

The THIRI manual, public edition

What THIRI is, how it reasons about harmony, and what we measured. Chapters for signed-in developers and Builders are on build.thiri.ai/manual.

Contents

1. What THIRI is (*Public*)
2. The five calls (*Public*)
3. MCP and agents: putting the engine in the loop (*Public*)
4. On the pedal: fitting the engine in 1.333 ms (*Public*)
5. Google Lyria, Magenta and Gemini: what THIRI can steer (*Public*)
6. Composer C: an agent that writes Csound with THIRI holding the harmony (*Public*)
7. How we measure (*Public*)
8. Glossary (*Public*)
9. Questions developers ask (*Public*)
10. Appendix: licences and hand-off rules (*Public*)

Part I • The engine

oo. What THIRI is

A harmony engine that computes rather than guesses, and the one sentence that explains why that matters to anyone putting music in front of a language model.

THIRI stands for The Harmonic Intelligence Reference Infrastructure. Older documents expand it differently; this is the locked form. In practice it is one thing: a harmony engine you can ask about chords, keys, voicings and progressions, and which answers by computing from pitch-class sets and a fixed grid of rules. It does not sample. Ask it the same question a hundred times and you get the same right answer a hundred times.

The problem it solves

Language models are good at intent, planning and taste. They are bad at arithmetic over pitch classes. Ask a model to voice a C13 for jazz piano and it will often produce something plausible and wrong, because it is predicting tokens, not spelling intervals. The error is invisible to a non-musician and obvious to anyone who plays.

THIRI sits beside the model as a tool. The model keeps the conversation; THIRI supplies the notes.

What you get

Five calls, over REST or as MCP tools:

CALL	QUESTION IT ANSWERS
<code>analyze</code>	What is this chord, and what is it doing in this key?
<code>resolve</code>	Which notes, MIDI numbers and frequencies is it?
<code>voicing</code>	How should an instrument play it after the previous chord?
<code>reharmonize</code>	What else could this progression be, and why does each option work?
<code>conduct</code>	Turn a plain-English direction into a band arrangement and a MIDI file.

Every answer is a JSON object with the theory attached: roman numerals, function, the scales that fit, the guide tones that moved.

Where it runs

- On the edge, at `chords.thiri.ai`, metered by API key.
- Inside Claude, Cursor and any MCP client, through `@bluesprincemedia/thiri-mcp`.
- In the browser, as the engine behind the instruments on this site.
- In C++, as `thiri_core` inside the THIRI VST and on the Elk Stomp pedal, pinned to the JavaScript engine by thousands of golden test vectors.

The rest of this manual walks through each of those, what was measured, and what is still open.

SOURCES

- [production/thiri-build-site/src/pages/docs/api.astro](#) (live responses, 2026-10-02)

01. The five calls

What each endpoint answers, what it needs, and what it returns, with the real responses captured from the live engine.

Everything THIRI does is reachable through five calls. They are the same five whether you reach them over REST at `chords.thiri.ai` or as tools in the MCP server. This chapter is the musician's reading of each one; the field-by-field reference is on the [API page](#).

analyze: what is this chord doing?

Give it a symbol and a key. It returns the chord's parts (root, quality, intervals, notes) and its job in the key: scale degree, roman numeral, function, and whether it is diatonic or borrowed.

Ask for `Dm7b5` in C and you get `iiø7`, "predominant (borrowed)", with Locrian and Locrian ♮2 as the primary scales and the altered and half-whole diminished scales as secondary colours. That is the answer a good theory teacher gives, and it is the same answer every time.

resolve: which notes, exactly?

Give it a symbol. It spells the chord: note names, interval names, semitones from the root, MIDI numbers, and frequencies in hertz at the default octave. `C13` comes back as C E G B♭ D A, MIDI 60 64 67 70 74 81, with Mixolydian and Lydian dominant as the scales that fit.

This is the call a synth, a notation program or a tuner needs. There is nothing to interpret; the numbers go straight into a note-on.

voicing: how should an instrument play it?

Give it a symbol, a style, and optionally the previous chord's notes. It returns an instrument-ready voicing and a voice-leading score. The styles are the ones arrangers use: rootless (the default), Bill Evans rootless, shell, triad, pad, guide tones (both, or either one), drop-2 and drop-3. An unknown style name falls back to the default without complaint, so read the `style` echoed in the response.

With the previous voicing supplied, THIRI minimises motion between chords. `C13` after a `Gm7` voiced `G3 B♭3 D4 F4` comes back rootless as `E3 A3 B♭3`: the third and seventh stay as guide tones, the thirteenth takes the fifth's seat, and the fifth and ninth are listed as omitted so you know exactly what was left out.

reharmonize: what else could this be?

Give it a progression and a key. It returns alternatives, each named by technique, with the exact changes and a one-sentence explanation. For `Gm7 C7 Fmaj7` in F the first alternative is the tritone substitution, `Gm7`

`G♭7 Fmaj7` : the dominant a tritone away shares the guide tones and slides the bass down a half step into the target. Seven more follow: a ii-V insertion, modal interchange, secondary dominants, a chain of dominants, Coltrane changes, the backdoor dominant.

The explanations are part of the output on purpose. An agent that shows its reasoning is one a musician can argue with.

conduct: make it a band

Give it a sentence. "4-bar swung loop in F for piano, bass and drums" returns a conductor object (tempo 80, key F, 4/4, four bars, swing 0.55, an energy curve), four lanes of note events (harmony, pattern, bass, drums) at 480 ticks per quarter, a lead sheet (`| Gm9 | C13 | Fmaj9 | Fmaj9 |`), and a complete multi-track MIDI file encoded in base64.

The engine is pure JavaScript. It decides; something else plays. On this site the Conductor instrument renders the lanes with Csound in the browser. In a DAW you drop the MIDI on four tracks.

What every response carries

- A JSON object you can snapshot. There is no sampling, so a golden file written today is still correct tomorrow unless the [changelog](#) says the grid changed.
- Two headers, `X-Quota-Limit` and `X-Quota-Used` , so a client always knows where it stands in the month.
- On failure, a stable error code and a human message. The [errors page](#) lists them.

What the engine will not do

It will not guess a chord it cannot parse; it returns `invalid_chord` and the symbol it choked on. It will not accept a mode as a key; ask for B♭ Lydian through the parent key of F and the chord you want. And it will not render audio: that is the job of whatever you plug it into, which is the subject of the next part.

SOURCES

- [chords.thiri.ai](#) responses captured 2026-10-02 (build site docs/api.astro)

03. MCP and agents: putting the engine in the loop

What the Model Context Protocol server is, how its five tools map to the REST calls, why the agent has to be told to use them, what the hosted endpoint costs in milliseconds, and how a second server turns the harmony into sound.

A language model is a good bandleader and a poor copyist. It knows what the harmony should do and cannot be trusted to spell the chord. The Model Context Protocol, MCP from here on, puts THIRI within the model's reach: the model keeps the intent, the engine supplies the notes.

What the server is

MCP is the standard way to hand a tool to an AI client. The client lists the tools a server offers; the model picks one, sends arguments as JSON, JavaScript Object Notation, and reads JSON back.

THIRI's server runs two ways.

Local. The npm package `@bluesprincemedia/thiri-mcp` speaks MCP over standard input and output. Claude Desktop, Claude Code and Cursor start it for you with `npx`. Your key never leaves your machine except as a bearer token to `chords.thiri.ai`. The shortest wiring is one line in Claude Code:

```
claude mcp add thiri --env THIRI_API_KEY=YOUR_KEY -- npx -y @bluesprincemedia/thiri-mcp
```

Hosted. For clients that cannot run a local process, such as Claude on the web and on mobile, the endpoint is `https://mcp.thiri.ai/mcp`. Add it as a custom connector with your key. It is JSON-RPC over HTTP, so only POST answers: a plain GET on the host returns 404 and a GET on `/mcp` returns 405. Listing the tools needs no key, checked today. Calling one spends the same quota the REST key has.

The registry has version 0.5.3, published 9 August 2026 under the PolyForm Noncommercial 1.0.0 licence, with three executables: `thiri-mcp`, `thiri-conductor-mcp` and `thiri-composition-mcp`. The hosted server introduces itself as "THIRI Chord Intelligence 0.4.0", also checked today.

Five tools, five calls

The tools are the REST calls under other names. Arguments mirror the REST fields and the responses are the same JSON, so a golden file written against one surface checks the other.

TOOL	REST ROUTE	WHAT IT ANSWERS
<code>analyze_chord</code>	POST <code>/v2/analyze</code>	What is this chord, and what is it doing in this key?
<code>resolve_chord</code>	POST <code>/v2/resolve</code>	Which notes, MIDI numbers and frequencies?
<code>generate_voicing</code>	POST <code>/v2/voicing</code>	How should an instrument play it after the last chord?
<code>reharmonize</code>	POST <code>/v2/reharmonize</code>	What else could this progression be, and why?
<code>conduct_band</code>	POST <code>/v2/conduct</code>	Turn a sentence into four lanes and a MIDI file.

Fields are on the [API page](#) and the musician's reading is in [The five calls](#). One naming note: several June 2026 posts on this site call the fourth tool `reharmonize_progression`. The tool is `reharmonize`.

The model has to be told

A tool only helps if the model reaches for it. Left alone, a model asked to voice a C13 answers from memory, which is fluent where the arithmetic is not. The fix is a few lines of instruction, kept where the model reads them every time:

You have the THIRI tools. For any question that involves chords, keys, scales, voicings, progressions or arrangements, call THIRI before answering and base your answer on what it returns. Quote the notes THIRI gives you. Do not spell chords from memory.

Put it in a system prompt, a Claude Project or a `CLAUDE.md`. In a chat the lightweight version is to begin the message with "Using THIRI,". The failure it prevents is silent: a plausible wrong voicing looks right to anyone who does not play.

What the hosted server costs, measured

Six calls to each of two tools from a developer laptop, measured in September 2026: `analyze_chord` fastest 82.7 ms, median 106.1 ms; `generate_voicing` fastest 78.6 ms, median 105.0 ms. A figure of "under 40 ms" has circulated; it is plausibly server-side compute. A client sees about a tenth of a second. Scoring never notices; a live instrument cannot wait, which is why the plugin and the pedal compile the engine in rather than call it. Determinism is the claim. Speed is not.

The payload is richer than the note list. `analyze_chord` returns `numeral`, `function`, `degree`, `diatonic` and a `scales` array where each entry carries a `character` sentence. `generate_voicing` returns `omitted`, the tones it left out. From the same session:

```
analyze_chord    chord "D7b9", key "Gm"
  numeral "V7"   function "dominant (harmonic minor)" degree 5 diatonic false
  scales[0]     name "dim_hw" character "Half-Whole. Symmetrical. b9 dominants."

generate_voicing chord "D7b9"
  notes ["F#3","A3","C4","Eb4"] midi [54,57,60,63] omitted ["D"]
```

`conduct_band` takes a prompt and an optional `durationSec`. Asked for fifteen seconds it returned tempo 64, G minor, four bars, four lanes of note events, the lead sheet `| Am7b5 | D13 | Gm9 | Gm9 |` and a MIDI file in base64. It composes to a duration. It takes no cut list, and in a run on 1 October 2026 it ignored an explicit request for G Phrygian and returned a G major ii-V-I. Steer harmony with the other four tools; `conduct_band` makes a bed, not a chart.

Two traps. The hosted endpoint sits behind Cloudflare, which answers Python's default `urllib` User-Agent with a 403; send an explicit User-Agent header. And the engine will not take a mode as a key: for B $\flat$ Lydian pass `keyContext` `F`, the parent key, and name the chord you want.

Agents that show their reasoning

`reharmonize` is built for this. Each alternative carries a `technique`, the new `progression`, a `changes` list saying exactly which index moved from what to what, and a one-sentence `explanation`. For `Gm7 C7 Fmaj7` in F the first alternative is `tritone_sub`, `Gm7 Gb7 Fmaj7`, explained as the dominant a tritone away sharing the guide tones with chromatic root motion into the target. The API page lists all eight techniques.

An agent that quotes these fields is one a musician can argue with. A decision log can read: D7 $\flat$ 9 at bar 8, V7, dominant function borrowed from harmonic minor, voiced F#3 A3 C4 E $\flat$ 4 with the root omitted because

the bass covers it, scale half-whole diminished. Every musical fact in that sentence is engine output. Only the placement is the agent's own, and it should say so.

One gap: Composer C, the Csound studio at `composer.thiri.ai`, fetches only `generate_voicing` for its songs and keeps the note list; the numeral, the function and the omitted tones arrive and go unused. Its score-to-picture module reads them, but that path runs in local mode only.

The dual-MCP Csound loop

The hosted server decides. It does not render. To hear it, run a second server beside it.

```
{
  "mcpServers": {
    "thiri": {
      "command": "npx",
      "args": ["-y", "@bluesprincemedia/thiri-mcp"],
      "env": { "THIRI_API_KEY": "YOUR_KEY" }
    },
    "thiri-conductor": {
      "command": "npx",
      "args": ["-y", "-p", "@bluesprincemedia/thiri-mcp", "thiri-conductor-mcp"],
      "env": { "THIRI_API_KEY": "YOUR_KEY" }
    },
    "thiri-composition": {
      "command": "npx",
      "args": ["-y", "-p", "@bluesprincemedia/thiri-mcp", "thiri-composition-mcp"]
    }
  }
}
```

This is the package README's config with one change. The README omits `-p`. Whether `npx` then selects the companion bin from a package with three executables is unverified here, so the config above names the package with `-p` and the executable after it.

As of package 0.5.3, `thiri-conductor-mcp` offers four tools: `conduct_band`, `render_audio`, `play_audio` and `search_corpus`. `render_audio` posts a prompt or a prior conduct result to `POST /v2/render`, where Csound runs server-side, and can play the returned WAV locally. No Csound install is needed. `thiri-composition-mcp` offers ten tools over a composition intermediate representation behind `POST /v2/compose`, with a local `play_composition` preview through fluidsynth.

Here two sources disagree. The [agent recipes](#) page on this site, written in June 2026, still describes the companions as rendering Csound on your machine and says a local Csound install is required; the copy live at the time of writing also named the removed `build_csound_score` and `render_csound_wav`. The package changelog for 0.5.0, dated 23 July 2026, removed those tools and moved rendering behind the API; 0.5.3 renamed the corpus tool to `search_corpus`. Follow the changelog, the newer source.

The flagship loop is four prompts, pasted in order: analyse the progression with `analyze_chord`; arrange it with `conduct_band`; render it with `render_audio`; listen with `play_audio`, critique voice leading and register balance, and send one revision back through `conduct_band`. The API page marks `/v2/render` and `/v2/compose` as not yet stable for third parties; ask before building a product on them.

Composer C closes the loop in one process instead: its server holds the key, fetches voicings over the hosted MCP and renders with Csound 6.18. No second MCP, no key in the browser. The dual-MCP version runs tonight from a config file.

Still open

- The hosted server reports version 0.4.0 while the package is 0.5.3. Same five tools, two version lines.
- `conduct_band` takes no cut list and ignores an explicit mode. Scored to picture worked around it with the other four tools.
- The latency figures are six calls to each of two tools from one laptop, not a benchmark; measure from where your agent runs.
- Composer C's song compiler ignores the reasoning payload it already receives.

SOURCES

- [production/thiri-build-site/src/pages/docs/mcp.astro](#) (facts verified 2026-10-02)
- [production/thiri-build-site/src/pages/docs/api.astro](#) (live responses captured 2026-10-02)
- [production/thiri-build-site/src/pages/lab/agent-recipes.astro](#)
- [production/thiri-build-site/src/content/blog/connecting-music-theory-mcp-to-claude.mdx](#) (2026-06-11)
- [production/thiri-build-site/src/content/blog/22-ways-to-build-with-thiri-mcp.mdx](#) (2026-06-10)
- <https://registry.npmjs.org/@bluesprincemedia/thiri-mcp> (read 2026-10-02: 0.5.3, published 2026-08-09, three bins)
- <https://mcp.thiri.ai/mcp> initialize and tools/list, probed without a key 2026-10-02

Part II • In production

10. On the pedal: fitting the engine in 1.333 ms

The Elk Stomp measurements: five harmony voices plus chord intelligence inside the audio callback, the three builds it took to get there, and the six disciplines that made the numbers trustworthy.

The question was empirical: does a deterministic harmony engine plus five voices of pitch shifting fit inside a 1.333 millisecond audio callback on a dual-core Cortex-A7? The hardware is Elk Audio's Stomp: STM32MP157, two cores at 400 to 800 MHz, 48 kHz, a 64-sample buffer, Elk Audio OS 1.2.2, Sushi 1.3.0. The plugin is HORN SXTN, the THIRI VST, running headless.

The answer

With 1.0 meaning the whole deadline, measured on the board, three sweeps of six points at 60 seconds each:

VOICES	AVERAGE	WORST BLOCK
0 (engine and tracker only)	0.568	0.726
1	0.577	0.749
3	0.588	0.757
5	0.592	0.788

The fifth voice costs almost exactly what the first did: 0.005 of the block per voice. The finding is the flatness, not the headroom.

Three builds, one CSV

The data file is append-only and keeps the failures. These are the five-voice rows for each build in that file (the README prints 5.94 for the first build from a different row). Read top to bottom:

BUILD	AVERAGE AT 5 VOICES	WORST BLOCK	VERDICT
Phase vocoder, unsliced tracker	2.08	5.99	one block in eight six times over deadline
Phase vocoder, tau-sliced tracker	1.83	2.93	over real time at one voice
PSOLA, tau-sliced tracker	0.59	0.79	ships

Three lessons came out of that table and they apply to any real-time DSP, not only ours.

A worst-case spike is not a CPU problem. The first build looked healthy on average while a single $O(n^2)$ analysis fired once per hop and landed in one block. Spreading the identical arithmetic across the hop's eight blocks cut the worst block eightfold with no change in output.

No buffer size fixes an average. The phase vocoder exceeded real time at one voice. Doubling the buffer doubles the deadline and the work; the fraction stays put.

The cheap algorithm won by 52x. PSOLA costs 0.005 per voice, the phase vocoder 0.254. The board was never the constraint; the algorithm was.

Two findings worth stealing

Pin the control surface. This one is a single paired session recorded in the README and the port map, not a sweep in the data file; the clean baseline across sweeps is 0.74 to 0.83 worst block. The audio thread lives on CPU 1. An unpinned hardware daemon and control app landed on it and took the worst block from 0.770 to 2.273 while the average sat at a reassuring 0.60. `taskset -c 0` put it back to 0.760. Elk's reference launcher does not pin.

NEON is not free on an A7. The expected four-times gain on the tracker's multiply-accumulate loop measured as 10.7 cycles per vector operation in accumulation chains, about scalar VFP speed. GCC will not auto-vectorise float loops without `-funsafe-math-optimizations`. Budget from measurement.

The six disciplines

1. **The bit-identity gate.** Render a fixed input offline, hash it, compare to a committed baseline. A change is either provably output-identical or it is a change.
2. **Void the rows.** A found confound voids every row before it. Re-sweep.
3. **The confound checklist.** Competing load, governor, affinity and priority, NEON attributes in the deployed binary, denormals. Run before every sweep; record per row.
4. **Hash, don't timestamp.** The board has no real-time clock. Every row carries the SHA-256 of the deployed binary.
5. **Figures from data.** The figure script asserts the claims before it draws. It caught three errors already published in slides.
6. **The average lies.** Sushi has no xrun counter; a block maximum at or over 1.0 is the proxy.

How the engine got there

The C++ core, `thiri_core`, is a port of the JavaScript engine, not a fork, pinned to it by golden vectors generated from the JS: 16,353 in the published engine's README, with larger sets on the plugin repository's current branches (the exact count differs by branch). The arpeggiator engine (Helix) is pinned separately, 18,080 of 18,080 vectors bit-for-bit, so the pedal and the web app play the identical arpeggio for the same chart, seed and macros. One trap worth recording: the engine must not normalise voice order, because the vectors reproduce the JavaScript's own inconsistency on purpose; consumers sort.

Still open

The live-input Sushi configuration on the main branch still selects the phase vocoder by default; only the file-source configuration selects PSOLA. A live horn through the harmonizer with harmony engaged is not yet on disk. That is the gate before any demo video, and the next measurement to add to the CSV.

SOURCES

- <https://github.com/BluesPrince/thiri-meets-elk> (README.md, METHOD.md, data/profiling-results.csv)

12. Google Lyria, Magenta and Gemini: what THIRI can steer

The 2026-10-01 experiments: which Google music models exist, what THIRI can enforce in each, what was measured, the Lyria scale value that does nothing, and a melody generator with no model that beat Magenta on the study cue.

For each Google music model the question is narrow: can THIRI's output reach it as a constraint, and does the result follow? This chapter records the 2026-10-01 study, run on the carving-film cue from Scored to picture.

Why measure instead of trust

An earlier falsification on Stable Audio 3 found that text chords scored 2 of 12 and that audio seeding was bleed, not conditioning. The rule from it: an audio model is not "THIRI-steered" until its chord-following beats a control with no conditioning. Steering has to be symbolic, or it has to be measured.

What each model can hold

Ranked by what THIRI can enforce, from Google's own documentation as read on 2026-10-01.

LEVEL	MODEL	WHAT REACHES IT
Notes	Magenta RealTime 2, released 2026-06-04	A mask over 128 pitches every 40 ms. No key, tempo or chord input.
Notes	Gemini 3.1 Pro	Notes as text or structured data, rendered by Csound.
Notes	Coconet, <code>coconet/bach</code>	Keeps fixed voices, fills the rest in four parts.
Chords	Magenta.js 1.23.1: ImprovRNN, MusicVAE <code>mel_chords</code> , two multitrack models	A chord symbol per step.
Key and tempo	Lyria RealTime	A <code>scale</code> enum of twelve major and relative-minor sets; <code>bpm</code> , beats per minute, 60 to 200.
Key and tempo	Lyria 3.5, Lyria 3 Pro and Clip	Prompt text only.
Nothing	Lyria 2, Magenta RealTime 1, MusicFX, Dream Track	No musical input THIRI can hold.

Lyria RealTime's scale enum lines up with THIRI's parent-major key context: G Phrygian is `E_FLAT_MAJOR_C_MINOR`, B $\flat$ Lydian is `F_MAJOR_D_MINOR`. It sets the seven notes, not the home note. Changing scale or tempo needs a context reset, a hard cut with up to two seconds of lag. Google's own page calls its key and tempo "imprecise".

What ran on this machine

An Intel Core i5 Mac on macOS 13. Magenta RealTime 2 cannot install there: magenta-rt 2.0.3 needs JAX 0.9 or newer, and jaxlib's last Intel-Mac wheel is 0.4.38. Magenta.js runs under bun. Lyria RealTime is remote. Gemini ran headless through AntigraVity.

Method

One cue for everything: `Gm7 | Ebmaj7 | D7#9 | Cm7`, twice, at 120 beats per minute, in G Phrygian. THIRI's `analyze_chord` and voice-led `generate_voicing` output is the shared input. One scorer labels every note as chord tone, THIRI chord-scale tone, or outside. One renderer reuses the carving film's theremin and strings. Each model gets controls: no chords, the wrong key, or a tritone-shifted progression.

Gemini: colour, not correctness

Gemini 3.1 Pro, three runs in each of three conditions. Without THIRI it wrote 82 to 85 percent chord tones and hit a chord tone on every beat 1 and 3. With THIRI's analysis in the prompt, 86 to 90 percent and no outside notes. With THIRI as a Model Context Protocol tool the model could call, 71 to 90 percent at about twice the tokens and time. The memory records the top run as 91; the report rounds the same 90.5 percent share to 90.

That is a ceiling effect: the model already knows the chord tones. Over D7#9 the THIRI runs used E $\flat$, B and A $\flat$ from the half-whole diminished scale the engine named, where the plain runs used B $\flat$. THIRI changed the colour, not the correctness.

Magenta.js: conditioning that works

Ten takes per condition under bun. MusicVAE `mel_chords` landed 82 percent chord tones on THIRI's chords, 35 percent with no chords, 14 percent with a tritone shift. ImprovRNN: 73, 52 and 26 percent, p at or below 0.003. All 60 takes took 73 seconds on the i5. The controls show the conditioning is the mechanism. One limit: `mel_chords` reads only root and triad, so D7#9 arrives as D major.

Lyria RealTime: eleven scales work, one does nothing

Twenty-one sessions of 32 seconds. Eleven of the twelve scale values put 76 to 83 percent of the audio's energy in their own seven notes, up to 47 points more than Lyria's own pick for the prompt. The wrong set, A major, dropped G Phrygian's share from 92 to 36 percent. Tempo held: 119.7 on most takes for a request of 120. Chords in the prompt text were not followed: the D7#9 needs F# and A, which got 2 percent of the energy; 55 percent sat in key overall.

`E_FLAT_MAJOR_C_MINOR` is a no-op. In four of four same-seed pairs in the study it returned audio bit-identical to `SCALE_UNSPECIFIED`, including a bright prompt whose default pick held A naturals. The client library sends the value correctly, so the evidence points at the server. That remains unverified until Google answers. A standalone repro script added a fifth identical pair, which is the count the drafted bug report carries. The value is exactly THIRI's G Phrygian set. The workaround for G-minor cues is `B_FLAT_MAJOR_G_MINOR`, which shares six of the seven notes.

Exp4: a melody generator with no model

Exp4 asks whether THIRI alone can write a better line than Magenta: rules plus a seeded beam search over THIRI's analysis. The per-bar palette is THIRI's primary chord-scale intersected with the key; a borrowed chord keeps the engine's own scale, so D7#9 gets half-whole diminished. Strong beats land on chord tones, guide tones preferred. Non-chord tones resolve by step and leaps are recovered. The form is a four-plus-four period with bars 5 and 6 restating bars 1 and 2. A hit marks the step that must carry the peak. No training, no weights.

Ten takes, scored against the real chords: 86 percent chord tones, 0 percent outside, 100 percent of strong beats on chord tones. Steps 51 percent, leap recovery 100 percent, motif recurrence 56 percent. All ten takes peaked on step 96, on F#5 then F5, the #9. Controls: key only 58 percent, no chords 38, tritone-shifted 15, p 0.00005. Each take composes in 0.12 seconds.

Same cue, same scorer: Magenta `mel_chords` 82 percent chord tones, 5 percent outside, 22 percent steps, no motif recurrence, a 39-semitone range; Gemini with THIRI in the prompt 88 percent, 45 percent steps, 60 percent motif recurrence. Exp4 beat Magenta on chord tones, outside notes, strong beats, steps, leap recovery and motif recurrence, and tied Gemini on correctness. The one measure it lost, semitone rubs against the pad, counts doubling a pad note as a rub.

Exp5: a Lyria bed under THIRI's arrangement

A percussion-and-texture bed, bass muted, scale `B_FLAT_MAJOR_G_MINOR`, replaced the carving film's drum loop under the delivered stems. The take came out at exactly 120.0 beats per minute, sits on B $\flat$ and D over the chords, and is levelled section by section to the old drum stem's energy: 18 to 27 dB under the

music by the memory's reading, with the build report recording applied gains of 18 to 29 dB. The mix measures -16 LUFS, loudness units relative to full scale; the video stream is bit-identical.

The timing number changed; the change is the lesson. The first reading put the bed's strong onsets within 3.3 ms median of the film's beat grid. It came from the spectral-flux envelope that had placed the bed, which, calibrated on clicks and drum hits, leads audible onsets by about 13.5 ms. Measured against ground truth on 2026-10-02, the bed plays 14.5 ms late as a signed median. Measure against ground truth, never against your own placement envelope. Also: the same seed, recorded again on 2026-10-02 in the film studio, came out at 120.97 for 120, so a seed does not pin the tempo; stretch to lock. And pad the bed's front with silence, since Lyria's downbeat can land before the bar line.

Caveats

- One cue. Every number above is for one eight-bar progression at one tempo.
- Exp4 wins partly by construction: its rules are what the musicality measures reward. Its lines are plainer, about 10 semitones of range and 6.5 pitch classes a take.
- Gemini was near ceiling before THIRI arrived; a harder cue might separate the conditions.
- The Lyria no-op is five of five with one client; Google's confirmation would close it.
- THIRI's own `conduct_band` ignored an explicit mode: asked for G Phrygian, it returned a G major ii-V-I. Steer with `analyze_chord`, `generate_voicing` and `resolve_chord`.

What is open

- The blind listen of Exp4 against Magenta and Gemini.
- More progressions for Exp4, then range and tension tuned by ear, before it becomes a THIRI tool.
- The Lyria bug report, drafted for the Google AI Developers Forum, unposted.
- Exp5 shifted 14.5 ms and rebuilt.
- Magenta RealTime 2, the only model that takes THIRI's exact pitches, untested for want of a machine.

SOURCES

- <https://ai.google.dev/gemini-api/docs/music-generation> (Lyria RealTime controls, as cited in the steering memory; not re-fetched 2026-10-02)

13. Composer C: an agent that writes Csound with THIRI holding the harmony

A chat-driven studio where a language model edits a typed song spec, THIRI supplies every voicing, Csound renders deterministically, and nothing is called finished until it has been measured.

Composer C is a music studio you talk to. You describe a song, a Claude agent edits a small JSON document, JavaScript Object Notation, that describes it, the document compiles to Csound, the server renders it, and the browser plays it. No digital audio workstation or plugin chain sits underneath. The code lives in the repository `vibe-studio`; the name Composer C appears nowhere in it. The hosted instance is `composer.thiri.ai`, with accounts, a guest demo, chat and THIRI harmony all on.

From a sentence to sound

Every stage is typed.

STAGE	WHAT HAPPENS
Intent	A chat turn, a one-click verb or a knob change produces edit operations
Edit	The operations are validated and applied to the song spec; they are the only write path
Harmony	Any chord without a voicing is fetched from THIRI and frozen into the spec
Compile	The spec becomes Csound text: one wet stem per section and stem group, where a stem is a named set of layers such as drums or bass, a mix per section, one master
Render	A queue runs the <code>csound</code> command line, converts to FLAC, the Free Lossless Audio Codec, and stores the result by content hash
Play	The browser decodes the stems and sums them in Web Audio

Compile is pure and deterministic. The SHA-256 of the Csound text is the cache key, so the same bars never render twice. Stems are wet and linear; the only saturation lives in the mix and master units. Section boundaries fall on control-rate block edges, so sections overlap-add sample-exactly. The renderer is Csound 6.18.

What you touch

Six presets ship in the repository. Verb chips, named `darker`, `punchier`, `breathe`, `solo` and `extend` in the README, apply a fixed edit without a model call; the tiers document keeps them keyless and free at every tier. Chat edits need an Anthropic key on the server. An arrangement view shows sections and layers. MIDI out, the Musical Instrument Digital Interface, reaches your hardware through the browser's Web MIDI, so a hosted studio can still play the synth on your desk. A `.mid` export exists; on the hosted instance it is gated to accounts.

With no keys at all, the studio still makes music.

The harmony fence

The agent is not allowed to invent chords.

The rule: the agent never writes pitches except inside clip rhythms; harmony comes from THIRI voicings through a `pitch_source` field. The agent names a chord. The harmony bridge walks the song's timeline, finds every symbol without a voicing, and fetches each one through the THIRI MCP, the Model Context Protocol server, passing the previous chord's notes so the engine can voice-lead. The result is snapshotted into the spec. Compile never touches the network.

The hosted instance holds its THIRI token on the server, so a visitor needs no key. If the MCP call fails, the bridge falls back to a local textbook stub and tags the voicing `source="local-stub"`, so the app degrades honestly, not silently. Its docstring calls it plain music theory, not the THIRI engine.

One disagreement sits in that fallback. Both THIRI engines spell a dominant thirteenth with the natural eleventh present, as `0 4 7 10 14 17 21`. The stub's table, read from `vibe/harmony.py` on 2026-10-02, omits it: `0 4 7 10 14 21`. They have not been reconciled, so a stubbed 13th chord may differ from the engine's. The source tag on the voicing tells you which you got.

Designed instruments

Beside the hand-written orchestra voices there is a second kind: an InstrumentSpec, a JSON document in one of three families: analog, granular and FM, frequency modulation. The agent fills a schema and never writes orchestra text. The schema validates, compiles to deterministic Csound, and gets knobs automatically. The first phase was committed on 2026-09-13. Knobs become p-fields appended after the existing ones, with defaults equal to the literal they replace, so every render that existed before the knob stays byte-identical.

The skill's rules came from measurements during that build:

- FM here is phase modulation. Feedback runs inside a user-defined opcode at a control rate of one sample. Setting that rate at instrument level corrupted the global bus writes: clipping, and a left-right correlation of zero.
- A granular cloud from a single-cycle wavetable needs random grain phase, or the grain rate combs the spectrum. Pitched chords need grains of 200 ms or more; 250 ms grains at 40 per second is exact. Grains under 80 ms smear pitch into sidebands.
- Pulse waves away from 50 percent duty carry DC, a direct-current offset. The compiler inserts `dcblock2` before the drive.
- A five-note chord at half amplitude per note clips a hot carrier. Auditions scale by one over the square root of the polyphony.

Every instrument that came out right is saved as a recipe with its measurements, and the saver refuses anything that clips or carries DC.

Measure before claiming

The project rule is one sentence: for anything rendered, measure it with `csound-listen` before claiming it works. On 2026-09-13 the measurement scripts caught three defects in demos that had passed their tests: an FM patch clipping with decorrelated channels, a granular pad smearing its chord into the wrong pitches, and an analog patch carrying a 4.7 percent DC offset.

The skill turns words into numbers. From its table:

SOMEONE SAYS	MEASURE	AUDIBLE AT ABOUT
brighter, darker	spectral centroid	±8 percent
punchier	attack under 10 ms, crest factor	+2 dB crest
wider, narrower	side-to-mid level	±2 dB
in tune, on the chord	pitch classes and cents	within ±15 cents
broken	clipped samples, DC above 0.005, correlation near zero on a mono source	any

A master cannot show you a buried part; render the stems and measure each one, as Scored to picture learned.

The film studio built on it

On 2026-10-01 and 02 a video-first page was built on the same spine: picture on top, one track per sound source. Seven sources were wired: Csound, THIRI strings, Stable Audio 3 running on the CPU through `sa3.cpp`, a Lyria RealTime bed, a Magenta ImprovRNN line, a Gemini melody, and a free library that carries each sound's licence. A conductor track holds a tempo map at 480 pulses per quarter note with bar one on a frame; THIRI fits the tempo to the named story hits.

Measured, from the build log:

- A Csound impact placed on frame 48 lands at sample 96,000, which at 48 kHz and 24 frames per second is frame 48 exactly.
- The exported stems null against the mix to −126 dBFS, decibels relative to full scale.
- A Stable Audio take equals the command-line render of the same seed to $1.2e-7$, at about 33 seconds of compute per second of sound effect on this machine.
- Lyria onsets land a median 2.3 ms from the sixteenth-note grid.
- Magenta with seed 5 reproduces exactly, with 100 percent chord tones on strong beats and 83 percent overall.
- Gemini puts 100 percent chord tones on strong beats, at about 35 seconds and 5,000 tokens per line.

Export writes stems that sum to the mix at −16 LUFS, Loudness Units relative to Full Scale, an MP4 with the picture stream copied untouched, a cue sheet, a credits file with each take's licence, third-party notices where an MIT-licensed sound is used, score and conductor MIDI files, a review page and a verification report.

As of 2026-10-02 this page runs locally behind a feature flag and is not part of the hosted studio. Two things are open: the review page's solo button plays about 20 ms behind the picture, and the master prompt can add but cannot yet remove, mute or regenerate a track in place.

Known limits

- Note placement is hard-coded to 4/4; a spec in another meter does not yet place notes correctly.
- The README describes a public `git clone`. The repository was private on 2026-10-02, so those instructions do not yet work; treat the README as the intended release shape.

SOURCES

- `vibe-studio/README.md`
- `vibe-studio/TIERS.md`
- `vibe-studio/CLAUDE.md`
- `vibe-studio/vibe/harmony_bridge.py`, `vibe/harmony.py` (module docstrings and the chord table)
- `vibe-studio/vibe/templates/` (directory listing, 2026-10-02)
- `composer.thiri.ai/healthz` (probed 2026-10-02: `thiri mcp`, `thiri_token true`, Csound 6.18; recorded in the playable-instruments source map)
- `github.com/BluesPrince/vibe-studio` returned 404 on 2026-10-02 (site map); build site `src/pages/met.astro` 'source not public yet'

Part III • Building on it

24. How we measure

The six disciplines behind every number THIRI publishes, from the bit-identity gate to measuring the worst block, and how they transfer from a pedal to a film score to an API.

Every number on this site is supposed to survive disbelief. That is a method, not a mood, and it was written down in the open while the Elk Stomp measurements were being taken. The six disciplines below are from that public write-up. They apply to a plugin on a pedal, a score against picture, and an API response alike.

1. The bit-identity gate

A change is either provably output-identical or it is a change. Render a fixed input offline, hash the output, and compare it to a committed baseline. If the hash matches, the optimisation changed nothing you can hear and the old measurements still stand. If it does not, you have a new build and every number for it starts at zero.

On the pedal this is how PSOLA replaced the phase vocoder without the voicing engine's output moving. In the studio it is how a snare's tuning knob was proven byte-identical at its default before the knob was allowed to exist. For the API it is the reason golden response files are safe: no sampling, so a snapshot taken today is still correct until the changelog says the grid changed.

2. Void the rows

When you find a confound, every measurement taken before the fix is void. Do not patch the numbers; re-sweep. It feels expensive and it is cheaper than one wrong decision made on a number that was quietly contaminated.

The data file in the public measurement repo keeps the voided rows on purpose. Reading it top to bottom is the real story, failures included.

3. The confound checklist

Before every sweep, answer the same questions and record the answers per row: is anything else competing for the CPU; what governor is the processor on; which core is the audio thread pinned to and at what priority; does the deployed binary actually carry the vector instructions you think it does; are denormals flushed. On the Stomp, an unpinned control app took the worst block from 0.770 to 2.273 of the deadline while the average sat at a reassuring 0.60. The checklist is what catches that before it reaches a slide.

4. Hash, don't timestamp

The board has no real-time clock and reports a year that is wrong. File modification times are meaningless there. So every row records the SHA-256 of the deployed binary, which is the only reliable statement of

what produced a number. The same rule reaches the film work: a delivered mix is identified by its hash, and the picture stream is checked bit-identical against the original before the mux is trusted.

5. Generate figures from the data

The figure script reads the data file, asserts the claims the prose makes, and only then draws. A figure that cannot be regenerated from the file is a drawing, not evidence. Doing it this way caught three errors that had already been published in slides, which is the point.

6. The average lies. Measure the worst block.

Real-time audio fails on its worst block, not its average. Sushi has no dropout counter, so a block maximum at or over the deadline is the proxy. Every headline failure in the Elk work was invisible to average CPU. The same instinct carries to picture: a mix can sit at a healthy loudness while one foley hit lands 30 ms late, so each sound is measured against its own frame and the worst one is the number that gets reported.

What this looks like in practice

- Units are stated and labelled. The Stomp numbers are fractions of the 1.333 ms block, and a fraction is never printed with "ms" after it.
- A claim names its row. "0.592 average at five voices" points at a specific line in a specific file with a specific binary hash.
- A disagreement between two sources is written down as a disagreement, not resolved by picking the nicer number. Where this manual says "unverified", that is the method speaking.
- A measurement is not a recommendation. The first phase-vocoder build looked fine on average and was never going to ship; the data said so before anyone tuned it.

Where to see the receipts

The harness, the 52-row data file and the generated figures are in the public measurement repository, under an MIT licence, so you can run the same discipline on your own DSP and disbelieve us properly. The scoring work's verify script and level tables follow the same rules; the studio's listening tools measure a render before anyone claims it sounds right.

SOURCES

- <https://github.com/BluesPrince/thiri-meets-elk/blob/main/METHOD.md>
- <https://github.com/BluesPrince/thiri-meets-elk/blob/main/README.md> (section: The method)
- [production/thiri-build-site/src/content/manual/11-scored-to-picture.md](#) (the same rules applied to picture)

Appendix

40. Glossary

Fifty-four terms a developer meets in this manual, from guide tones to `SharedArrayBuffer`, each defined in a sentence or two with a musical example where it helps.

Terms are grouped by where you meet them. Each number names its chapter or page.

Harmony

- **Pitch class.** A note name with the octave removed; C4 and C5 are one. THIRI computes from sets of them.
- **Guide tones.** The third and seventh, the notes that fix a chord's quality. In a 2-5-1 in F they fall by half steps, F to E and B flat to A.
- **Voice leading.** Moving each voice the shortest distance into the next chord. The `voicing` call scores it, lower is smoother; chapter 2 cut 754 semitones to 449 across 32 bars.
- **Avoid note.** A scale tone that clashes with the chord when held. In the engine, the eleventh over any chord with a major triad on the bottom, so close C major skips F.
- **Roman numeral.** A chord named by its degree in the key, upper case major, lower case minor. `Dm7b5` in C is `iiø7` on the API page.
- **Predominant.** The job of a chord that leads to the dominant, as ii does. The `function` field of `analyze` uses the word.
- **Borrowed chord.** A chord taken from the parallel major or minor. `analyze` sets `borrowed` true and adds the word to the function.
- **Modal interchange.** Reharmonising with borrowed chords, one of the eight `reharmonize` techniques: `Gm7 C7 Fmaj7` becomes `Gm7b5 Cm Fm`.
- **Secondary dominant.** A dominant resolving somewhere other than the tonic, such as `D7` before `Gm7` in F.
- **Tritone substitution.** Swapping a dominant for the one a tritone away. C7 becomes G flat 7: same guide tones, bass sliding a half step into F.
- **Backdoor dominant.** The dominant a whole step below the tonic, B flat 7 resolving into C major. The technique name is `backdoor`.
- **Coltrane changes.** Key centres a major third apart, each reached through its own dominant, the pattern of "Giant Steps".
- **Chord-scale.** The scale that fits a chord in context. `analyze` names primary and secondary scales; D7 flat 9 in G minor gets half-whole diminished.

Voicings

- **Close position.** All voices within an octave. The engine's true close voicing is the scale down from the lead with avoid notes skipped, a cluster of seconds.
- **Rootless voicing.** Leaves the root to the bass, the API default. `C13` after `Gm7` returns E3 A3 B flat 3 with G and D omitted.

- **Drop-2.** A four-note close voicing with the second voice from the top moved down an octave. `Cmaj7` under E5 gives E5 B4 G4 C4.
- **Drop-3.** The same with the third voice: E5 C5 G4 B3. Both count the original stack.
- **Doubled lead.** The melody repeated an octave below as a fifth voice, added after the drop.
- **Lead sheet.** Chord symbols on a bar grid, no voicings. `conduct` returns one as text: `| Gm9 | C13 | Fmaj9 | Fmaj9 | .`

Notes and time

- **MIDI note number.** The Musical Instrument Digital Interface's integer for a pitch, 0 to 127, C4 at 60. `resolve` prints them beside names and frequencies.
- **ppq.** Pulses per quarter note, the tick grid of a MIDI file. `conduct` writes 480, so a bar of 4/4 is 1,920 ticks.
- **Swing ratio.** How late the off-beat eighth is pushed; in the usual convention 0.5 is even eighths. `conduct` returns `swing 0.55` on a loop it also labels `groove` straight, so the engine's own scale is undocumented (unverified).
- **Onset.** The instant a sound starts, measured in the audio, not scheduled in the score.
- **Foley sync.** How close a placed sound lands to its frame of picture. Chapter 11 measured a worst case of 1.0 ms over 31 sounds.
- **Click flag.** A sample-level jump the verify script marks. The carving film raised 41, all the sounds' own transients; film one raised none.
- **Stem.** One instrument or layer rendered alone. Stems showed what the mix hid: strings burying the theremin by 8 dB.

Loudness

- **LUFS.** Loudness Units relative to Full Scale, the broadcast loudness measure. The carving film delivered at -16.0 LUFS.
- **True peak.** The highest level once the waveform is reconstructed between samples, in dBTP. The same film measured -1.5 dBTP.

DSP on the pedal

- **Formant.** A resonance of a voice or instrument body that stays put when the pitch moves. A shifter that keeps formants moves the note and leaves the vowel.
- **PSOLA.** Pitch-Synchronous Overlap-Add. Grains two periods wide are cut at pitch marks and relaid at the new period, so timbre stays. 32 ms latency; 0.005 of the block per voice, measured on the board.
- **Phase vocoder.** Pitch shifting in the frequency domain. The plugin's runs a 2048-point fast Fourier transform with a 512-sample hop and cepstral formant correction: 42.7 ms latency and 0.254 of the block per voice, 52 times PSOLA.
- **Audio callback.** The function the host calls once per buffer. Miss its deadline and the output drops out.
- **Block.** One buffer. On the Elk Stomp, 64 samples at 48 kHz is 1.333 ms; every Elk figure here is a fraction of that block, not a time.

- **Worst block.** The costliest single callback in a run. Five PSOLA voices averaged 0.592 of the block and peaked at 0.788, measured on the board.
- **xrun.** A buffer underrun or overrun, the audible dropout. Sushi has no xrun counter, so a block at or over 1.0 is the proxy.
- **Elk Stomp.** Elk Audio's pedal development board: an STM32MP157 with two Cortex-A7 cores, Elk Audio OS, Sushi and a 64-sample buffer.
- **Sushi.** Elk's headless plugin host. It loads the VST3, reports timings as block fractions and is driven over gRPC.

Verification

- **Deterministic.** Same input, same output, byte for byte, so a response can be a golden file.
- **Golden vectors.** Outputs from the canonical JavaScript engine asserted against the C++ port: 16,353 in the published engine README; on the plugin repository the count differs by branch, 23,859 on the scale-close voicing branch and 28,479 on another as of 2026-10-02 (chapter 20).
- **Bit-identity gate.** Render a fixed input offline, hash it, compare to a committed baseline. Either it matches or it is a change.
- **Oracle.** A reference that does not guess. THIRI is deterministic, so a model's note scores as chord tone, chord-scale tone or outside with no human judge.
- **Blast radius.** The share of compiled stem units whose hash changes after an edit. `add_solo` touched 12 to 20 percent; a tone verb touched 100, because stems render wet.

Protocol and keys

- **MCP.** The Model Context Protocol, the standard for handing a tool to an AI client. THIRI's server exposes the five calls as five tools, in JSON, JavaScript Object Notation.
- **stdio.** Standard input and output. The local server speaks MCP over them, started by `npx`, so the key stays on your machine.
- **Hosted connector.** The remote server at `mcp.thiri.ai/mcp` for clients that cannot run a process, like Claude on the web. JSON-RPC over HTTP; a GET answers 405.
- **Bearer token.** The key, sent as `Authorization: Bearer sk_live_...` on every request. Shown once, rotated with a 24-hour overlap; a bug report carries only its prefix.
- **Rate limit.** Requests per minute per key: 30 free, 300 Builder. Over it is `429 rate_limited`.
- **Quota.** Calls per calendar month: 1,000 free, 10,000 Theory Pro, 100,000 Developer, 1,000,000 Licence. `X-Quota-Limit` and `X-Quota-Used` report it on every response; failed calls do not count.

Csound and the browser

- **Csound.** The sound compiler behind Composer C and the Conductor. Version 6.18 renders server-side; a WebAssembly build plays in the browser.
- **Orchestra.** The Csound text that defines instruments; the score says when they play. The Conductor's is the public file `thiri-band.orc`.
- **ksmps.** Samples per control block. Every Csound event lands on a block edge; at 48 kHz with `ksmps` 16, one frame of 24 fps picture is exactly 125 blocks.
- **WASM.** WebAssembly, the browser's compiled-code target. The Conductor imports `@csound/browser` and synthesises inside the page.

- **SharedArrayBuffer.** Memory that browser threads can share, unlocked only on cross-origin isolated pages. This site sets those headers on the three sequencer pages; the Conductor runs Csound without it.
- **VST3.** Steinberg's plugin format, the one Ableton Live and Sushi load. THIRI's plugin is built on JUCE and ships as VST3, Standalone and the headless pedal build.

SOURCES

- [production/thiri-build-site/src/content/manual/01-the-five-calls.md](#)
- [production/thiri-build-site/src/content/manual/02-voicings.md](#) (drop examples, voice-leading numbers)
- [production/thiri-build-site/src/content/manual/03-mcp-and-agents.md](#) (D7b9 chord-scale, hosted server)
- [production/thiri-build-site/src/content/manual/04-chord-vocabulary.md](#) (avoid-note rule)
- [production/thiri-build-site/src/content/manual/10-on-the-pedal.md](#) (block, worst block, per-voice costs, 16,353 vectors, xrun proxy, Stomp hardware)
- [production/thiri-build-site/src/content/manual/11-scored-to-picture.md](#) (LUFS, true peak, foley sync, stems, frame arithmetic)
- [production/thiri-build-site/src/content/manual/13-composer-c.md](#) (Csound 6.18)
- [production/thiri-build-site/src/content/manual/21-csound-lessons.md](#) (ksmps and the 125-block frame)
- [production/thiri-build-site/src/content/manual/22-ableton-and-daws.md](#) (JUCE and VST3)
- [production/thiri-build-site/src/content/manual/23-evaluating-llm-harmony.md](#) (oracle, blast radius)
- [production/thiri-build-site/src/pages/docs/api.astro](#) (live responses captured 2026-10-02: numerals, functions, techniques, ppq, swing, MIDI numbers, quota headers)
- [production/thiri-build-site/src/pages/docs/mcp.astro](#) and [install.astro](#) (stdio, hosted connector, 405 on GET)
- [production/thiri-build-site/src/pages/docs/auth.astro](#) and [errors.astro](#) (tiers, rate limits, rotation overlap)
- [production/thiri-build-site/src/pages/docs/support.astro](#) (key prefix in bug reports)
- [production/thiri-build-site/vercel.json](#) (cross-origin isolation headers on the sequencer pages)
- [production/thiri-build-site/src/pages/innovate/conductor.astro](#) lines 66 and 148 (@csound/browser import, useSAB false)
- [production/thiri-build-site/public/thiri-band.orc](#)
- [production/thiri-build-site/src/content/manual/20-porting-the-engine.md](#) (golden vector counts by branch)

41. Questions developers ask

Twenty short questions with honest answers about determinism, scope, keys, limits, browsers, offline use, DAWs, the pedal, logging, support and what is public about what comes next.

Short answers for developers on the application programming interface, API from here on, and the Model Context Protocol server, MCP.

Why not just ask the model?

Because the model predicts tokens and harmony is arithmetic over pitch classes. A fluent wrong voicing looks right to anyone who does not play. THIRI computes the notes; the model keeps the conversation. Measured on 1 October 2026 on one cue, Gemini 3.1 Pro alone put 82 to 85 percent of its melody notes on chord tones. With THIRI's analysis in the prompt it reached 86 to 90 percent, and the scorer marked no note as outside the chord or its THIRI chord scale. THIRI changed the colour, not the correctness. See [Google models](#).

Is it deterministic?

Yes. There is no sampling and no model in the loop. The same request returns the same bytes, so you can snapshot responses as golden files. Only a grid change alters an answer, and those are marked Engine on the [changelog](#). The C++ port inside the plugin is pinned to the JavaScript engine by golden vectors: 16,353 in the published engine, 23,859 on the current plugin branch. Two caveats. An unknown `style` on `/v2/voicing` falls back to the default without an error. And `conduct` once ignored an explicit mode.

Does it do melody or rhythm?

Harmony is the product. `conduct` is the exception: it returns four lanes of timed note events at 480 ticks per quarter note, including a pattern lane and a drum lane, plus a MIDI file, Musical Instrument Digital Interface. The Helix arpeggiator on this site turns voicings into motion. A rules-based melody generator exists as a research experiment on one cue, not as a public tool, see [Google models](#).

Can it render audio?

The engine never does. It returns notes; something else plays them. On this site the Conductor instrument renders with Csound in the browser. `POST /v2/render` exists behind the conductor MCP server and runs Csound server-side, but the API page marks it not yet stable for third parties.

What does a free key get?

All five calls, over the REST endpoints and the MCP server. 1,000 calls a calendar month, 60 a minute, one active key, no card. Over quota is a hard stop with `429 quota_exceeded` until the first of the month.

What do the paid plans add?

Theory Pro: 10,000 calls a month, 120 a minute, two active keys, \$5 a month at the founder rate.
Developer: 100,000 calls a month, 300 a minute, ten keys, \$20 a month at the founder rate. Licence: 1,000,000 calls a month, 1,000 a minute, 25 keys, \$1,200 a year, for commercial use of the MCP server. The founder rates are locked for life for anyone who subscribes by 31 December 2026. Plans are bought on the keys page and paid by card; the account's keys move to the new limits within a minute. Every tier is a hard cap. T.H.I.R.I. Builders on Skool is the community and does not change a key's limits.

Is there a rate limit?

Three. Per key per minute, 30 free and 300 Builder, returning `429 rate_limited`. Per IP, a flood throttle that normal use never meets. Per month, the quota above. Successful `/v2` calls count; 4xx responses do not. Every response carries `X-Quota-Limit` and `X-Quota-Used`.

Can I call it from a browser?

Only from an allowlisted origin. In the worker source, cloned 23 July 2026, the cross-origin resource sharing list, CORS, holds four THIRI hosts; the deployed worker, dated 1 September, was not re-read, so treat the list as unverified. A page on your own domain is blocked either way, and a key in client code is visible to anyone. Put a small server-side proxy in front. The API page says this site's instruments do that; the Helix client source disagrees, posting straight to `chords.thiri.ai` with the key in session storage, allowed because `build.thiri.ai` is on the list. Conductor's keyless plays do use the proxy, at `/api/play/`.

Is the engine open source?

The MCP server is source-available: `github.com/BluesPrince/thiri-mcp`, PolyForm Noncommercial 1.0.0 for recent releases and MIT for earlier ones. The measurement harness `thiri-meets-elk` is MIT. The engine itself is not published. Results are public, the API is key-gated, and the derivation is sealed. See [licences](#).

Can I run it offline?

Not today. The local MCP package speaks to your client over standard input and output, then calls `chords.thiri.ai` with your key. The plugin and the pedal compile the engine in and need no network, but neither has a public download yet. Helix, Band Studio and the sequencers on this site keep working without a key through a local chord parser, which is not the full engine. Conductor has no local fallback and needs the network, through your key or the site's shared-key proxy.

Which DAWs?

One digital audio workstation, DAW, has a confirmation on record: Ableton Live 11.3.43 ran the THIRI VST3, Steinberg's plugin format, on a live saxophone on 2026-09-26. The hackathon demo in August ran the same desktop plugin in Live. Every automated test runs under Elk's host, Sushi. Routing rules and the typed chart are in [Ableton Live, plugins and DAWs](#).

What about the pedal?

HORN SXTN on the Elk Audio Stomp is a working prototype, not a product. Measured on the board as fractions of the 1.333 ms block: five voices at 0.592 average and 0.788 worst block, 0.005 per added voice. The team took first place in the Elk Audio challenge and second in Roland Project LYDIA at the MUTEK Montréal hackathon, 22 to 23 August 2026. The board measurements came afterwards. See [On the pedal](#).

How do I report a wrong voicing?

Send the endpoint or tool, the exact request body, the response with its code, the time in UTC, your key prefix and never the full key, and what you expected musically. "C13 should keep E and B♭" gets fixed faster than "wrong voicing". Server bugs go to the GitHub issues of `thiri-mcp`. Engine answers go to the Builders room, where people who play can argue the voicing.

Does it log my queries?

Yes, and the terms say so. Usage logs, meaning volumes, latency and error rates, are kept per key. Request bodies are stored to improve the chord vocabulary. Do not put secrets in a chord name.

What languages?

Any language that can send an HTTP POST with a JSON body, JavaScript Object Notation. The docs carry curl, Node and Python. The MCP package needs Node.js for `npm`. Chord symbols are standard jazz notation, flats as `b` and sharps as `#`. Keys are tonics; a mode goes in as its parent key.

Can agents use it in n8n?

Yes. `n8n-nodes-thiri` 0.1.0 is a community node, published 15 June 2026 per the npm registry. The install page says it exposes all five endpoints and works as a tool for n8n AI Agents; that is unverified against the package source. Install it under Settings, Community Nodes, then add a THIRI API credential holding your key.

Where is the changelog?

At </docs/changelog/>, newest first, with release dates from the npm registry. Releases are tagged on GitHub. Pin a version in production: `npm -y @bluesprincemedia/thiri-mcp@0.5.3`.

How do I get support?

The Skool community is the fastest answer and where outages are posted first. Email gets a reply within a business day on Builder; the address is on the [support page](#). A 20-minute call books at </book>. The health endpoint is `GET https://chords.thiri.ai/health`. A public status page is on the roadmap.

How fast is the hosted MCP?

About a tenth of a second from a laptop. Six calls per tool in September 2026 gave medians of 106 ms for `analyze_chord` and 105 ms for `generate_voicing`. A figure of "under 40 ms" has circulated; it is plausibly server-side compute, not what a client sees.

What is coming next?

Only what is public. The changelog's unreleased entry lists the developer site rebuild, a play space per instrument, case studies for the pedal and the scoring work, this manual in Markdown and PDF, and the

Get Started Guide delivered after sign-in. A signed plugin download is on the list, per the licence table in [licences](#), not a dated roadmap. Larger tiers are not published.

SOURCES

- [production/thiri-build-site/src/pages/docs/api.astro](#) (live responses captured 2026-10-02; the CORS and proxy note; /v2/render status)
- [production/thiri-build-site/src/pages/docs/auth.astro](#) (tiers, rate limits, logging, rotation)
- [production/thiri-build-site/src/pages/docs/errors.astro](#) (429 codes)
- [production/thiri-build-site/src/pages/docs/mcp.astro](#) (source repository, licence, hosted endpoint, version pin)
- [production/thiri-build-site/src/pages/docs/install.astro](#) (clients, n8n install steps)
- [production/thiri-build-site/src/pages/docs/support.astro](#) (channels, bug-report checklist, health endpoint)
- [production/thiri-build-site/src/pages/docs/changelog.astro](#) (release dates from the npm registry, read 2026-10-02; the unreleased entry)
- [production/thiri-build-site/src/pages/keys.astro](#) (pre-release developer terms, access-code panel)
- [production/thiri-build-site/src/lib/helix/thiriClient.ts](#) (direct browser calls to chords.thiri.ai, local chord fallback)
- [production/thiri-build-site/src/pages/innovate/conductor.astro](#) and [src/pages/api/play/\[tool\].ts](#) (no local fallback; keyless plays go through the server-held play key since commit 0bf2b1a, 2026-10-02; the instruments inventory of the same day recorded the earlier key gate)
- [scratch inventory thiri-playable-instruments.md](#) (key-gating summary: Helix, Band Studio and sequencers fall back locally, 2026-10-02)
- [production/thiri-build-site/src/content/manual/03-mcp-and-agents.md](#) (hosted latency, six calls per tool, September 2026)
- [production/thiri-build-site/src/content/manual/10-on-the-pedal.md](#) and [20-porting-the-engine.md](#) (board numbers, golden vector counts)
- [production/thiri-build-site/src/content/manual/22-ableton-and-daws.md](#) (the Live 11.3.43 confirmation of 2026-09-26)
- [production/thiri-build-site/src/content/manual/90-licences-and-hand-off.md](#) (what is open, what is sealed; the signed download row)
- [production/thiri-build-site/src/content/manual/12-google-models.md](#) (Exp4 melody generator)
- [production/thiri-build-site/src/content/manual/30-the-builders-playbook.md](#) (access-code panel, where bugs go)
- <https://github.com/BluesPrince/thiri-meets-elk> (README.md, data/profiling-results.csv)
- [scratch inventory memory-and-plans-inventory.md](#) themes 1.3, 2.8, 2.10 and 9 (2026-10-02)

90. Appendix: licences and hand-off rules

What is open, what is source-available, what is private, and the rules for sound libraries and other people's work when THIRI output ships in a product or a film.

THIRI is one engine with several front doors, and they are not all licensed the same way. This appendix says what you may do with each, and the rules we follow when other people's sound or work passes through a THIRI project.

The THIRI pieces

PIECE	LICENCE	WHAT IT MEANS FOR YOU
The REST API at chords.thiri.ai	Pre-release developer terms, accepted when you create a key	Build and ship apps on it. The engine, endpoints and brand stay ours. No uptime guarantee yet.
@bluesprincemedia/thiri-mcp (the MCP server)	PolyForm Noncommercial 1.0.0 for recent releases; earlier releases MIT	Free for personal, research and non-commercial use. Commercial embedding of the server: talk to us. The LICENSE file in the version you install is authoritative.
thiri-meets-elk (the measurement harness and data)	MIT	Use the method, the scripts and the data however you like, with attribution.
thiri-meets-lyria (the Google experiments)	Apache-2.0 with NOTICE	Private at the time of writing; the licence applies when it opens.
The Csound agentic skill pack	Apache-2.0	Use and adapt the skills; keep the notice.
The THIRI engine itself (JavaScript canonical, C++ port)	Not published	Results are public, the API is key-gated, the derivation is sealed. Engine internals are not discussed in public without a written go-ahead.
The THIRI VST / HORN SXTN plugin	No public release yet	The Ableton install video shows the desktop path; a signed download is on the list.

Trademark: THIRI is a trademark of Blues Prince Media. We do not say "patent pending" anywhere, and neither should material that quotes us.

Third-party pieces inside our builds

PIECE	RULE
Elk Audio Stomp SDK	Not public. Never publish its link, installer name or paths. Reproduction of the pedal work needs Elk's devkit.
Sushi (Elk's audio host)	AGPL. A commercial pedal needs a licence from Elk.
JUCE 8	AGPLv3 or commercial, dual-licensed. A shipped plugin needs the commercial licence or AGPL compliance.
Sonniss GDC bundles	Fine inside a mix or stems for a film; no raw redistribution; no use for model training.
Splice samples	Inside productions only; never in a corpus; never raw files handed off.
VSCO 2 Community Edition, Freesound CC0 items, Kenney packs, the uisfx set	CC0. Credit is a courtesy, not a condition.
Unity UI Audio Collection	MIT. Ship the notice.
World of Warcraft audio bundled in a third-party sound repo	Blizzard's. Refused in code; never used.
Lyria RealTime output	Google's terms; carries SynthID; regenerable from a seed.
Gemini and Stable Audio outputs	Google API terms; Stability Community License respectively.

Hand-off rules for a scored film

- The filmmaker receives the mix, the stems and the credits file. Never raw library files.
- Their films and their breakdown documents stay theirs; nothing of theirs is shown publicly without their written permission, and the page says so when it has it.
- Every delivery ships with a measured report: loudness, true peak, click count, sync per sound, key.
- Found sounds are credited per file in the package, with the source and the licence.

Hand-off rules for the pedal work

- Elk's private materials and Elk staff names stay in private documents.
- Photographs of other people need their consent before they appear anywhere public.
- The measurements, the method, the port map and the figures are public, with the column labelled and the binary hashed.

Hand-off rules for research posts

- A number names its source. If two sources disagree, the post says so.
- Nothing derived from a fabricated figure is ever republished, even after correction.
- A negative result is published with the same care as a positive one; the measurement repo keeps the failed rows on purpose.

SOURCES

- <https://registry.npmjs.org/@bluesprincemedia/thiri-mcp> (licence field, read 2026-10-02)
- <https://github.com/BluesPrince/thiri-meets-elk> (LICENSE: MIT)
- scratch inventory research-docs-inventory.md §14, consolidated licensing and consent register (2026-10-02)