Fraimic REST API Guide

Control and monitor your Fraimic from your smart home

Fraimic Engineering

Version 0.2.16 — March 2026

Contents

At a Glance . . . 3
Getting Started . . . 4
- Finding Your Frame's Address . . . 4
- Making Requests . . . 4
- A Quick Test . . . 4

The Endpoints . . . 6
- Get Device Info . . . 6
- Get Battery Status . . . 7
- Restart Device . . . 7
- Enter Deep Sleep . . . 8
- Refresh Display . . . 8
- Upload Image . . . 8

Home Assistant Integration . . . 10
- Step 1: Add REST Sensors . . . 10
- Step 2: Add Shell Commands . . . 10
- Step 3: Reload — No Restart Required . . . 11
- Step 4: Create Scripts for Dashboard Buttons . . . 11
- Step 5: Add Dashboard Buttons . . . 12

Automation Ideas . . . 13
- Low Battery Alert . . . 13
- Refresh Display at Sunrise . . . 13
- Rotate Artwork on a Schedule . . . 13

Troubleshooting . . . 15
- "Connection refused" or "No route to host" . . . 15
- "Empty response" or timeout . . . 15
- Image upload returns `invalid_image_size` . . . 15
- Home Assistant sensors show "unavailable" . . . 15
- Shell commands don't seem to work . . . 16

Tips & Notes . . . 17

At a Glance

Your Fraimic art frame has a built-in REST API — a set of web endpoints that let you control the frame and check its status from other devices on your local network. You can monitor battery level, trigger a display refresh, upload artwork, or put the frame to sleep — all from your phone, computer, or smart home system like Home Assistant.

No internet required for local communication. No account needed. Everything runs on your WiFi network between your devices and the frame.

What you can do with the API:

- **Monitor your frame** — check battery, WiFi signal, firmware version, and display status
- **Control your frame** — restart it, put it to sleep, or refresh the display
- **Display custom artwork** — upload `.bin` image files directly to the frame
- **Integrate with Home Assistant** — add Fraimic sensors and controls to your smart home dashboard

The API is designed for Home Assistant users, tinkerers, and anyone who wants to automate their Fraimic beyond the normal tap-and-speak experience. If you're just using the frame normally, you don't need this guide — everything works through the built-in portal and voice recording.

Getting Started

Finding Your Frame's Address

Every API request starts with your frame's network address. There are two ways to reach it:

Method	Address	When to Use
mDNS	`http://fraimic.local`	Preferred — works on most home networks
IP address	`http://192.168.1.XX`	Fallback if `fraimic.local` doesn't resolve

You can find the frame's IP address on the portal at `fraimic.local/info`, or in your router's DHCP client table.

Important: Your frame must be awake and connected to WiFi. When it's in deep sleep, it's completely unreachable — tap the frame to wake it first.

Making Requests

The API uses standard HTTP. You can use any tool that speaks HTTP — `curl` from the command line, a web browser, Home Assistant's REST integration, or any programming language.

- **GET** requests retrieve information (battery status, device info)
- **POST** requests trigger actions (restart, sleep, refresh, upload)

No authentication is required. Anyone on your local network can access these endpoints.

A Quick Test

Open your browser and navigate to:

`http://fraimic.local/api/info`

You should see a JSON response with your frame's current status. If that works, you're ready to use any of the endpoints below.



Figure 1: Browser showing the JSON response from /api/info

The Endpoints

Get Device Info

Returns a complete snapshot of your frame's current state — firmware version, WiFi connection details, battery status, registration, and display timing.

`GET /api/info`

Try it in your browser or with curl:

```
curl http://fraimic.local/api/info
```

What you get back:


```
{
  "firmware_version": "0.2.16",
  "wifi": {
    "connected": true,
    "ssid": "MyNetwork",
    "rssi": -52,
    "channel": 6,
    "ip": "192.168.1.42"
  },
  "battery": {
    "percent": 78,
    "voltage_mv": 3850,
    "charging": false,
    "cable_connected": false,
    "source": "fuel_gauge"
  },
  "device": {
    "registered": true,
    "time_synced": true,
    "uptime_s": 3600
  },
  "settings": {
    "voice_recording": true,
    "keep_awake": false
  },
  "display": {
    "last_refresh": "2026-03-18T10:30:00",
    "next_refresh": "2026-03-21T03:00:00"
  }
}
```

Key fields explained:

Field	What It Tells You
`wifi.rssi`	Signal strength in dBm. -30 to -50 is excellent, -50 to -60 is good, below -70 is weak.
`battery.percent`	Current charge level.
`battery.source`	`"fuel_gauge"` (hardware sensor, accurate) or `"adc"` (voltage estimate, less precise).
`device.registered`	Whether the frame is paired with a Fraimic account.
`device.uptime_s`	Seconds since last boot — useful for knowing if the frame recently restarted.
`display.next_refresh`	When the next scheduled display refresh will happen.

Get Battery Status

A lightweight version of `/api/info` that returns only battery data. Use this when you want to poll frequently without requesting the full device snapshot.

```
GET /api/battery
```

```
curl http://fraimic.local/api/battery
```

Response:

```
{
  "percent": 78,
  "voltage_mv": 3850,
  "charging": false,
  "cable_connected": false,
  "source": "fuel_gauge"
}
```

Polling recommendation: Every 60 seconds is fine for battery monitoring. For general status checks, `/api/info` every 5 minutes strikes a good balance between freshness and not overloading a battery-powered device.

Restart Device

Reboots the frame. Logs are flushed and statistics saved before the restart — nothing is lost.

```
POST /api/restart
```

```
curl -X POST http://fraimic.local/api/restart
```

Response:

```
{"status": "restarting"}
```

The frame will be unreachable for about 10–15 seconds while it reboots and reconnects to WiFi.

Enter Deep Sleep

Puts the frame into deep sleep to conserve battery. The frame becomes completely unreachable until tapped awake or woken by its internal timer.

```
POST /api/sleep
```

```
curl -X POST http://fraimic.local/api/sleep
```

Response (success):

```
{"status": "sleeping"}
```

Response (blocked):

```
{"error": "charging_cable_connected"}
```

Deep sleep is blocked when a charging cable is connected — this is intentional. The frame stays awake while charging so you can access the portal and it can check for firmware updates.

Refresh Display

Triggers a full E-Ink display refresh cycle. The refresh runs in the background and takes approximately 20–30 seconds to complete.

```
POST /api/refresh
```

```
curl -X POST http://fraimic.local/api/refresh
```

Response:

```
{"status": "refresh_started"}
```

Note: This re-renders the current image on the E-Ink display. It does not download a new image from the server — it refreshes what's already shown. Useful for clearing ghosting artifacts that build up over time on E-Ink screens.

Upload Image

Upload a `.bin` image file to display on the frame. The image renders automatically after a successful upload — no separate refresh step needed.

```
POST /api/image
Content-Type: application/octet-stream
```

The file must be exactly **960,000 bytes** — a 1200x1600 grayscale image at 4 bits per pixel.

```
curl -X POST \
```

```
  -H "Content-Type: application/octet-stream" \
  --data-binary @my_artwork.bin \
  http://fraimic.local/api/image
```

Response (success):

```
{"status": "rendering", "bytes_received": 960000}
```

If something goes wrong:

HTTP Status	Error	What Happened
400	`invalid_image_size`	File isn't exactly 960,000 bytes
400	`file_too_large`	File exceeds 1 MB
501	`unsupported_content_type`	Sent a JPEG or PNG — only `.bin` format is supported currently
503	`buffer_not_ready`	Frame is busy (e.g., mid-refresh). Wait a few seconds and try again.

About the `.bin` format: This is a raw 4-bit grayscale image where each byte holds two pixels (high nibble first, low nibble second). Conversion tools for JPEG/PNG to `.bin` are available in the Fraimic firmware repository's `tools/` directory. Native JPEG and PNG upload support is planned for a future firmware update.

Home Assistant Integration

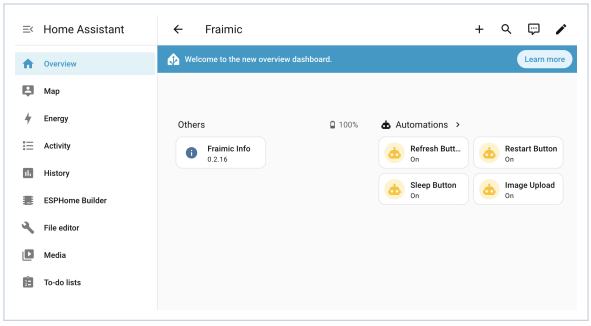


Figure 2: Home Assistant dashboard with Fraimic battery sensor and control buttons

This section walks you through adding Fraimic sensors and controls to your Home Assistant dashboard. By the end, you'll have battery monitoring, WiFi signal tracking, and one-tap buttons for restart, sleep, refresh, and image upload.

Step 1: Add REST Sensors

Open your Home Assistant `configuration.yaml` file and add the following block. This creates sensors that poll your frame every 5 minutes:

```yaml
rest:
  - resource: http://fraimic.local/api/info
    scan_interval: 300
    sensor:
      - name: "Fraimic Battery"
        value_template: "{{ value_json.battery.percent }}"
        unit_of_measurement: "%"
        device_class: battery
      - name: "Fraimic WiFi Signal"
        value_template: "{{ value_json.wifi.rssi }}"
        unit_of_measurement: "dBm"
        device_class: signal_strength
      - name: "Fraimic Firmware"
        value_template: "{{ value_json.firmware_version }}"
      - name: "Fraimic Charging"
        value_template: "{{ value_json.battery.charging }}"
      - name: "Fraimic IP"
        value_template: "{{ value_json.wifi.ip }}"
```

Step 2: Add Shell Commands

These give Home Assistant the ability to send action commands to your frame:

```
shell_command:
  fraimic_restart: >-
    curl -s -X POST http://fraimic.local/api/restart
  fraimic_sleep: >-
    curl -s -X POST http://fraimic.local/api/sleep
  fraimic_refresh: >-
    curl -s -X POST http://fraimic.local/api/refresh
  fraimic_upload: >-
    curl -s -X POST
    -H "Content-Type: application/octet-stream"
    --data-binary @/config/www/fraimic/{{ filename }}
    http://fraimic.local/api/image
```

Note on the upload command: Place your `.bin` image files in `/config/www/fraimic/` on your Home Assistant server. The `{{ filename }}` variable is passed when you call the command from a script or automation.

Step 3: Reload — No Restart Required

After saving `configuration.yaml`, go to **Developer Tools > YAML** in Home Assistant and reload:

- **REST entities**
- **Shell Commands**

A full Home Assistant restart is not needed. Your new sensors and commands will appear immediately.

Step 4: Create Scripts for Dashboard Buttons

Go to **Settings > Automations & Scenes > Scripts > Add Script**, or add directly to your `scripts.yaml`:

```
fraimic_restart:
  alias: "Fraimic Restart"
  icon: mdi:restart
  sequence:
    - service: shell_command.fraimic_restart

fraimic_sleep:
  alias: "Fraimic Sleep"
  icon: mdi:sleep
  sequence:
    - service: shell_command.fraimic_sleep

fraimic_refresh_display:
  alias: "Fraimic Refresh Display"
  icon: mdi:monitor
  sequence:
    - service: shell_command.fraimic_refresh
```

```yaml
fraimic_display_dinosaur:
  alias: "Display Dinosaur"
  icon: mdi:dinosaur
  sequence:
    - service: shell_command.fraimic_upload
      data:
        filename: dinosaur.bin
```

Step 5: Add Dashboard Buttons

Add these button cards to your Fraimic area dashboard for one-tap control:

```yaml
type: horizontal-stack
cards:
  - type: button
    name: Restart
    icon: mdi:restart
    tap_action:
      action: perform-action
      perform_action: script.fraimic_restart
  - type: button
    name: Sleep
    icon: mdi:sleep
    tap_action:
      action: perform-action
      perform_action: script.fraimic_sleep
  - type: button
    name: Refresh
    icon: mdi:monitor
    tap_action:
      action: perform-action
      perform_action: script.fraimic_refresh_display
  - type: button
    name: Dinosaur
    icon: mdi:dinosaur
    tap_action:
      action: perform-action
      perform_action: script.fraimic_display_dinosaur
```

Automation Ideas

Home Assistant's real power is automation — things that happen automatically based on conditions. Here are some ideas to get you started.

Low Battery Alert

Get a push notification on your phone when the frame drops below 20%:

```yaml
automation:
  - alias: "Fraimic Low Battery Alert"
    trigger:
      - platform: numeric_state
        entity_id: sensor.fraimic_battery
        below: 20
    action:
      - service: notify.mobile_app_your_phone
        data:
          title: "Fraimic Battery Low"
          message: >-
            Battery at {{ states('sensor.fraimic_battery') }}%.
            Time to connect the charger.
```

Refresh Display at Sunrise

Refresh the E-Ink display each morning to keep it looking crisp:

```yaml
automation:
  - alias: "Fraimic Sunrise Refresh"
    trigger:
      - platform: sun
        event: sunrise
    action:
      - service: shell_command.fraimic_refresh
```

Rotate Artwork on a Schedule

Upload a different image each day of the week. Place the `.bin` files in `/config/www/fraimic/` named by day:

```yaml
automation:
  - alias: "Fraimic Weekly Art Rotation"
    trigger:
      - platform: time
        at: "08:00:00"
    action:
```

```
- service: shell_command.fraimic_upload
  data:
    filename: >-
      {{ ["monday.bin", "tuesday.bin", "wednesday.bin",
          "thursday.bin", "friday.bin", "saturday.bin",
          "sunday.bin"][now().weekday()] }}
```

Troubleshooting

"Connection refused" or "No route to host"

What it means: Your device can't reach the frame over the network.

Check these things:

1. **Is the frame awake?** If it's in deep sleep, it won't respond to network requests. Tap the frame to wake it.
2. **Same network?** Your device and the frame must be on the same WiFi network.
3. **mDNS working?** Try the IP address directly instead of `fraimic.local`. Not all routers support mDNS resolution. Find the IP at `fraimic.local/info` or in your router's DHCP table.
4. **Guest network isolation?** If your device is on a guest network, it may be blocked from reaching devices on the main network.

"Empty response" or timeout

What it means: The frame is reachable but didn't respond in time.

What to do:

- The frame may be busy (mid-recording, refreshing the display, or downloading an image). Wait 30 seconds and try again.
- If it persists, restart the frame via the portal (`fraimic.local/action?mode=restart`) or by power cycling it.

Image upload returns `invalid_image_size`

What it means: The uploaded file isn't exactly 960,000 bytes.

Common causes:

- **Wrong file format** — you may have sent a JPEG or PNG instead of a `.bin` file
- **Truncated transfer** — the file didn't upload completely
- **Wrong resolution** — the `.bin` must be generated from a 1200x1600 image at 4 bits per pixel

---

Home Assistant sensors show "unavailable"

What it means: Home Assistant can't reach the frame to poll its status.

Check these things:

1. **Frame is awake and connected** — visit `fraimic.local/info` from your browser to confirm

2. **mDNS resolution** — Home Assistant may not resolve `fraimic.local`. Try replacing it with the frame's IP address in your `configuration.yaml`
3. **Network segmentation** — if Home Assistant runs in Docker or on a separate VLAN, it may not be able to reach the frame's subnet. Check your network routing.
4. **Frame in deep sleep** — sensors will show "unavailable" when the frame is sleeping. This is normal behavior — they'll update automatically when the frame wakes up.

Shell commands don't seem to work

What to do:

1. **Test the curl command directly** — SSH into your Home Assistant server and run the curl command manually to see if you get a response
2. **Check Home Assistant logs** — go to **Settings > System > Logs** and look for errors related to shell commands
3. **Verify curl is available** — it's included by default in Home Assistant OS, but custom installations may need it installed separately

Tips & Notes

- **Deep sleep awareness:** When the frame is in deep sleep, it's completely unreachable over the network. The frame wakes on tap or on its internal timer. Plan automations around this — for example, don't try to upload an image at 3 AM when the frame is likely sleeping.
- **Polling frequency:** Use `/api/info` for general monitoring (every 5 minutes is a good balance). Use `/api/battery` for more frequent battery checks (every 60 seconds). Avoid polling more aggressively than that — the frame is battery-powered and each request costs a small amount of energy.
- **mDNS:** `fraimic.local` resolves via mDNS (multicast DNS). Most home networks support it out of the box. If yours doesn't, use the frame's IP address instead. The IP is visible on the portal at `fraimic.local/info` or in your router's DHCP client table.
- **Image format:** The `.bin` format is a raw 4-bit grayscale image at 1200x1600 resolution. JPEG and PNG upload support is planned for a future firmware update. In the meantime, conversion tools are available in the firmware repository's `tools/` directory.
- **Multiple frames:** If you have more than one Fraimic, each has its own IP address. Use IP addresses instead of `fraimic.local` to distinguish between them in your Home Assistant configuration.

This guide covers Fraimic firmware v0.2.16. For the latest version, check with your Fraimic contact.