lingcang728 Merge pull requ... 2ca4164 · 40 minutes ago

.agent_context	chore(agent): uncom...	3 weeks ago
.github	docs(appimage): 把 CI ...	2 days ago
docs	chore(release): 2.2.0	6 hours ago
functions/api	feat(开放平台): 登记 Ze...	6 hours ago
migrations	fix(startup): 起不来的...	yesterday
packaging	chore(release): 2.2.0	6 hours ago
public	feat(ui): 首页视觉重构...	3 weeks ago
scripts	feat(开放平台): 登记 Ze...	6 hours ago
server/zepp	feat(开放平台): 登记 Ze...	6 hours ago
src-tauri	fix(重放): 重放一次以上...	1 hour ago
src	fix(膳食平衡): 环是空的 ...	1 hour ago
tests	feat(开放平台): 登记 Ze...	6 hours ago
.dockerignore	feat(linux): Flatpak、d...	3 days ago
.gitignore	fix(重放): 重放一次以上...	1 hour ago
.nvmrc	release: 2.0.0 —— 同...	3 days ago
CHANGELOG.md	docs(changelog): 补上 ...	1 hour ago
LICENSE	feat: publish ZeppBrid...	3 weeks ago
NOTICE	feat(ui,data): 新增身体...	last week
README.md	fix(startup): 起不来的...	yesterday
README.zh-CN.md	fix(startup): 起不来的...	yesterday
bundle-budget.json	chore(bundle): 合并 m...	12 hours ago
index.html	release: 2.0.0 —— 同...	3 days ago
package-lock.json	build(deps): typescript...	2 days ago
package.json	chore: 发版分支跟上 m...	2 hours ago
tsconfig.json	feat: publish ZeppBrid...	3 weeks ago
tsconfig.node.json	feat: publish ZeppBrid...	3 weeks ago
vite.config.ts	build(vite): 显式钉死产...	2 days ago
vitest.config.ts	perf(frontend): 首屏从 ...	last week
wrangler.jsonc	feat(feedback): 补全设...	last week

About

A local-first Windows desktop bridge for syncing, exploring, and exporting your Zepp wearable health data.

zeppbridge.pages.dev

health-data | local-first | rust | tauri | vue | windows | zepp

Readme | MIT license | Activity | 92 stars | 2 watching | 14 forks | Report repository

Releases 28

ZeppBridge 2.1.2 Latest — yesterday

+ 27 releases

Contributors 4

- lingcang728 LingCang Hu...
- claude Claude
- george-petrakis G. Petrakis
- Autumnhui 丘天惠

Languages

Rust 55.7% | Vue 24.3% | TypeScript 11.6% | JavaScript 3.5% | Python 3.2% | PowerShell 1.1% | Other 0.6%

README | MIT license

ZeppBridge

Your Zepp data, handed back to you.

View, archive and export your Amazfit health records on your own Windows, macOS or Linux machine.

CI passing | license MIT | Windows supported | macOS Apple Silicon community tested | Linux builds only | version v2.1.2

简体中文

> **Important**
>
> ZeppBridge is an independent, unofficial open-source project. It is not affiliated with or endorsed by Zepp Health, Huami or Amazfit. Use it only with accounts and data you are entitled to access.

> The app ships in English and Chinese; it follows your system language on first launch, and Settings has a switch. This page and its Chinese counterpart are kept in step with each other, and nothing here is described more generously than it is implemented.

Isn't this already in the Zepp app?

It is — but only on your phone, only the way the official app chooses to show it, and it lives on someone else's server. ZeppBridge addresses a few specific things:

- **See it on a real screen.** Long-term trends for heart rate, sleep, workouts, recovery, stress and SpO_2, over 7 days / 1 month / 6 months.
- **The data sits on your own computer.** Everything lands in one file on your machine. It keeps working offline, across phone changes, account deletions and app redesigns.
- **You can backfill history from before you installed it.** Month by month, pausable and resumable, and honest about which months the cloud genuinely had nothing for versus which ones simply haven't been fetched yet.
- **Backups that actually restore.** Whole-database snapshots with checksums and integrity checks, and a record-count diff shown before you restore.
- **Export whenever you want.** JSON, CSV and GPX — drop them into Excel, Strava or your own scripts.
- **Hand it to an AI in one step.** Pick a date range and data types; the app packages it in a model-readable shape, strips identifying details, and copies it to your clipboard.
- **Usable without opening a window.** Ships with a non-interactive CLI (schedulable via Task Scheduler or cron) and a read-only MCP server, so a model can query your local data without the data leaving your machine.

One thing worth stating plainly: **it will not prettify your data.**

If you didn't wear the watch that day, the chart has a gap. If your watch never measured something, the interface says "not provided" — never `0`. No GPS track means no map. In health data, an invented smooth curve is worse than an honest gap.

The same applies to the word "complete": the interface only claims a **complete local copy** once the coverage ledger shows every month chunk has reached a conclusion. Until then it says "a local copy of the range that synced successfully".

Which devices are supported

If your device syncs to the Zepp app, it's worth trying. ZeppBridge reads what your account holds in the cloud; it does not talk to the watch, so it is not tied to specific models.

The bundled catalogue recognises 52 Amazfit products across the **GTR, GTS, T-Rex, Balance, Active, Bip, Cheetah, Falcon, Helio and Band** families (watches, bands, straps, rings). Recognised devices show the correct model name and product image; unrecognised ones still sync — they just show a generic name, and you can identify yours by hand.

Which metrics you actually get depends on what your watch measures. After connecting, the settings page lists this for your account, item by item.

Download and install

Get the latest build from Releases.

Windows

1. Download `ZeppBridge_<version>_x64-setup.exe` (or `.msi`) and run it.
2. There is no code-signing certificate yet, so Windows may warn about an unknown publisher. Choose **More info → Run anyway**.
3. Later versions install over the top; your data is not touched.

macOS (Apple Silicon)

> **This is an unsigned build.** There is no Apple Developer ID certificate and no notarisation, so macOS will refuse to open it until you clear the quarantine flag yourself. The steps below are a deliberate workaround, not a fix — see #2.

1. Download `ZeppBridge_<version>_aarch64.dmg` and drag `ZeppBridge.app` into Applications.
2. First launch will fail. Which message you get depends on your macOS version:
   - **"unidentified developer" → Right-click the app → Open → Open.**
   - **"ZeppBridge is damaged and can't be opened"** → right-clicking will *not* help. Run this in Terminal, then open the app normally:

     ```
     xattr -dr com.apple.quarantine /Applicat:
     ```

   The app is not actually damaged. That message is what Gatekeeper says about any downloaded bundle that is not notarised. Only run a command like this for software you have decided to trust — you can read every line of this one on GitHub and build it yourself.
3. macOS builds are covered by CI (compile, clippy, tests) and one contributor smoke test on Apple Silicon. The maintainer does not own a Mac and cannot independently verify sync or keychain behaviour. If that matters to you, prefer Windows.

Why it stays this way for now: notarisation itself does not need a Mac — CI already runs on macOS runners and could sign and notarise there. What is missing is an Apple Developer Program membership (99 USD/year), which the project has not bought. If that changes, this section goes away.

Linux (x86_64)

> **Builds, but nobody has run it yet.** CI compiles it, runs the tests and builds the packages on every push. What has *not* happened is a full sign-in-and-sync cycle on a real Linux desktop — including whether the token lands in your keyring correctly. Treat this as a build you are helping to test, not a release.

Flatpak, `.deb`, `.rpm` and an AppImage are published on the release page. Nothing is signed; check downloads against `SHA256SUMS.txt`.

```
sudo apt install ./ZeppBridge_<version>_amd64.del
sudo dnf install ./ZeppBridge_<version>_x86_64.r|
flatpak install ./ZeppBridge_<version>_x86_64.fl
```

The Linux guide covers where the data goes, how the token is stored when you have no keyring, and how to build from source.

There is also a headless container image with just the CLI and MCP server, for keeping a library synced on a NAS or a server. It cannot sign in — that still needs the desktop app once.

Not supported: Intel Macs, mobile.

What is verified on which platform

Same app, same interface, same features on all three — what differs is how much of it anyone has actually checked. Asking here beats guessing.

	Windows 10/11 (x64)	macOS Apple Silicon	Linux x86_64
Interface and features	identical	identical	identical
Built in CI	yes	yes	yes
Automated tests in CI	yes	yes	yes
Installer opens without a workaround	yes (unknown-publisher warning)	**no** — see the unsigned-build note above	yes
Sign-in, sync, export	verified by the maintainer on every release	contributor smoke test only	**nobody yet**
Credential store	Credential Manager, verified	Keychain, not independently verified	Secret Service, **nobody yet**
Auto-update	verified	built, not independently verified	n/a — your package manager

The maintainer develops on Windows and does not own a Mac or use Linux on the desktop. Nothing above is a statement that macOS or Linux is broken — it is a statement about who has checked what. If you use either and something misbehaves, a report is genuinely useful.

From 1.0.0 the local database's schema and upgrade path are treated as something to maintain long-term: every migration takes an automatic backup first, and snapshots can be verified and restored. Your data stays local — but the snapshots live on the same disk as the database, so **if you are worried about drive failure, copy one somewhere else yourself.**

First connection

1. Open ZeppBridge and go to **Settings** in the sidebar.
2. Click connect. The **official Zepp login page** opens in its own window; sign in with your usual credentials.
3. Once it says connected, the window closes and the app runs its first sync. Give it about 40 seconds.

Both mainland-China and international accounts work; the app detects which regional server you belong to.

The first sync fetches 30 days so there is something on screen quickly, then keeps going in the background until 180 days are in. The progress is visible and you can stop it at any time. Later syncs are incremental.

Every "last N days" selector in the app — on the training and body screens, and on the export page — reads your **local** library, not the cloud. If you pick a range that reaches further back than what this machine holds, the app says so and offers to fetch the rest. A blank stretch in a chart means *not fetched yet*, never *you recorded nothing then*.

For history older than 180 days, use **Long-term archive and full history** in Settings: choose 1/2/3 years or a custom start, and it fetches month by month. You can stop at any point and continue later. Before starting, it estimates disk usage from the actual rate your own data accumulates — not from a hard-coded constant.

If the range exceeds your local retention window, the app requires you to enable long-term archiving first; otherwise the history you just fetched would be cleaned up after the next successful sync.

Stuck at login? See the connection guide for troubleshooting and two fallback methods.

What you get

Trends

Page	What it shows
Overview	Heart rate over the last few hours, today's steps, last night's sleep structure, this week against your own previous 28 days, and entry points to body and training status. Every card opens
Heart rate	The full 24-hour curve, plus per-day trends for resting heart rate and HRV under two definitions
Daily activity	Per-day trends for steps, distance, active calories and active minutes
Body status	Recovery, stress, SpO_2, HRV, respiratory rate and resting heart rate over time
Training status	VO_2max, training load, lactate threshold, PAI, and whether recent volume is high or low
Recent records	Every sleep session and workout, each openable in detail
Workout detail	Distance, pace, heart rate, per-kilometre splits, GPS track; running also shows power and form
Devices	Where each device's model came from (catalogue match or your own assignment), firmware, most recent data — reassignable at any time
Data health (Settings → Advanced and maintenance)	Per-stream fetch / parse / write state — whether a gap means "not synced" or "nothing was ever measured"

A metric with no data does not sit there showing "—"; it simply does not appear. And a curve breaks wherever more than 15 minutes passed without a sample, rather than drawing a straight line between the two ends.

Post-workout insight and weekly report

After a workout, the app compares it against your own history: recent runs in the same distance band, and how pace, heart rate and training load differ — along with how many samples that rests on and how confident it is. **The baseline is you, not a population norm.** When there aren't enough samples it says so, rather than lowering the bar to produce a sentence. These are facts and evidence; interpretation is left to an AI.

Hand it to an AI

Several prompt templates are built in (performance summary, training insight, recovery assessment, sleep analysis). Pick a template and range, and the app packages the data, strips device identifiers and precise locations, copies it to the clipboard and opens the AI site you chose.

A workout detail page has its own "hand to AI" button, scoped to **that one workout**: the workout itself and the per-point metrics recorded while it was happening. Per-day records such as sleep and step counts do not go with it.

Packages over 2 MB are written to a file on your desktop instead, ready to drag into the conversation.

Export files

- **JSON** — full structured data, for scripts or models
- **CSV** — tabular summary for spreadsheets
- **GPX** — standard tracks for Strava, Garmin and others

What an export contains: workout summaries (type, start and end, distance, calories, average and peak heart rate, training load), daily metrics (steps, resting heart rate, HRV, SpO2, stress, respiratory rate, PAI, VO2max), and sleep sessions with their stage timeline. Choosing **Full** instead of **Summary** adds per-second workout series and individual heart rate readings.

`.fit` is its own export format, one file per workout, written into a folder you pick. It carries the per-second series ZeppBridge decoded from Zepp's workout detail: GPS track, heart rate, speed, altitude, running power, ground contact time and vertical oscillation, plus per-kilometre laps and pause events. Fields that were never measured are simply absent — nothing is padded to make the file look complete. Cadence is deliberately left out: its unit cannot be reconciled against any summary field we hold, and a wrong unit would silently read twice too high.

What an export does not contain: `.tcx`, account details, tokens, or device serial numbers. GPS tracks appear in GPX and FIT, and only for workouts that actually carry a track.

It does not get heavier over time

Raw cloud payloads are the largest thing in the local database. ZeppBridge stores them compressed — anything newly synced arrives compressed, and the first launch after an update compacts the existing ones in the background and reclaims the disk space, with progress at the top of the window that disappears when it finishes.

Before replacing a payload it decompresses it again and compares byte for byte, skipping any that does not match: the raw payload is the only basis for re-parsing locally, so not compressing is always better than compressing it wrongly. A measured 211 MB database came out at 55 MB.

Leave it running

Closing the window leaves the app in the tray, still syncing on its own. If you do not want it running, right-click the tray icon and quit.

Without a window

Each release also ships `zeppbridge-tools-<version>-<platform>.zip` containing two programs:

- `zeppbridge-cli` — non-interactive: `status`, `sync`, `export`. Exit codes are a stable contract, so it schedules cleanly under Task Scheduler or cron.
- `zeppbridge-mcp` — read-only MCP server over stdio. No ports, no network. Lets a model query your local data without the data leaving your machine.

See CLI and MCP for usage and configuration examples. The MCP section in Settings also offers a block of text you can paste straight to an AI, so it can walk you through configuring it for your machine.

Local read-only REST

Settings can enable a read-only endpoint bound to `127.0.0.1` only, for your own scripts. It is off by default, requires a token once enabled, returns no credentials, and never listens on the local network.

What's changed

Per-version changes are in CHANGELOG.md. When Settings → Software update → Check for updates finds a new version, it also shows you the release notes directly, and reports progress while downloading.

FAQ

Does my computer need to stay on? No. Each launch catches up on the period you missed.

Can I stop using the Zepp phone app? No. The chain is: watch → Zepp app on your phone → Zepp cloud → ZeppBridge. Your watch still needs the phone app to upload. Open it occasionally.

Could this get my account banned? ZeppBridge uses your own credentials and **only ever issues read requests** — there is not a single write request anywhere in the project; you can grep for it. Behaviourally it is the same as opening the official app to look at your data. It is still an unofficial use, and we cannot make guarantees on Zepp's behalf.

A metric came back empty. First check whether your watch actually measured it. Some metrics (lactate threshold, VO_2max) only update after specific workouts, a handful of times a year. The settings page reports each one for your account — note that **"not retrieved" is not the same as "your watch doesn't support it"**: Zepp's API returns an empty response for data that doesn't exist *and* for stream names that were never valid, so emptiness alone proves nothing.

Where is my data?

- **Windows**: a `data` folder next to the install directory (not `%APPDATA%`). Settings → Advanced has a button to open it.
- **macOS**: `~/Library/Application Support/com.zeppbridge.ZeppBridge/data`
- **Linux**: `~/.local/share/zeppbridge/data` (Flatpak: `~/.var/app/com.zeppbridge.app/data/zeppbridge/data`). An AppImage or an unpacked tarball keeps `data/` next to the executable — see the Linux guide.

The app will not start — a window never appears. Two things to look at, in this order:

1. **The error dialog.** From v2.1.2 on, a start-up failure shows a dialog naming the exact folder and the OS error instead of exiting silently. Earlier versions exited without a word, which looked like "the tray icon is there but clicking Open does nothing" — that icon was a leftover from a process that had already gone.
2. **The log.** `logs/zeppbridge.log` inside the data folder (see the previous question), plus `logs/startup-error.log` if the last launch failed before the window existed. Attach those to a bug report; they contain paths and version numbers, no account data.

The most common cause on Windows is a data folder the app cannot write to — the `.msi` installs into `Program Files`, where a standard user has no write access. From v2.1.2 the app falls back to `%APPDATA%\zeppbridge\ZeppBridge\data` when that happens (unless a database is already sitting in the blocked folder, in which case it says so rather than quietly starting empty). You can also point it anywhere with the `ZEPPBRIDGE_DATA_DIR` environment variable.

Is my data still there after uninstalling? Yes. Uninstalling leaves the `data` folder, backups, coverage ledger and settings alone. Delete it manually if you want it gone.

Can I back up and restore the database? Yes. Settings can create a whole-database snapshot at any time, each with a SHA-256 and an integrity check. Restores are queued and applied at the next launch — the only moment a file can be swapped atomically — and the queue step shows a record-count diff first. See backup and restore.

I have more than one watch — will the data get mixed together? No. Every record carries which device it came from, and the interface keeps them apart.

Does anything get sent to your servers? Health data, workout details and credentials never leave your machine. Only if you explicitly confirm "submit an error report" does the app send application/parser versions, OS, safe model hints and field structure for unrecognised products, firmware version, unknown workout codes with counts, and the numeric error code from the most recent request the Zepp cloud rejected (the number, which data stream, and when — never any text the cloud returned). It never sends accounts, tokens, serial numbers, device IDs, GPS, health values, raw responses or local paths. There is no automatic telemetry and no background crash reporting.

Privacy

- **Credentials** live in the OS credential store (Windows Credential Manager / macOS Keychain / Linux Secret Service), not in a plaintext file. On a machine with no keyring you can explicitly switch to a file or environment store — an acknowledged downgrade, documented in the Linux guide.
- **Health data** is an unencrypted database file on your computer. If you share the machine, use separate OS accounts.
- **AI packages are redacted first**: device identifiers, MAC addresses and precise GPS are stripped, and the file lists what was removed. Precise tracks are only included if you opt in.
- **Maps render locally.** No requests go to any third-party map service.
- **Error reports require explicit confirmation**, use a fixed allow-list, are built locally, need no GitHub account, and are never auto-published as issues.
- Syncing contacts Zepp's servers, so this is not a fully offline application.

See security and privacy. Report security issues through GitHub's private vulnerability reporting, not a public issue.

For developers

Tauri 2 + Vue 3 + Rust. The core lives in the `zeppbridge-core` crate; the desktop app, CLI, MCP server and local REST endpoint are all thin adapters over it — SQL, unit conversion and missing-value rules are never duplicated.

```
npm ci
npm run tauri dev
```

- Development — build gates, command contracts, local REST API, acceptance order
- Architecture — product boundaries, Zepp API mapping, verified vs unverified list
- CLI and MCP — exit-code contract, read-only tools, scheduling examples
- Backup and restore — snapshots, restore flow, coverage ledger
- Linux — Flatpak, deb/rpm/AppImage, data locations, credential stores
- Docker — headless CLI/MCP image, scheduling, reproducible builds
- UI guidelines — design tokens, page structure, components

Documentation is available in English and Simplified Chinese; every page links to its counterpart. Issues and PRs are welcome in either language. Before changing anything, read the "unverified" list in the architecture document — this project has an explicit standard for what counts as an established fact.

Acknowledgements

Zepp's API is undocumented; whether a data stream exists at all is only knowable from people who have already made it work. The API mapping draws on:

- m4ary/zepp-health-cli — event surface partitioning and field values
- Thejuampi/icu — an independent reproduction of the same APIs, useful as cross-validation
- H3llK33p3r/zepp-fit-extractor (Apache-2.0) — workout detail decoding

None of them are bundled; ZeppBridge draws on the API facts they recorded.

Licence