

TECHNICAL DOCUMENT

SPKR HQ In Depth

How SPKR HQ is built and how it works: the app, the stream studio engine, TALKIE, the Discord tab, the CPU pinning, the temperature helper, your data, updates, the server side, and how it is made and tested.

DESCRIBES	DATE	MADE BY	PLATFORM
SPKR HQ 2.1.0	September 2026	Sp3akR Software	Windows 11, x64

CONTENTS

1. Summary
2. The big picture
3. The desktop app
4. TALKIE: chat and voice
5. The stream studio engine
6. Going live
7. The DISCORD tab
8. GAMESX3D: pinning and performance
9. The temperature helper
10. Your data and settings
11. Installing and updating
12. The server side
13. How it is made and tested
14. Security and privacy by design
15. Parts made by others
16. Glossary

1 Summary

SPKR HQ is one Windows program that combines voice and text chat, streaming to friends, a full stream studio for Twitch, YouTube and Kick, Discord's own web app, and tools that keep games fast on AMD Ryzen processors. It is written in C# on Microsoft's .NET 10, draws its window with WPF, renders streams on the graphics card with Direct3D 11, and talks to its own server for accounts, TALKIE and streaming.

PART	WHAT IT IS BUILT WITH
Language and runtime	C# on .NET 10, compiled ahead of time (ReadyToRun) and shipped self-contained: no .NET install needed
Window and controls	WPF with WPF-UI (Fluent design), MVVM pattern with CommunityToolkit.Mvvm, ten colour themes
Web pages inside the app	Microsoft Edge WebView2 (TALKIE, DISCORD, web overlays of the studio)
Stream studio engine	Direct3D 11, Direct2D and DXGI (through Vortice.Windows), Windows Graphics Capture, Media Foundation, WASAPI audio
Encoding and sending	FFmpeg as a separate program: NVIDIA NVENC (H.264, HEVC, AV1), AMD AMF or x264; WebRTC (WHIP) and RTMP
Noise suppression	RNNoise by Xiph.Org, built from its C source into the program
CPU temperature	A small native helper (C# compiled to machine code with NativeAOT) and a C++ bridge to AMD's Ryzen Master Monitoring SDK
Settings	One SQLite database file, SPKRHQ.db; secrets encrypted with Windows' own data protection (DPAPI)
Installer and updates	Inno Setup 6, per user without admin rights; self-update with SHA-256 checked downloads
Server	Node.js services with SQLite, LiveKit (open-source WebRTC media server) for voice and streams, in Docker, behind Cloudflare
Size	One program file of about 250 MB, packed into an installer of about 76 MB

Code

About 30,000 lines of C#, 3,000 of XAML and 7,000 of JavaScript, plus HTML and CSS; 606 automated tests

2 The big picture

SPKR HQ is not one process doing everything. It is a few parts with one job each, so that a problem in one of them (a stuck graphics driver call, a crashing encoder) never takes the rest down. All of them come out of one installer.

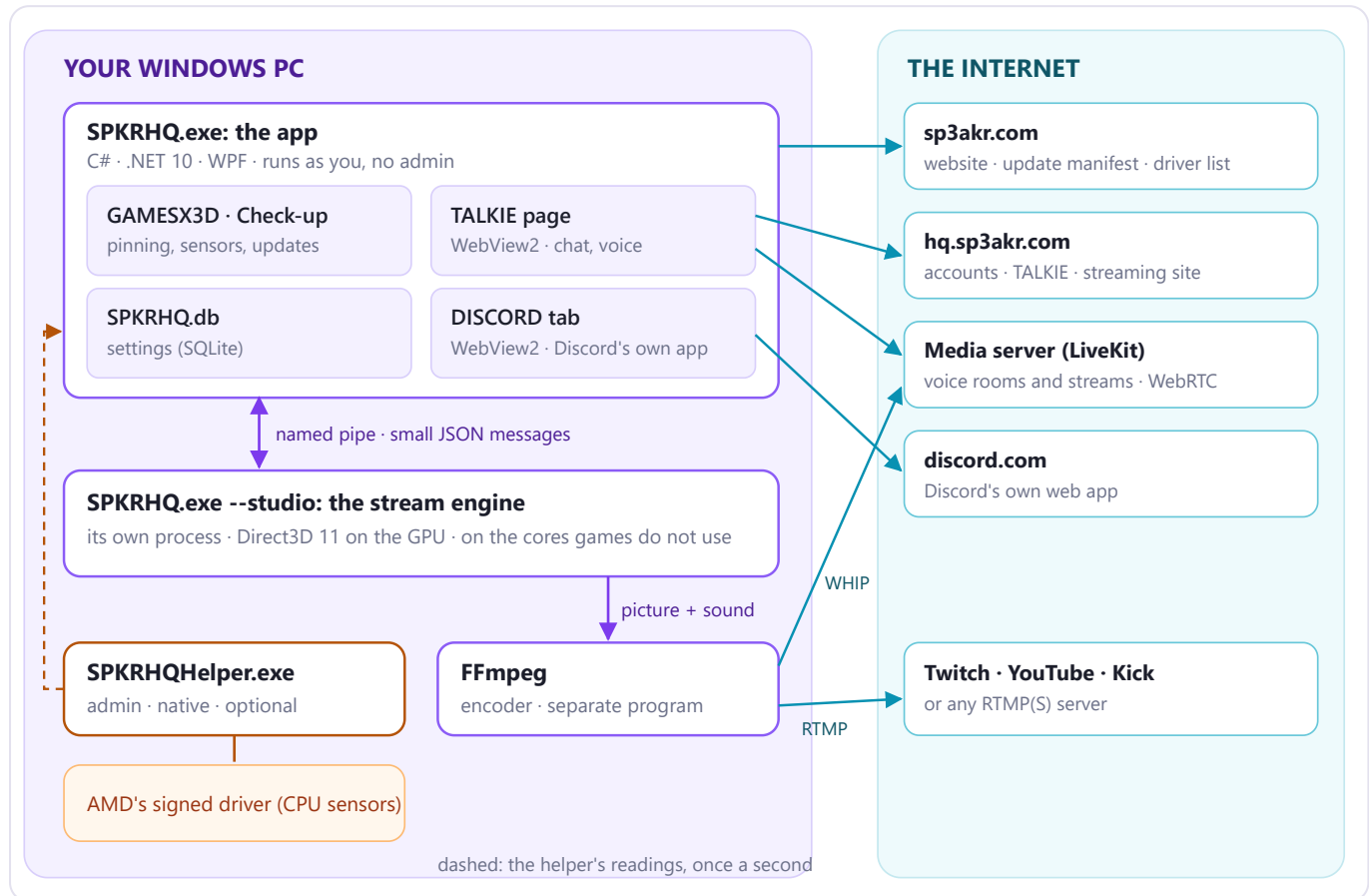


Figure 1. The parts of SPKR HQ and what they talk to. Everything on the left comes from one installer; the app is the only part you see.

The app

The window, the tray icon, TALKIE and DISCORD, the games list, the pinning, settings and updates. It runs as you, without admin rights.

The stream engine

The same program started a second time in its studio mode. It composes scenes on the graphics card and hands the finished picture and sound to FFmpeg.

FFmpeg

The well-known open-source encoder, downloaded once and run as a separate program. It encodes and sends the stream; SPKR HQ does everything before that itself.

The temperature helper

Optional. The only part that needs admin rights, because reading the CPU temperature needs a kernel driver. It lives in a folder only administrators can change.

3 The desktop app

One file, several jobs

Everything the PC runs comes out of one program file, `SPKRHQ.exe`. It is a self-contained .NET 10 program: the .NET runtime, WPF and every library are inside, and the code is compiled to machine code ahead of time (ReadyToRun) so it starts quickly. The file decides what it is from how it is started: without arguments it is the app, with `--studio` it is the stream engine, and a few test modes exist for the build (chapter 13).

The window

The window is WPF, Microsoft's desktop UI framework, styled with WPF-UI (Fluent design, the look of Windows 11) and SPKR HQ's own theme: ten colour themes change every colour of the window at once, the TALKIE page included. The code follows the MVVM pattern (views, view models and services apart), with CommunityToolkit.Mvvm. The top bar lives inside the window's title area, so it takes no extra room; pages open as cards over TALKIE, so a voice call never stops because you open Settings.

Starting

1. Only one SPKR HQ runs per Windows user. A second start (a shortcut, a link from the website) hands its request to the running one and closes.
2. A loading screen in your theme shows what is happening. The first start of a new version tidies the settings folder first (chapter 10).
3. The services start: the game list, the pinning timer, the sensors, the update check.
4. The window opens where you left it (screen, place and size are remembered). The loading screen closes when the first page is really drawn and TALKIE is signed in.

The log records how long each step takes. On the developer's PC a full start takes about five seconds, with the window visible after about four.

In the tray

With the window closed, SPKR HQ keeps running in the tray: games stay pinned, messages still arrive as Windows pop-ups, and an update can be put in. It starts with Windows if you want (a per-user startup entry, no admin).

4 TALKIE: chat and voice

TALKIE is a web application, served from `hq.sp3akr.com` and shown inside the app with WebView2 (the engine of Microsoft Edge). Being a web page means TALKIE can be improved on the server without an app update, while the app around it adds what a web page cannot do: global keys, Windows pop-ups, the game you are playing, streams from the studio.

How the app and the page talk

- They exchange small messages (WebView2's message channel). The app only listens to messages from exactly the TALKIE page; anything else is ignored.
- The app signs the page in with a short-lived session that it gets with your account. The page never sees your password or your stream key.
- The page can only ask the app for a small, fixed set of things, such as a fresh session or putting an invite code on the clipboard.
- The TALKIE browser has its own profile, the microphone is allowed for that page only, and developer tools and other permissions are switched off.

Voice

Voice rooms use WebRTC, the same real-time technology video calls in a browser use, through a LiveKit media server (an open-source SFU) run by Sp3akR. Everybody sends their voice once to the server, which forwards it to the others in the room. The browser engine provides echo cancelling, noise suppression and automatic volume; SPKR HQ adds a noise gate you set on a live meter, per-person volume, and mute and deafen keys that also work inside a full-screen game. Voice is never recorded.

Chat, friends and rooms

Messages, pictures, friends, servers and rooms are kept by the TALKIE service on the server (chapter 12). Private rooms are enforced by the server: people who are not allowed never receive the room, its messages or its voice. Limits keep it healthy: 25 people in a voice room, 50 in a server, four streams in one room.

5 The stream studio engine

The stream studio is SPKR HQ's own engine, not a copy of OBS. It runs as a separate process (the same program in its `--studio` mode), placed on the CPU cores games do not use. The app sends it small JSON messages over a named pipe (scene switches, layer positions, mixer levels) and gets back a live preview, meters and statistics. If the engine ever stops, the app restarts it, and the app itself keeps running.

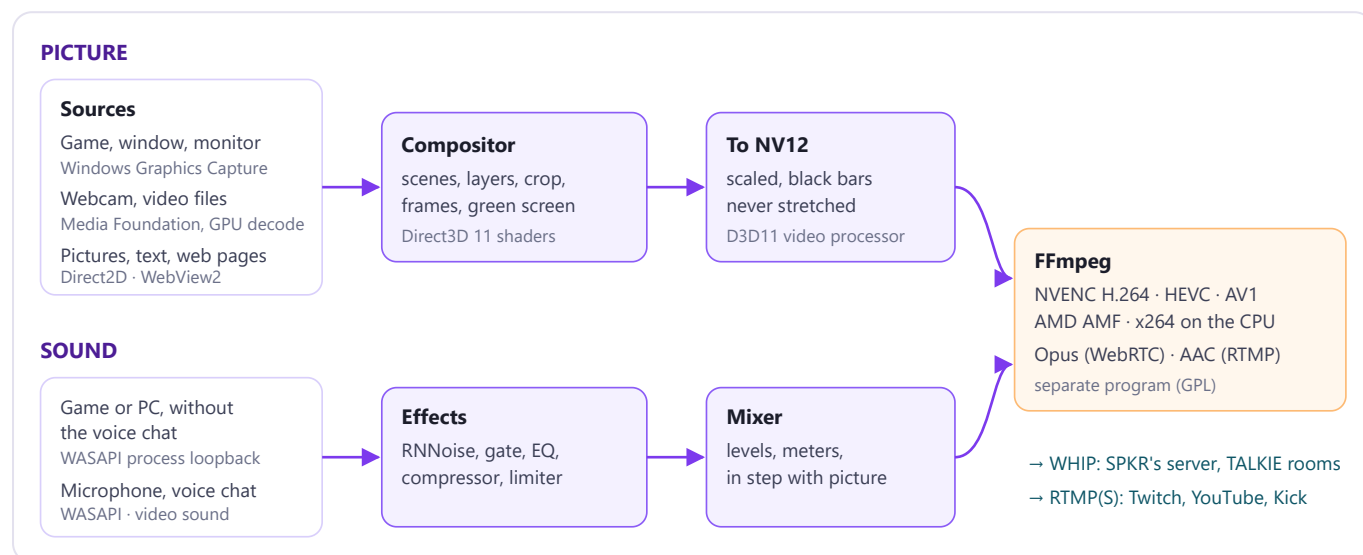


Figure 2. A frame's way through the engine. Everything left of FFMpeg happens inside SPKR HQ; the picture stays on the graphics card until it is converted for the encoder.

Picture

- **Capture.** Games, windows and monitors are captured with Windows Graphics Capture, Windows' own capture API: no yellow border, nothing injected into the game. "Game window only" keeps your desktop and notifications out of the stream.
- **Webcam and video.** Media Foundation opens the camera (best of 1080p or 720p, up to 60 fps) and decodes video files on the graphics card, with their sound.
- **Pictures, text, web pages.** Text is drawn with Direct2D. Web overlays (StreamElements, Streamlabs, or SPKR's own alerts and chat) are real web pages in an off-screen WebView2 whose picture is taken with its see-through parts kept.
- **Compositor.** Scenes of layers are drawn with Direct3D 11 shaders: position, crop, round and circle frames, borders, mirror, and a green screen key with similarity, smoothness and spill.
- **Output.** The GPU's video processor converts the canvas to NV12 (the format encoders take) and scales it to the stream size, adding black bars when the shape differs. Three read-back buffers let the CPU collect frame $n-2$ while the GPU draws frame n , and the output keeps its own clock: exactly the chosen number of frames per second go out, even when the network slows down.

Sound

- The sound of games or the whole PC comes from WASAPI process loopback, which can leave out SPKR HQ's own sound. Viewers hear the game, and a friend watching you does not hear their own voice a second later.
- The microphone and the voice chats have channels of their own. Every channel can run through RNNoise (a neural network noise remover by Xiph.Org, built from source into the program), a noise gate, an equalizer, a compressor and a limiter.
- The mixer keeps the sound in step with the picture. This is measured on every build: a test sends a flash and a beep every second through the real output path and FFmpeg, and fails when they drift more than 20 ms apart.

6 Going live

FFmpeg does the encoding and the sending. SPKR HQ downloads an unmodified build of it once, checks its size and SHA-256, and runs it as a separate program that gets the finished picture and sound. Keeping FFmpeg apart keeps its licence (GPL) apart from SPKR HQ, and lets it be updated on its own.

WHERE TO	HOW
TALKIE voice room	WebRTC (WHIP) into SPKR's media server; only the people in that room can watch, and the stream ends when you leave
Your channel on hq.sp3akr.com	WebRTC (WHIP); anyone can watch in a browser, others can co-stream next to you
Twitch	RTMP(S). Connecting uses Twitch's device sign-in (you type a code at twitch.tv/activate); SPKR HQ then fetches the stream key and sets the title and the game category
YouTube	RTMP(S) with your stream key; AV1 on graphics cards that can encode it, otherwise HEVC or H.264
Kick, Facebook, Trovo, others	Any RTMP or RTMPS address with its stream key

Every destination keeps its own resolution (720p to 4K, ultrawide 21:9 and 32:9, or your own size), frame rate, bitrate and constant or variable bitrate. The encoder is NVIDIA NVENC by default (the picture goes from the graphics card to its encoder, so the game keeps its frame rate), AMD AMF when there is no NVIDIA card, or x264 on exactly the CPU cores you tick. Whether a card can really encode HEVC or AV1 is tested by encoding a few black frames, not guessed from a list.

Alerts and chat on stream

SPKR alerts listen to Twitch through Twitch's official EventSub API over a WebSocket, only while a layer needs it: follows, subs, gifted subs, resubs, bits and raids, each with its own text, picture, sound and animation. The Twitch chat layer shows names in their colours with emotes, leaves commands and bots out, and removes lines a moderator deletes. Tokens are kept encrypted by Windows and never shown to a page.

7 The DISCORD tab

The DISCORD tab is Discord's own web app (discord.com) in a WebView2 of its own, with its own profile, the way a browser shows it. It is not a modified Discord client. You sign in on Discord's page; SPKR HQ never sees your password, your messages or your data.

- **Only hiding, never scripts.** SPKR HQ hides some parts of the page (download banner, Nitro, Shop, Quests, gift buttons) with plain CSS rules, and can recolour it on your screen. The list of rules is kept on sp3akr.com, so it can follow Discord's changes without an app update. No script is ever added to Discord's page.
- **Keys.** Mute, deafen, push-to-talk and navigation shortcuts work everywhere, also inside a game, by pressing Discord's own keyboard shortcuts in the page.
- **Notices.** Discord's web notifications are caught and shown as SPKR HQ pop-ups, with unread counts on the DISCORD button, the taskbar and the tray.
- **In the background.** Once opened on a PC, the tab can start quietly with SPKR HQ, so messages and calls come in without opening it, and a call keeps going on every page.

Discord's rules still apply. Discord's terms may count any change to how its app looks as a violation. The Legal page explains this; you can switch the tab off entirely in Settings.

8 GAMESX3D: pinning and performance

Why pinning

A Ryzen 9 7950X3D, 7900X3D, 9950X3D or 9900X3D has two dies (CCDs) of cores, and only one has the extra 3D V-Cache that makes games fast. AMD's driver only steers games it recognises. SPKR HQ reads the processor's layout from Windows (the cache topology per core, through `GetLogicalProcessorInformationEx`), finds the die with the bigger L3 cache itself, and keeps every game you add on that die.

How it pins

Pinning sets the game's process affinity, the same thing Task Manager's "Set affinity" does: nothing is injected into the game and no driver is involved. A timer checks every second, so a launcher, overlay or anti-cheat that resets it is corrected within moments. A game that refuses (one running as administrator) is shown as such and left alone. On a two-die Ryzen without V-Cache games are kept on the first die; on a single-die X3D the app says there is nothing to pin.

Finding games

The scanner reads the libraries of Steam (its library folders and app manifests), Epic, Xbox and the Microsoft Store, Ubisoft, EA and Rockstar, plus folders you add. It works out which exe is the game itself from the name, the Steam and engine files next to it, and by leaving servers, editors and crash handlers out. Pictures come from Steam, from the Steam or Epic store page, or from an Xbox game's own folder; play time from Steam's own records, or counted by the app.

Numbers and Check-up

Load per core and per die comes from Windows' performance counters, the graphics card from NVIDIA's NVML, and the CPU temperature from the helper (chapter 9). The Check-up page checks the Windows settings with a measured effect on games (Game Mode, GPU scheduling, Resizable BAR, memory speed, refresh rate, harmful boot tweaks, driver crashes in Windows' log). It offers Fix and Undo only where that is safe, and remembers the old value so Undo puts back exactly that. Driver and BIOS notices come from one small list on sp3akr.com that a server job refreshes daily; the app reads it at most twice a day and never downloads or installs a driver.

9 The temperature helper

Reading the CPU temperature of a Ryzen needs a kernel driver, and loading a driver needs administrator rights. So this one job lives in a small separate program, `SPKRHQHelper.exe`, and the app itself never needs admin rights.

- **Native code.** The helper is C# compiled to machine code with NativeAOT: about 4 MB, no .NET runtime, starts instantly. A small C++ bridge talks to AMD's Ryzen Master Monitoring SDK, which uses AMD's signed driver.
- **Installed once, protected.** Pressing Enable shows AMD's licence and asks Windows for permission once. The helper then copies itself into `C:\ProgramData\SPKR\SPKRHQ\helper`, a folder only administrators can change, and registers a task that starts it at sign-in with the highest rights. A per-user folder is never used for anything that runs as administrator.
- **One-way readings.** Once a second it writes the temperatures, clocks and power into a small file in your settings folder, which the app reads. The app can ask it to stop with a stop file, without admin rights.
- **Updated only when it changes.** The helper has its own version number. The app only asks you to press Repair when that version changes, not with every app update; the build refuses a changed helper whose version was not raised.
- **Uninstalling** removes the task, the helper and the driver service again.

10 Your data and settings

Two folders, apart on purpose: the program can be replaced by every update, your settings stay.

```
%LOCALAPPDATA%\SPKR\SPKRHQ\          the program: SPKRHQ.exe (+ its uninstaller)
%LOCALAPPDATA%\SPKR\SPKRHQSettings\    your settings
  SPKRHQ.db        all settings in one database
  Browser\         Talkie\  Discord\  Studio\  (sign-ins and caches of the web views)
  Media\           pictures and videos you added to the studio
  Cache\           game pictures, overlay pages, licences, FFmpeg
  Logs\            spkrhq.log
  Backup\          a copy of SPKRHQ.db from each of the last three days
C:\ProgramData\SPKR\SPKRHQ\helper\     the temperature helper (only when enabled)
```

SPKRHQ.db

The settings are one SQLite database with one entry per area: settings, games, stream, studio, Check-up, Discord servers, the driver list. SQLite writes every save completely or not at all, also when the PC loses power halfway through. Once a day a copy goes to Backup (the newest three are kept); if the database is ever damaged, SPKR HQ sets it aside and puts back the newest healthy copy by itself, and says so. Until version 2.0 the settings were one .json file per area; the first start of 2.1 moved them into the database after reading every entry back and comparing it, and kept the old files for 30 days.

Secrets

Stream keys and sign-in tokens (your SPKR HQ key, Twitch, YouTube, your own servers) are encrypted with Windows' Data Protection API for your Windows user before they are saved. Copied to another PC or user they cannot be read; there you simply sign in again. Your SPKR HQ password is never stored on the PC.

What leaves your PC

SPKR HQ has no telemetry, no analytics and no advertising. Only what a feature needs goes to a server, and only when you use it: your account and TALKIE content to hq.sp3akr.com, your stream while you stream, a feedback message when you send one, and the small update and driver lists it fetches. The full list is on the Legal page.

11 Installing and updating

The installer is made with Inno Setup 6. It installs for your Windows user only, into your own AppData folder, so it needs no administrator rights. It moves the data of older versions to their new place, shows a progress window in SPKR HQ's own style during an update, and keeps your settings when you uninstall.

1. Every five minutes, and when you open the window, the app reads a small manifest on sp3akr.com: the newest version, its file, size and SHA-256.
2. A newer version is downloaded in the background and checked: size and SHA-256 must match, or it is thrown away.
3. It never installs during a stream, a TALKIE voice room or a Discord call: then it waits, and the Install update button lets you pick the moment. Otherwise it goes in by itself.
4. The installer runs silently, the app closes for a few seconds and opens again on the page you were on.

The download on the website stays on one fixed version with a published SHA-256, so the file people download stays the same one; once installed, SPKR HQ updates itself to the newest version. The installer is not yet signed with a paid certificate, so Windows SmartScreen warns about the first download; the published SHA-256 lets you check you have the real file.

12 The server side

SPKR HQ's online parts run on servers of Sp3akR's own, not on a big cloud platform, each service in its own Docker container, behind Cloudflare.

SERVICE	WHAT IT DOES AND WHAT IT IS MADE OF
sp3akr.com	This website, the installers, the update manifest, the driver list and the Discord hide list. Static files served by nginx.
hq.sp3akr.com, accounts and site	Accounts with email confirmation and password reset by email, the streaming site, channels, the admin page and feedback. Node.js with Express and WebSockets, data in SQLite. Passwords are stored only as salted script hashes.
TALKIE service	Servers, rooms, messages, pictures, friends, invites and who is online. Node.js with WebSockets, data in SQLite.
Media server	LiveKit, an open-source WebRTC media server (SFU): TALKIE voice rooms, streams in voice rooms and on the site. Streams arrive by WHIP; a TURN relay helps networks that block direct connections.

Voice and streams pass through the media server live and are never recorded. The services are tested automatically too (their own test suites), and every change on the server is made with a backup of the previous version kept.

13 How it is made and tested

Who

SPKR HQ is made by one developer, Sp3akR, with an AI coding assistant (Claude, by Anthropic) as a partner. Sp3akR decides what gets built, how it should work and what is good enough; the assistant writes a large part of the code with him, and every version is built, tested and tried before it goes out.

How fast

The first line was written on 12 September 2026. Version 1.0.0, a small tool that kept games on the right CPU die, came out the next day. Ten days after the first line, and 72 versions later, 2.1.0 is a chat, voice, streaming and performance app. Short steps, each one tested and released, are how it grew so quickly without breaking.

Tools

.NET 10 SDK and C#, Visual Studio Build Tools (the Microsoft C++ compiler for the AMD bridge, RNNoise and NativeAOT), Inno Setup 6, Git, PowerShell for the build, and Node.js and Docker for the server side.

The build gate

One script builds a release, and it stops at the first thing that is not right:

1. **606 unit tests** of the logic: pinning, scanning, streaming settings, the settings database and its move, legal text, themes, and more.
2. **Engine tests on the real graphics card, through the finished program:** a rendered scene must match a reference picture; frames, mirror and green screen too; letterboxing; the noise remover must remove noise and keep speech; picture and sound must stay within 20 ms; video files must play, loop and end; web layers must keep their see-through parts; the alerts and chat pages must show and clear.
3. **The helper** must be native, carry AMD's files unchanged, and install correctly in a test folder.
4. **One file:** the publish folder may contain SPKRHQ.exe and nothing else.
5. **Smoke test:** the finished program must start and draw a page. This rule exists since one version once shipped that did not start.
6. **Click check:** every button of the top bar is pressed and must really open its page.
7. **Settings move:** the tidy-up of 2.1 runs on a made-up old folder and on a copy of a real one, and every setting must come out the same.
8. **Installer test:** a test build of the installer runs against a made-up older PC in a sandbox folder: data must move, old folders must go, uninstalling must keep the settings.

Only then is the installer made, its SHA-256 written into the update manifest, and the release published.

14 Security and privacy by design

- **Least rights.** The app runs as you. Only the optional helper runs as administrator, from a folder only administrators can change.
- **Checked updates.** Every download (update, FFmpeg) is checked for size and SHA-256 before it is used.
- **Encrypted secrets.** Keys and tokens are encrypted with Windows' Data Protection for your user; passwords are never stored on the PC, and on the server only as salted script hashes.
- **Locked-down web views.** Each web view has its own profile. TALKIE's page is checked by its exact address before the app listens to it. Studio web layers get no developer tools, no downloads, no new windows and no permissions.
- **Nothing injected.** Games are pinned from outside with Windows' own affinity setting and captured with Windows Graphics Capture; Discord's page only gets CSS.
- **No tracking.** No telemetry, analytics or ads. Voice and streams are never recorded. Data is never sold or shared.
- **One program file.** The program ships as a single file, which leaves fewer loose parts to tamper with.

15 Parts made by others

SPKR HQ stands on good open-source and vendor work. Each part keeps its own licence.

PART	USED FOR	LICENCE
.NET 10 and WPF	runtime and window	MIT
WPF-UI	Fluent controls	MIT
CommunityToolkit.Mvvm	MVVM	MIT
H.NotifyIcon	tray icon	MIT
Vortice.Windows	Direct3D 11, Direct2D, DXGI, Media Foundation from C#	MIT
Microsoft Edge WebView2	TALKIE, DISCORD, web layers	Microsoft
Microsoft.Data.Sqlite, SQLite	settings database	MIT, public domain
RNNoise (Xiph.Org)	noise suppression	BSD
FFmpeg	encoding and sending (separate program, downloaded once)	GPL
AMD Ryzen Master Monitoring SDK and driver	CPU temperature	AMD's licence
LiveKit	WebRTC media server and TALKIE's voice client	Apache 2.0
Node.js, Express, ws, better-sqlite3, nodemailer	server side	MIT and similar
Inno Setup	installer	Inno Setup licence

The licences that come with the program are in the app under Settings, LEGAL.

16 Glossary

3D V-Cache, CCD	AMD's stacked extra L3 cache on one die (CCD, core complex die) of an X3D processor; games run faster on that die.
Affinity	The set of CPU cores a program may run on. Setting it is a normal Windows feature.
DPAPI	Windows' Data Protection API: encrypts data so only the same Windows user on the same PC can read it.
FFmpeg	Open-source program that encodes, converts and sends audio and video.
MVVM	Model-View-ViewModel, a way to keep the screen apart from the logic behind it.
NativeAOT	Compiling C# to machine code ahead of time: a program that needs no .NET runtime.
NVENC, AMF, x264	The video encoders of NVIDIA and AMD graphics cards, and a well-known encoder that runs on the CPU.
NV12	The picture format video encoders take in (brightness at full size, colour at half size).
RTMP(S)	The protocol Twitch, YouTube and Kick receive streams with.
SFU	Selective Forwarding Unit: a media server that receives each stream once and forwards it to everyone who should get it.
SQLite	A small database engine that keeps a whole database in one file.
WASAPI	Windows' audio interface; "process loopback" captures what programs play, with the option to leave some out.
WebRTC, WHIP	Real-time audio and video as used by browsers; WHIP is the standard way to send a stream into a WebRTC server.
WebView2	The Microsoft Edge engine inside a Windows app, used to show web pages.
Windows Graphics Capture	Windows' own API to capture a window or screen on the graphics card.

WPF

Windows Presentation Foundation, Microsoft's framework for desktop windows.

SPKR HQ In Depth · describes version 2.1.0 · September 2026 · Sp3akR Software, the Netherlands ·

Sp3akR.v1@gmail.com. Names and trademarks belong to their owners; SPKR HQ is not made by, affiliated with or endorsed by Discord, Twitch, YouTube, Kick, AMD, NVIDIA, Microsoft, Valve, Epic Games or the makers of the parts above.