API

The Hukamnama API

7 min read · Updated 19 November 2026

The public API for the daily Hukamnama from Sri Harmandir Sahib: its routes, the record, checking the text against its hash, pending days, caching, errors, and how its spec is generated.

https://docs.gurbanisoul.com/api/hukamnama/

A second, separate API: the daily Hukamnama from Sri Harmandir Sahib, Amritsar, as JSON, at `https://api.gurbanisoul.com`. It is free, needs no key and no account, and may be called from a server or straight from a browser. It is not the Knowledge Base API described on the other pages of this section: it has its own host, its own version and **its own reference**.

Live since 1.7.0 (17 November 2026), when the Hukamnama launched on gurbanisoul.com. Reads are rate-limited per address at the edge (120 a minute); `/health` and device registration pass through to the service.

Each day is located by Ang from the Hukamnama SGPC publishes. Every line is then read from the verbatim Sri Guru Granth Sahib Ji corpus and proved against a hash before it is served. SGPC's page only finds the shabad; no character is taken from it. Gurbani Soul is not affiliated with SGPC.

A first request

```
curl https://api.gurbanisoul.com/v1/hukamnama/today
```

"Today" is the date in Amritsar (UTC+05:30), wherever the request comes from. The Hukamnama usually appears after about 05:30 in the morning there. Before that the answer is a **pending** one.

The routes

Request	What it returns
`GET /v1/hukamnama/today`	The Hukamnama for the current date in Amritsar, or a pending answer.
`GET /v1/hukamnama/YYYY-MM-DD`	One day, by its date in Amritsar, from 2026-11-01 up to today there.

Request	What it returns
`GET /v1/hukamnama?limit=7`	The latest published days, newest first: 1 to 30, 7 by default. `include=lines` adds every line.
`GET /v1/hukamnama/index`	Every published day, newest first: date, Ang, raag, author and hash, without the text.
`GET /v1/status`	Whether today's Hukamnama is published yet, and the latest published date.

Parameters, every field and every response are in **the reference**.

The record

A published day carries the whole shabad, line by line.

Field	Meaning
`amritsar_date`	The day this Hukamnama belongs to, in Amritsar. Show this date, never the reader's.
`status`	`published`, or `pending` on an answer with no lines.
`ang`, `ang_end`	The Ang the shabad opens on and the Ang it ends on.
`citation`	The citation to show with the text: "Sri Guru Granth Sahib Ji · Ang N".
`lines`	Every line in order: `gurmukhi` (verbatim), `translit` (a rule-based reading aid), `is_heading`, `is_rahao`, `author`.
`rahao_gurmukhi`	Optional. The shabad's ਰਹਾਉ couplet, verbatim, in order (the same lines that carry `is_rahao` and the verse just before it). Present only when the shabad has exactly one ਰਹਾਉ ; nothing is guessed. A convenience for small surfaces such as widgets.
`author`, `raag`	The day's author and raag. When the verses have more than one author, as in a Vaar, `author` is `null` and each line carries its own.
`range_sha256`	The hash that proves the lines (below).
`scripture_sha256`	The fingerprint of the edition of the text the lines were read from.
`published_at`	When the day was published here, in UTC.
`corrected_at`	Present only when the day's range was corrected after it was published: when, in UTC. `published_at` stays the first publication.
`url`, `attribution`, `terms_url`	The day's page, the credit to show with its link, and the terms.
`lines[].en`, `translation`	Dr. Sant Singh Khalsa's English translation, where it is included: a separate, labelled layer, never the Gurbani.

Check the text yourself

Join the `gurmukhi` of every line with a newline, take the SHA-256 of the UTF-8 bytes, and compare it with `range_sha256`. If they differ, show nothing: late is better than wrong.

```
import hashlib
import json
import urllib.request

with urllib.request.urlopen("https://api.gurbanisoul.com/v1/hukamnama/today") as res:
    day = json.load(res)

if day["status"] == "published":
    text = "\n".join(line["gurmukhi"] for line in day["lines"])
    assert hashlib.sha256(text.encode("utf-8")).hexdigest() == day["range_sha256"]
```

The service applies the same check before it writes a day, the website applies it again before it shows one, and the app applies it before it renders one.

A pending day

When no Hukamnama is published for the day asked for, the answer is still `200`, with `"status": "pending"`, no lines, `expected_by` (05:30 in Amritsar on that day, in UTC) and `latest`, the most recent published date. This is the answer before the morning's publication, and for a day that was held back because it could not be proved. A day is never served under another day's date: to show the most recent Hukamnama, ask for `latest` by its own date and show that date.

A corrected day

If a published day turns out to be the wrong shabad, its range is replaced. The record then carries `corrected_at`, and its `range_sha256` and lines are the corrected ones; `published_at` stays the time of the first publication. The index carries `corrected_at` for the day too, and the website's feeds show the day as an updated entry. A client that keeps days should compare `range_sha256` (or send `If-None-Match`) rather than assume a published day never changes. The API's answers are not cached at the edge, so a correction is visible at once; a browser may keep a past day for its `max-age` (one hour). The website's cached pages are cleared by the service after the correction.

When the service is paused

The owner can pause the service without a deploy. While it is paused every day reads as `pending`, the list and the index are empty, and `/v1/status` says `"paused": true`. Nothing is served until it is resumed; then everything returns as it was.

Caching

Every `200` carries a strong `ETag`. Send it back in `If-None-Match` and an unchanged answer is a `304` with no body.

Answer	`Cache-Control`
today, published	`public, max-age=60`
a past day	`public, max-age=3600, s-maxage=86400`
pending, status	`public, max-age=30`
the list, the index	`public, max-age=60`

Once a minute is as often as anyone needs to ask.

Errors

An error is JSON with `error` (a few words), `code` and `request_id`.

Status	`code`	When
400	`invalid_request`	the date is not a real YYYY-MM-DD date, or `limit` is not 1 to 30
404	`not_found`	a date after today in Amritsar or before 2026-11-01, or an unknown path
405	`method_not_allowed`	anything but `GET`, `HEAD` or `OPTIONS`
429	`rate_limited`	too many requests from one address; wait for `Retry-After`
503	`unavailable`	the API cannot read its files; nothing is served rather than something unproved

What you can rely on

- **Version 1 only grows.** Fields may be added; none is renamed or removed without twelve months' notice. The keys the shipped iOS app decodes are pinned by a test.
- **No guarantee of availability.** The API is run with care, on a best-effort basis. When a day cannot be proved it is held back.
- **Open to any origin.** Reads carry `Access-Control-Allow-Origin: *`.

Showing and crediting it

The rules and the terms are on **the Developers page**[1]: show the Gurmukhi exactly as it arrives and whole, name it "Hukamnama · Sri Harmandir Sahib" with its Amritsar date and citation, keep any transliteration or translation apart under its own label, never call it official, and credit it with the words and link in `attribution`. The same page has an embed snippet that needs no code, and the feeds (`https://gurbanisoul.com/feeds/hukamnama.xml` and `.json`).

How the spec is made

`contract/hukamnama.openapi.json` is generated, not written. The API is two programs: the Hukamnama service writes one file per published day (`services/hukamnama`, `public.py`) and an edge Worker serves them (`services/hukamnama-edge`, `core.js`). `tools/gen_hukamnama_openapi.py` runs both: it publishes three sample days from the pinned corpus (one of them with two authors), lets the service write its files once without and once with the English layer, sends sample requests through the Worker's real handler, including an empty archive and the error cases, and infers every schema from the answers. So the spec cannot describe a field that is not served.

```
python3 tools/gen_hukamnama_openapi.py            # regenerate (needs db/sggs.sqlite and
node)
python3 tools/gen_hukamnama_openapi.py --check    # exit 1 on any difference
```

What is written by hand is one sentence per route and one per field; a served field without a sentence fails the generator. The Hukamnama service's tests (`tests/test_openapi.py`, run by `web-ci · python`) fail when the committed spec is stale, when a key the iOS app decodes stops being required, when the spec holds any Gurmukhi, or when the Developers page describes a field the API does not serve.

Links

1. https://gurbanisoul.com/en/developers

Sri Guru Granth Sahib Ji — Knowledge Base wiki. Generated from commit 7783889. Updated 19 November 2026. This copy may be out of date; the page is at https://docs.gurbanisoul.com/api/hukamnama/