Reference · Translating the site

Translating gurbanisoul.com

3 min read · Updated 4 November 2026

How the marketing site is translated into Punjabi and French: the message catalogs, the checks, the human review gate and how a language is switched on.

https://docs.gurbanisoul.com/website/translation/

The marketing site (home, Features, The watch, Learn, Support, Privacy, Contribute) is built once per **enabled** language under `/en`, `/pa` and `/fr`. English is the source. The Knowledge Base, the app and the scripture are not translated by this process.

The rules that never bend

- **Scripture is never translated here.** Gurbani on any page comes from the corpus, verbatim and cited by Ang. A message file never contains Gurbani; the `NoGurbaniInTranslations` gate checks every translated string against the corpus.
- **A language ships whole or not at all.** It is switched on only when every string is translated, current and signed off by a fluent reviewer. There is no partly-French page.
- **English governs legal text.** The translated Privacy and Contribute pages say so, with the date of the English version.
- **Honest about English.** The app, the Knowledge Base, the newsletter, Stripe checkout and its receipts are English; translated pages say so where it matters.

Where the words live

File	What
`frontend/src/i18n/messages/en.json`	The source: every interface string, as ICU messages (`{count}`, `{x, select, …}`, `<link1>…</link1>`).
`frontend/src/i18n/messages/<lang>.json`	A translation, key for key.
`frontend/src/i18n/messages/<lang>.lock.json`	Per key, a hash of the English it was translated from, so a later English edit marks it **stale**.
`frontend/src/i18n/reviews/<lang>.json`	The review record: `reviewer`, `date`, `catalogSha256` (the exact catalog signed off), `notes`.
`frontend/src/i18n/locales.json`	The languages and the `enabled` list.

A message keeps its placeholders and tags exactly: `{angs}` stays `{angs}`, and `<link1>…</link1>` wraps the words that are the link, wherever the sentence puts them. Typographic rules (French spacing before `; : ! ?`, Gurmukhi punctuation) follow the

style guide; words follow the glossary.

The workflow

1. **Draft.** The draft catalog is written from the glossary and the style guide, then `node frontend/scripts/i18n-status.mjs --lock <lang>` records which English each key was translated from.
2. **Review sheet.** `node frontend/scripts/i18n-status.mjs --sheet <lang> sheet.html` writes every key with its English, the translation and its relative length.
3. **Review on the rendered pages.** A preview build with the language enabled is put on staging; the reviewer reads every page at phone and desktop width, not only the sheet.
4. **Sign-off.** The reviewer's corrections are applied, then the review record is written with the sha256 of the final catalog (`shasum -a 256 frontend/src/i18n/messages/<lang>.json`). Privacy and Contribute also need the legal review (gate G4).
5. **Switch on.** The language is added to `enabled` in `locales.json`, and `node frontend/scripts/gen-i18n-redirects.mjs` rewrites the redirects.

`node frontend/scripts/i18n-status.mjs` reports every language (missing, stale, extra, placeholder mismatches, review) and **fails CI** when an enabled language is not ready, or when any catalog is broken. Any English edit after sign-off marks keys stale and blocks the next release until they are re-translated and re-reviewed.

The checks behind it

- **Pseudo-locale build** (`npm run build:pseudo`) and `scripts/ci/check_i18n_coverage.py`: any interface English typed into a page instead of the catalog fails CI.
- **`LanguageRouting`, `SeoInvariants`**: every page declares its language, links within it, and names its language versions reciprocally (hreflang).
- **`test-i18n`**: every message parses; placeholders and tags render; values are never markup.
