168.72.77
```

腳本預設讀 `OTA_TOKEN` env，沒設就 fallback 到 `admin-token`。中途若斷線也會顯示友善的提示，減少客戶的疑慮。

03

策略三：使用 PlatformIO 開發環境直接上傳 (開發者專用)

如果您是內部團隊且擁有完整的 PlatformIO (PIO) 原始碼專案，您可以直接透過指令觸發本機端編譯與上傳，專案已為此內建了處理邏輯。此策略適合開發階段使用。

前置作業：安裝 PlatformIO CLI

如果您尚未安裝 PlatformIO 的命令列工具，可以透過以下任一方式安裝：

- **使用 Homebrew (macOS/Linux)：**

```bash
brew install platformio
```

- **使用 Python `pip` (全平台)：**

```bash
pip install -U platformio
```

安裝完成後，可執行 `pio --version` 確認。

執行 PIO 更新

在專案根目錄執行以下指令：

```bash
pio run -e opta_ota -t upload
```

- 這個指令會自動觸發編譯（若程式碼有更動），並使用自訂腳本 `scripts/pio_ota.py` 將剛編譯出的 `.bin` 發送到預設的設備 IP（`192.168.51.32`）。
- 如果您需要上傳至不同 IP 的設備，可以覆寫環境變數：

```bash
pio run -e opta_ota -t upload --upload-port 192.168.1.100
```

- 腳本內建了容錯處理。若上傳成功但設備為加速重啟而主動斷線，PlatformIO 依然會正常提示 **SUCCESS**。

背後運作機制 (若需要客製化)

這套 PIO 的對接機制依賴了專案中的兩項設定：

1. **`platformio.ini` 內的環境變數設定**：利用了 `upload_protocol = custom` 攔截原生的 USB 燒錄行為。
2. **自訂上傳腳本 `scripts/pio_ota.py`**：負責讀取 `.bin`，並打 API 給 `/api/ota/upload`。Token 讀 env `OTA_TOKEN`，沒設就 fallback 到 `admin-token`（向下相容）。

04

策略四：觸發設備向 Config Server 拉取 (Pull from Config Server)

若設備已經正確設定 `Config Server`（例如官方伺服器 `192.168.51.32:3001`），您可以要求設備自行向 Config Server 檢查版本並下載更新，節省本機頻寬。

1. 觸發版本檢查清單

通知設備開始向 Config Server 查詢可用的韌體版本：

```bash
curl -X POST "http://$DEVICE_IP/api/ota/versions" \
  -H "Authorization: Bearer $TOKEN"
```

(此為非同步操作，會回傳 `{"status":"fetching"}`)

2. 獲取版本資訊與下載連結

等待 2~3 秒後，使用 `GET` 查詢結果：

```bash
curl -s "http://$DEVICE_IP/api/ota/versions" \
  -H "Authorization: Bearer $TOKEN" | grep -A 5 -B 5 "versions"
```

此指令會列出支援的版本與對應的 `downloadUrl`（或內部邏輯保留的最新版本資訊）。

3. 指示設備啟動下載並更新

您可以指定特定的下載 URL 與版本號，迫使設備啟動下載任務：

```bash
curl -X POST "http://$DEVICE_IP/api/ota/trigger" \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "url": "http://192.168.51.32:3001/api/firmware/download/mes-gateway-v5.9.9-2(
    "version": "5.9.9"
  }'
```

(若未在 Body 中指定 URL，設備將預設抓取它剛剛查詢到的「最新可用版本」)

設備收到此指令後，會回傳 HTTP 200 `{"ok":true,"message":"OTA download started"}`。
隨後設備會在背景連線至指定的 URL 下載韌體，下載並寫入完成後會自動重啟。

05

特例情境：全新出廠設備實體安裝 (USB Bootstrap)

對於**剛出廠的全新 Opta 設備**，或是無法連網而需要「救磚」的設備，因為裡面尚未包含本專案的 Web Server 功能，故**在此之前提到的網路 OTA (策略一～四) 皆不可用**。

針對這種只拿到 `.bin` 檔案且第一次必須插 USB 線的「開荒」情境，**強烈建議直接使用 Arduino 官方命令列工具 `arduino-cli`**，這比設定 PlatformIO 簡單且穩定許多：

1. 安裝 `arduino-cli`

- **macOS / Linux**: 可使用 Homebrew 快速安裝：

```bash
brew install arduino-cli
```

- **Windows**: 請至 Arduino CLI GitHub 首頁 下載執行檔並加入環境變數。

2. 安裝 Opta 核心通訊支援

初次使用 `arduino-cli`，需要讓它下載並認識 Opta 設備的底層驅動：

```bash
arduino-cli core update-index
arduino-cli core install arduino:mbed_opta
```

3. 接線並尋找連線 Port

將 Opta 設備透過一條能傳輸資料的 USB-C 線接上電腦，接著執行：

```bash
arduino-cli board list
```

在輸出的清單中，找出標示為 Opta 或是 mbed 設備的 Port 號。*(例如 Windows 為 `COM3` 等，macOS/Linux 為 `/dev/cu.usbmodem12201` 這類字眼)*

4. 一行指令燒錄 `.bin`

準備好後，在 `.bin` 檔案所在的資料夾下執行這行指令（請把 `<PORT>` 改成剛才查到的埠號）：

```bash
# 此處的檔案名稱以 mes-gateway.bin 為例
arduino-cli upload -b arduino:mbed_opta:opta -p <PORT> -i mes-gateway.bin
```

這個指令背後會自動幫你：

1. 將原本設備的 USB Port 以 1200 bps 進行觸碰，強迫設備進入 DFU Bootloader 燒錄模式（此時設備 LED 亮起漸暗漸亮）。
2. 背景呼叫 `dfu-util` 將 `.bin` 準確寫進 Flash `0x08040000` 的正確位置。

畫面上看到 `Download done.` 以及 `File downloaded successfully` 字樣即大功告成，設備會自動重啟並套用全新韌體！接著您就可以改用前面的「OTA 策略」，未來都不再需要插 USB 了。

06

驗證更新結果

等待設備重啟（約需 10~15 秒）後，您可以透過 `/api/status` 或 `/api/config` 查詢當前的韌體版本，確認更新是否成功。

```bash
# 查詢設備狀態與版本
curl -s "http://$DEVICE_IP/api/status" \
  -H "Authorization: Bearer $TOKEN"

# 預期會看到類似以下的 JSON 片段：
# {
#   "firmwareVersion": "5.9.9",
#   "uptime": 25,
#   ...
# }
```