電ハニ ローカル API と Home Assistant の説明書

家の WiFi の中から、電ハニとハニカムシェードを動かすための説明です。Home Assistant(以下 HA)などのホームオートメーションや、自分で作るプログラムから使えます。家の WiFi の外(外出先)からは使えません。

電ハニを家の WiFi の中から動かす道は 2 つある。

道	向いている使い方
MQTT(7 章)	Home Assistant(以下 HA)。設定を書かずに、ハニカムシェードが HA の機器として出てくる
HTTP の API(2〜6 章)	HA の RESTful Command、Node-RED、スクリプトなど

1. つなぎ先

項目	値
アドレス	`http://<電ハニの IP>`(ポート 80)か `http://denhani-<MAC の下 4 桁>.local`(例 `denhani-1a2b.local`)
見つけ方	mDNS の `_denhani._tcp`。TXT に `id`(端末 ID)・`fw`(ビルド番号)・`api`(`1`)
暗号化	なし(http。家の WiFi の中だけ)
認証	合言葉(LAN トークン)を決めたときだけ要る。`Authorization: Bearer <合言葉>`。決め方は 1.1
IP	変わると呼べなくなるので決めておく。いちばん確実なのは、ルーターの設定で電ハニの MAC アドレスに番号を予約する(DHCP の予約)こと。アプリの「IP アドレスの固定」を使うときは、ルーターが自動で配る範囲の外の番号にする(範囲の中だと、ほかの機器と重なって電ハニに届かなくなる)

電ハニが 2 台以上あるときは、台ごとにアドレスが違う。ハニカムシェードの番号(`id`)も台ごとに 0 から数える。

1.1 合言葉(LAN トークン)

初めは無い(同じ WiFi の機器なら誰でも呼べる)。アプリの「電ハニ」タブ → 詳しい設定 → 「ローカル API と Home Assistant」で決める(8〜64 文字の英数字と `- _ .`)。決めると `/v1` の全部に要る(無いか違えば

401 `unauthorized`)。ただし `GET /v1/info` だけは合言葉が無くても 200 で `id`・`name`・`model` と `"auth":true,"authorized":false` を返す(アプリがどの台の合言葉を使うか決めるため)。アプリは決めた合言葉を覚えて使う。ほかのスマホのアプリには、同じ画面で同じ合言葉を入れる。

2. 返事の形

- 返事はいつも JSON(CSV の書き出しを除く)。
- 成功: `200`(終わった)か `202`(受け付けた。電波はこのあと出る)。変える物は `{"ok":true, …}`。
- 失敗: `4xx` / `5xx` と

```
{"ok": false, "error": "not_learned", "message": "このハニカムシェードはリモコンを覚えていません"}
```

`error` は短い英字の印で、プログラムはこれを見る。`message` は人に見せる日本語の文(変わることがある)。

HTTP	error	いつ
400	`bad_action` `bad_request` `nothing_queued` `rejected`	指定が正しくない
401	`unauthorized`	合言葉が違う・無い
403	`forbidden_origin`	ブラウザのほかのページから変えようとした(`Origin` か `Sec-Fetch-Site` が付いた POST・DELETE。ファーム 0.5.6〜)
403	`forbidden_host`	`Host` が IP・`.local`・`.lan`・`.home`・`.home.arpa`・`.internal`・ドットの無い名前のどれでもない(DNS の付け替えを防ぐ。0.5.6〜)
404	`not_found`	そのハニカムシェード(または API)が無い
409	`not_learned` `busy`	リモコンを覚えていない / `wait=true` でほかの送信や学習と重なった
503	`queue_full`	待ち行列(32 件)が満杯

3. ハニカムシェードの一覧

```
GET /v1/shades
```

```
{"shades":[
  {"id":0,"name":"リビング南","learned":true,"actions":["up","stop","down","dew"],"alexa":true,
   "last":{"action":"up","source":"mqtt","time":1761300000,"uptime":5400}},
  {"id":1,"name":"ハニカムシェード2","learned":false,"actions":[],"alexa":false,"last":null}
],"max_shades":24}
```

項目	中身
`id`	ハニカムシェードの番号(0〜23)。動かすときに使う。消すと番号は空き、次に足したハニカムシェードが使い回す
`actions`	送れる動作。`up`(上げる)・`stop`(止める)・`down`(下げる)・`dew`(結露防止)
`alexa`	Alexa(スキル「電ハニ」)に出ているか
`last`	**最後に出した指示**。`source` = `lan`(この API・アプリ)/ `cloud`(外出先)/ `alexa` / `automation` / `scene` / `mqtt`。`time` は UNIX 時刻(時計が合っていなければ `null`)。電源を切ると消える(`null`)

ハニカムシェードからの返事は無い(リモコンと同じ一方通行の電波)。今の位置は電ハニにも分からない。分かるのは「最後に何を送ったか」だけ。純正のリモコンで動かした分は分からない。

1 台だけは `GET /v1/shades/{id}`。

4. ハニカムシェードを動かす

```
POST /v1/shades/{id}/command
Content-Type: application/json

{"action": "up"}
```

- 既定は**電ハニの中の待ち行列に積んで 202** を返す(重なっても捨てない。電波は順に出る。1 件 約 0.5 秒)。
- `"wait": true` を付けると、電波を出し終えてから 200 を返す(ほかの送信や学習と重なったら 409 `busy`。そのときは送らない)。
- 本文はフォーム(`action=up`)やクエリ(`?action=up`)でもよい。`u` `s` `d` `h` でもよい。

```
curl -X POST http://denhani-1a2b.local/v1/shades/0/command -H 'Content-Type: application/json' -d '{"action":"up"}'
```

まとめて動かす

```
POST /v1/commands
{"items":[{"shade":0,"action":"up"},{"shade":2,"action":"dew"}]}
```

返事は `202 {"ok":true,"queued":2,"skipped":[]}`。積めなかった物は `skipped` に `{"shade":2,"error":"not_learned","message":"…"}`。1 件も積めなければ 400 `nothing_queued`。フォームなら `items=0:up,2:dew`。

5. 電ハニの情報

```
GET /v1/info
```

```
{"api":1,"id":"dnh-0123456789ab","name":"1
階","model":"DenHaniV4","firmware":"0.5.0","build":"1024d",
 "host":"denhani-1a2b","ip":"192.168.1.50","mac":"01:23:45:67:1A:2B","wifi":
{"ssid":"…","rssi":-52},
 "uptime":5400,"time":1761300000,"shades":3,"max_shades":24,"cloud":true,"mqtt":true,"auth":false}
```


`POST /v1/identify` で LED が 10 秒、緑と青の交互で点滅する(どの台かを目で見つける)。

6. オートメーションとカスタムボタン

電ハニが自分で時刻を見て動かす予定(アプリで作る)。

呼び方	中身
`GET /v1/automations`	予定とカスタムボタンの一覧。場所、今日の日の出・日の入り、各予定が今日動く時刻(`at`、0 時からの分)つき。`items` は `{"s":番号,"a":"u|s|d|h"}`
`POST /v1/automations/{id}/run`	その予定を今すぐ動かす
`POST /v1/scenes/{id}/run`	そのカスタムボタンを動かす(ハニカムシェードごとに決めた動作をまとめて送る)

作る・直す・消すはアプリ専用(`/v1/app/automations` など。仕様書のタグ app)。

7. MQTT と Home Assistant の自動検出

電ハニが家の MQTT ブローカー(HA の Mosquitto など)に自分からつなぐ。HA の MQTT の連携が入っていれば、**学習したハニカムシェードが HA の機器として自動で出てくる**(設定ファイルは書かない)。

7.1 つなぎ方

1. HA に MQTT のブローカー(アドオンの Mosquitto broker)と MQTT の連携を入れておく。電ハニ用のユーザー(HA のユーザーでよい)を作る
2. アプリの「電ハニ」タブ → 詳しい設定 → 「ローカル API と Home Assistant」に、ブローカーの場所(HA の IP)・ポート(1883)・ユーザー名・パスワードを入れて保存
3. 数秒で HA の「設定 → デバイスとサービス → MQTT」に電ハニ(電ハニの名前)が出て、ハニカムシェードが付いてくる

7.2 HA に出る物

電ハニ 1 台が HA の機器 1 つになり、学習したハニカムシェードごとに次の 2 つが付く。

HA の種類	名前	中身
cover(ハニカムシェード)	ハニカムシェードの名前	開ける = 上げる、閉める = 下げる、停止 = 止める。状態は**最後に出した指示**(開いた / 閉じた / 止まった)。位置(%)は無い
button	「<ハニカムシェードの名前> 結露防止」	結露防止を送る

- 電ハニの電源が切れると、HA では「利用不可」になる(LWT)。
- ハニカムシェードを足す・消す・名前を変えると、HA にも出直す。消したハニカムシェードは HA からも消える。
- HA を起動し直したとき(`homeassistant/status` が `online`)は、電ハニが検出の知らせを出し直す。

7.3 トピック(ほかの MQTT の道具から使うとき)

トピック	向き	中身
`denhani/<端末 ID>/availability`	電ハニ →	`online` / `offline`(保持)
`denhani/<端末 ID>/shade/<id>/set`	→ 電ハニ	`up` / `stop` / `down` / `dew`
`denhani/<端末 ID>/shade/<id>/state`	電ハニ →	`open` / `closed` / `stopped`(最後に出した指示。保持)
`homeassistant/cover/<端末 ID>_<id>/config`	電ハニ →	HA の検出の知らせ(保持)
`homeassistant/button/<端末 ID>_<id>_dew/config`	電ハニ →	同上(結露防止)

端末 ID は `GET /v1/info` の `id`(`dnh-` で始まる)。MQTT は暗号化しない(`mqtt://`。家の中のブローカーだけ)。

8. HTTP の API から HA を使う(MQTT を使わないとき)

```
rest_command:
  denhani_1f:
    url: "http://denhani-1a2b.local/v1/shades/{{ shade }}/command"
    method: post
    content_type: "application/json"
    payload: '{"action":"{{ action }}"}'
    # 合言葉を決めたときだけ:
    # headers:
    #   Authorization: "Bearer <合言葉>"

template:
  - cover:
      - name: "リビング南"
        unique_id: denhani_1a2b_0
        device_class: shade
        open_cover:  [ { action: rest_command.denhani_1f, data: { shade: 0, action: up } } ]
        close_cover: [ { action: rest_command.denhani_1f, data: { shade: 0, action: down } } ]
        stop_cover:  [ { action: rest_command.denhani_1f, data: { shade: 0, action: stop } } ]
```

- 電ハニの中で待ち行列に積むので、同じ時刻に別々のオートメーションから呼んでも捨てられない。
- 送れたかは HTTP の番号で分かる(`rest_command` の `response_variable` の `status`。202 が受け付けた)。

9. まだ無いもの

- ハニカムシェードの位置(電波が一方通行なので作れない。分かるのは最後の指示まで)
- HTTPS と MQTT の TLS(家の WiFi の中だけで使う前提)
- 状態の変わり目を HTTP で知らせる道(WebSocket など)。今は MQTT の state で知らせる