

FARMGOD

API Setup & Usage Guide

Table of Contents

Introduction

1. Access & Authentication

- Turning On API Access
- Base URL
- Getting a Token
- Using the Token
- Two-Factor Accounts

2. Core Concepts

- Locations, DeviceSets & Miners
- Managed Users & Effective Scope

3. Common Endpoints

- Miners
- FarmGod Configuration & Curtailment
- DeviceSets
- Managed Users & Billing

4. Conventions & Errors

- Requests & Responses
- Common Error Responses

5. Appendix: Recipes

- Log In & List Your Fleet
- Reboot Every Offline Miner in a DeviceSet
- Pull a Managed User's Billing Stats

INTRODUCTION

The FARMGOD API

Most of what you see in the FARMGOD Interface can also be pulled through this API — current fleet status, per-miner stats, and recent history — alongside every action you can take: changing configuration and triggering actions on miners (reboots, power mode changes, curtailment settings, and so on). If you want to script something instead of clicking it, pull FARMGOD data into your own systems, or automate a workflow, this is the surface to build against. This guide assumes you're comfortable making HTTP requests from a script, a language of your choice, or a tool like `curl`.

THIS ISN'T THE WHOLE API

FARMGOD's API is large — this guide curates the endpoints customers use most often, with enough detail to get real work done, rather than listing every route exhaustively. If you need something not covered here, ask your account contact.

CHAPTER ONE

Access & Authentication

Direct API access is a separate permission from your normal dashboard login, and getting a request successfully authenticated is almost always the only hard part — once you have a valid token, every endpoint in this guide works the same way.

Turning On API Access

On **FARMGOD Cloud**, every account can already call the API — there's nothing to turn on. On a **self-hosted** FARMGOD instance, an account needs API access explicitly enabled before it can authenticate directly against the API; login to the FARMGOD Interface itself is unaffected either way.

API Access CHECKBOX, USERS

On self-hosted instances, a Manager can flip this on for any user from **Users** (main guide, Chapter Four). Managers and the account's primary administrator always have API access and can't have it turned off from here.

Base URL

Every endpoint in this guide is relative to a base URL that depends on how you access FARMGOD:

FARMGOD Cloud — if your dashboard is at `metrics.<yourdomain>`, your API base is `https://api.<yourdomain>`.

Self-hosted — your API base is your dashboard's own host with `/api` appended, e.g. `https://<your-farmgod-host>/api`.

Getting a Token

POST `/login/access-token` FORM-ENCODED

Standard OAuth2 password grant. Send `grant_type=password`, `username` (your account email), and `password` as `application/x-www-form-urlencoded` body fields — not JSON.

EXAMPLE

```
curl -X POST https://api.yourdomain.com/login/access-token \
  -d "grant_type=password&username=you@example.com&password=yourpassword"
```

A successful response looks like:

```
{ "access_token": "eyJhbGciOi...", "token_type": "bearer" }
```

Tokens are JSON Web Tokens and expire **8 hours** after issue. There's no refresh-token endpoint — once a token expires, call `/login/access-token` again to get a new one.

Using the Token

Send the token on every request after login as a standard bearer header:

```
Authorization: Bearer eyJhbGciOi...
```

A missing or malformed header returns `401 Not Authenticated`; a well-formed but expired, invalid, or account-mismatched token returns `403 Could not validate token`.

Two-Factor Accounts

If your account (or the whole instance) has two-factor authentication enabled, `/login/access-token` won't hand back a token on the first call. Instead it returns **HTTP 202** with an `X-2FA-STATUS` header (`2FA_OTP_REQUIRED` or `2FA_SETUP_PENDING`) and an `X-2FA-TOKEN` header identifying that login attempt.

GET /login/2fa-confirm QUERY PARAMS: TOKEN, OTP

Complete the login by sending the `X-2FA-TOKEN` value back as `token`, along with the current one-time code as `otp`. A successful call returns the same `{ access_token, token_type }` shape as a normal login.

SCRIPTING AGAINST A 2FA ACCOUNT

An OTP-in-the-loop login doesn't fit an unattended script well. If you're automating something long-running, ask whoever manages your account about a dedicated account for API use instead of reusing a personal, 2FA-protected login.

CHAPTER TWO

Core Concepts

The API is organized around the same entities as the dashboard — knowing how they nest makes every endpoint in the next chapter self-explanatory.

Locations, DeviceSets & Miners

Location

The top-level container — one physical site, with its own FarmGod configuration (curtailment, pools, alerts, and so on). Most endpoints take a `location_id` to scope the request.

DeviceSet

A rack, sub-site, or logical grouping of miners within a Location, identified by a `deviceset_id`. Batch actions are typically run against a DeviceSet's miners.

Miner

One physical ASIC, identified by a `device_id`, belonging to exactly one DeviceSet.

Managed Users & Effective Scope

If your account hosts other customers (the Managed Users model covered in the main guide), your token authenticates as *you*, but many endpoints — including the ones that read DeviceSets and miners — operate on whichever account owns the data being requested, scoped the same way it's scoped in the dashboard: by which DeviceSets that account owns. You don't need a separate token per managed user for read access to your own fleet.

Common Endpoints

Grouped the way the dashboard itself is organized. Every example below assumes the `Authorization: Bearer <token>` header from Chapter One on every request.

Miners

POST /miner/table-page

The same query the miner tables in the dashboard use — search, filter, sort, and paginate across your fleet. Body fields include `profile` (which table view — e.g. `asics_online`, `offline`, `all`), `location_id` or `device_set_ids` to scope it, `page` / `page_size`, `search`, `sorting`, and `column_filters`. Returns `{ rows, total, page, page_size }`.

POST /farm/{deviceset_id}/batch

Runs an action against multiple miners in one DeviceSet. Body: `{ "action": "reboot", "device_ids": ["...", "..."] }`. `action` is the same lowercase action name the dashboard's Actions menu uses (`reboot`, `power_on`, `power_off`, `sleep`, and so on); some actions take a value via `action_p` — a power mode change, for instance, passes the target mode there. For `power_off` / `power_on` specifically, don't also set `is_curtailed` — Kollektor's executor plugin flips it automatically (1/0) once the action is confirmed against the physical miner, never before; see the Curtailment Guide's "Custom API-Driven Curtailment" example for the full mechanism.

FarmGod Configuration & Curtailment

GET /farmgod/?location_id={id}

Returns the full FarmGod configuration for a location — every field from the Settings tabs covered in the main guide, curtailment included (`curtail_enabled`, `curtail_mode`, `curtail_scope`, and the rest, using the same field names as this guide's Curtailment Guide describes).

POST /farmgod/?location_id={id}

Updates that same configuration — send only the fields you want to change. This is the endpoint the dashboard's Save Configuration button calls; the same confirmation-worthy transitions covered in the Curtailment Guide (disabling curtailment while miners are curtailed, for one) apply here too, since the safety logic lives server-side, not just in the dashboard's UI.

GET /curtailment/price/{location}

The live grid price FARMGOD's own curtailment engine is currently using for that location's ERCOT hub or load zone.

DeviceSets

GET /farm

Lists the DeviceSets you're allowed to see.

POST /farm

Creates a new DeviceSet — name, Location, electricity price/currency, an IP range to scan, and miner admin credentials, the same fields as the dashboard's Create Set flow.

DELETE /farm/{deviceset_id}

Deletes a DeviceSet and every miner inside it — same destructive scope as "Delete Set" in the dashboard, not "Delete Miners." There's no separate API call for the miners-only version; remove miners individually first if that's what you need.

Managed Users & Billing

GET /users/managed

Lists your managed (customer) accounts — the hosting-provider model from the main guide.

GET /users/managed/{user_id}/stats

Energy consumed, uptime, and cost for one managed user over a period — the same numbers behind the magnifying-glass usage dashboard in Users.

GET / PUT /users/managed/{user_id}/billing-profile

Reads or sets that user's billing markup (flat cents-per-kWh or percentage) — the same setting used when the dashboard generates a report or invoice for them.

Conventions & Errors

Requests & Responses

Every request and response body is JSON, except the login endpoints, which are form-encoded on the way in (standard OAuth2 password grant) and JSON on the way out. There's no API versioning in the URL path — endpoints are stable at the paths shown in this guide.

Common Error Responses

Status	Meaning	Typical cause
401	Not authenticated	Missing or malformed <code>Authorization</code> header
403	Could not validate token / Access Denied	Expired or invalid token, or the account lacks permission for that action
403	API access is disabled for this account	Self-hosted account without the API Access flag turned on — see Chapter One
404	Not found	A DeviceSet, miner, or user ID that doesn't exist or isn't yours
422	Validation error	Request body failed schema validation — the response's <code>detail</code> is a list of per-field errors instead of a single string

Every error response has a JSON body with a `detail` field — either a plain string, or, for validation errors, a list of objects each naming the offending field and what was wrong with it.

APPENDIX

Recipes

Four worked examples using `curl`, covering login through to real, common tasks. Swap in your own base URL, credentials, and IDs.

Log In & List Your Fleet

```
TOKEN=$(curl -s -X POST https://api.yourdomain.com/login/access-token \
-d "grant_type=password&username=you@example.com&password=yourpassword" \
| jq -r .access_token)

curl -s https://api.yourdomain.com/farm \
-H "Authorization: Bearer $TOKEN"
```

Reboot Every Offline Miner in a DeviceSet

```
curl -s -X POST https://api.yourdomain.com/miner/table-page \
-H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" \
-d '{"profile":"offline","device_set_ids":
["YOUR_DEVICESET_ID"],"fetch_mode":"all"}' \
| jq -r '.rows[].DEVID' > offline_ids.txt

curl -s -X POST https://api.yourdomain.com/farm/YOUR_DEVICESET_ID/batch \
-H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" \
-d '{"\action\":"reboot","\device_ids\":"$(jq -R -s -c 'split("\n")
[: -1]' offline_ids.txt)}'
```

Pull a Managed User's Billing Stats

```
curl -s "https://api.yourdomain.com/users/managed/USER_ID/stats?
period=mtd" \
-H "Authorization: Bearer $TOKEN"
```



FARMGOD LLC
1017 W. San Antonio St., Suite 101
Lockhart, TX 78644
farmgod.io