All tutorials

The dubbing studio, service by service: the walkthroughs, in order, showing what you will see on screen.

CONTENTS

Issued on 2026-09-08 for version 6.190.0.
The screenshots are taken inside the application, on a demonstration episode. A control moved in a later version reaches the online page before this document.

`https://axeho.studio`

Getting started with the voice / m&e separation

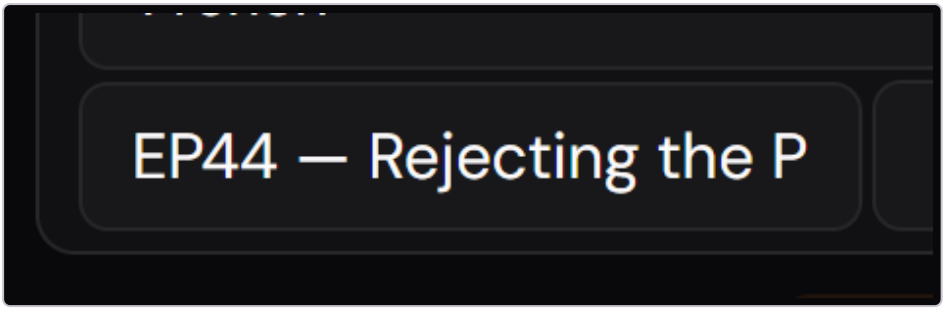
Get the M&E out of an episode delivered as a mix

The client sent a single file. You must dub over it without losing the music.

`A few minutes per episode, unattended`

01 **Name the project first**

The project name decides the episode folder. Without it, the elements land next to the sources and the final render will not find them on its own, Axeho tells you before starting, not after.



The project name field, in the top bar, filled in with the episode name.

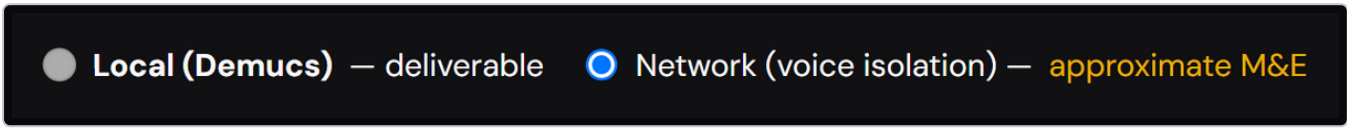
02 **Pick the sources**

Videos or audio files, one or fifty. The sound is extracted for you: no need to prepare WAV files.

The source picker buttons, videos or audio files, and the output folder.

03 **Choose the route**

The **local** route returns a deliverable M&E and needs no network. The **online** route is quicker to set up, but its M&E is a subtraction: good for a mock-up, never for broadcast. The produced file name carries the difference.



The two separation routes: local, deliverable, and online, whose M&E is approximate.

04 **Start it, and move on**

The work continues in the background. You can switch tabs, transcribe, write the band. A final report lists the files produced and any failures.

The Separate button, which starts the job in the background.

This tutorial online : `https://axeho.studio/en/tutorials-voice-music-separation.html`

Getting started with the rythmo band

CONTENTS

Align a rythmo band on an episode

You receive an episode and its script. Here is how to turn that into a band you can record from.

15 minutes

01 **Load the episode**

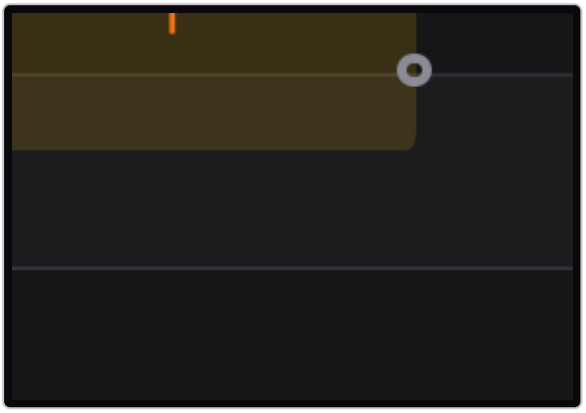
The band is built from the script's timecodes. **Script** imports an SRT, a DetX, an Excel or a Word file; **Vidéo** loads the film from disk, **URL** and **YT** from an address. Video is not required in order to write, but without it you get neither the waveform under the text nor playback.

The loading row: Script, Video, URL, YT, and adding characters and lines.

02 **Set frame rate and reference, before anything else**

i/s is the programme's frame rate: 25 for European television, 24 for cinema. **TC réf** is the timecode origin. 10:00:00:00 is the dubbing convention: that is why an episode starting at zero displays as ten hours.

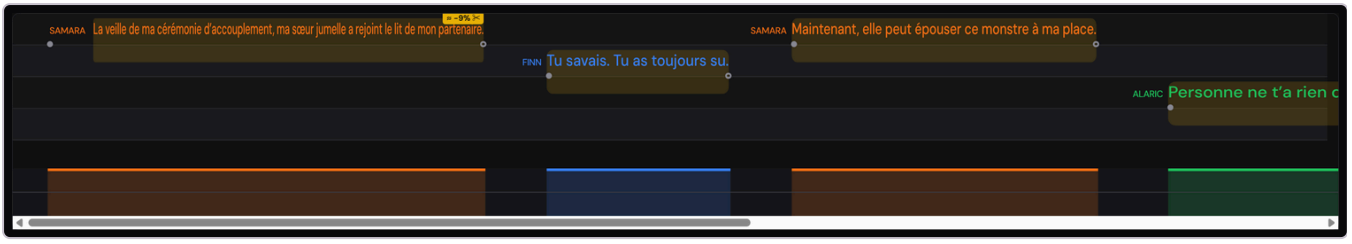
> Set both **before** aligning anything. Changing the frame rate afterwards redefines the length of a frame, and every alignment you made by hand drifts by that much.



The frame rate field (25 frames per second) and the timecode reference field (10:00:00:00).

03 Choose the reading density

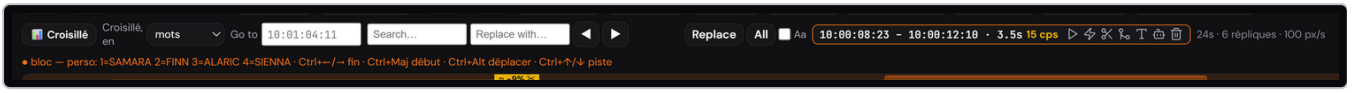
Three track heights. **Serrée** (tight) shows a whole scene's structure at a glance, that is the view below. **Large** (wide) reads from a distance, in the booth. The setting is remembered between sessions.



The band at tight density: four character tracks visible at once.

04 Select a block and read its state

Clicking a block reveals its bar: in and out timecodes, duration, then the **reading rate in characters per second**. The colour is a judgement, not decoration. **Green**: the text fits the window. **Yellow**: there is room left, like the 15 cps below. **Red**: the actor will not be able to say it. Thresholds depend on the block's length, because one speaks faster over three seconds than over half a second.

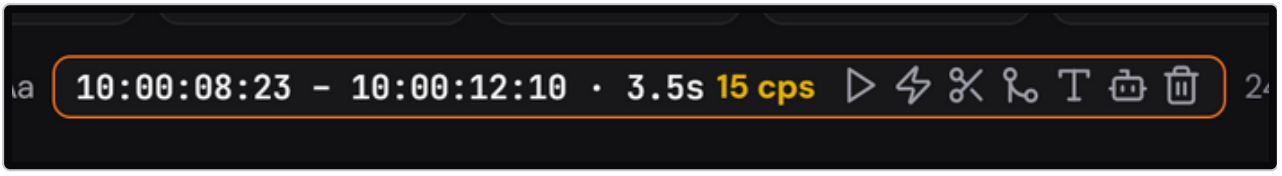


The selected block's bar: timecodes, duration, characters per second, and the alignment shortcuts.

05 The line's seven commands

The same bar carries everything you do to a line. ▷ plays it. ⚡ snaps it to the voice, using the word timecodes from the transcription. ✂ cuts at the cursor, ⑃ merges it with the next. **T** opens the labial suggestions, synonyms that keep the same mouth shape. 🤖 rewrites it to fit its window. 🗑 deletes it. Recording, on the other hand, lives in its own bar just above the band — always visible, whether a line is selected or not.

> Deleting lives **only** here. It used to sit on the block itself, in a five-pixel square: an irreversible action one misclick away, on every line of the episode. It now requires selecting the line first.



The block bar and its seven commands: play, align, split, merge, lip cues, rephrase, delete.

06 Align frame by frame, from the keyboard

The orange line lists the four gestures. **Ctrl+←/→** moves the block's **end**, **Ctrl+Shift+←/→** its **start**, **Ctrl+Alt+←/→** the **whole** block without changing its length, **Ctrl+↑/↓** moves it to another track. Each press is one frame, a twenty-fifth of a second at 25 fps. Keys **1** to **9** reassign the character without leaving the keyboard.

> The shortcut works even when the cursor is **inside the block's text**, the most common case, since one aligns while re-reading.

The shortcut line under the block bar: digits for characters, arrows for edges and tracks.

07 Fix the overruns

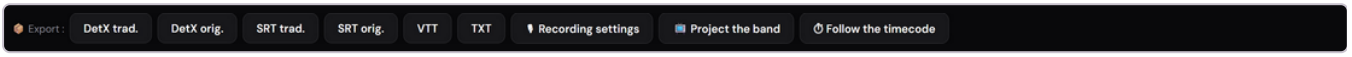
✂ Ajuster les débordements goes through the episode, finds lines too long for their window and rewrites them to fit, character, series bible and rhythm preserved. On a single orange badge, the same gesture handles one line only. **Ctrl+Z** undoes it.

> This command calls the AI, so it uses your own access and budget. Everything else on this page works offline.

Two blocks flagged as overflowing by an orange badge, one of them tinted red on the band.

08 Export the band

DetX is the booth format, in original or adapted version. **SRT** and **VTT** go to subtitling, **TXT** to proofreading on paper.



The export row: translated and original DetX, translated and original SRT, VTT, TXT.

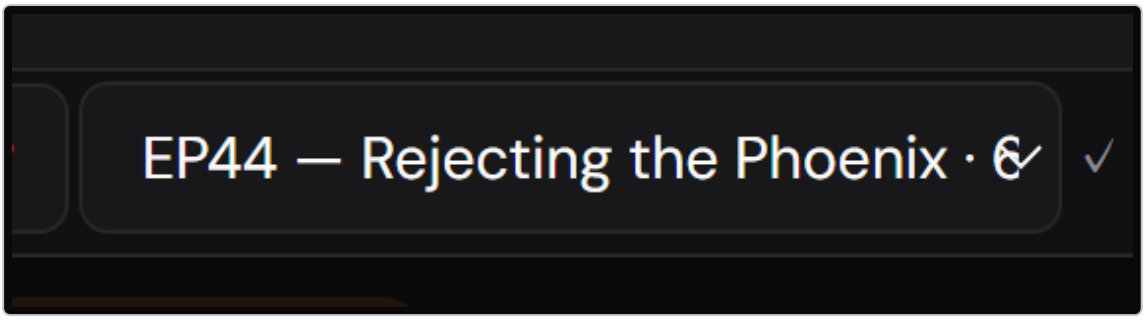
Record an actor in a home studio

An actor records from home, straight inside the application: their band filtered to their own role, with the microphone wired to the record button.

`5 minutes to get familiar, then the length of the episode being recorded`

01 Open the episode

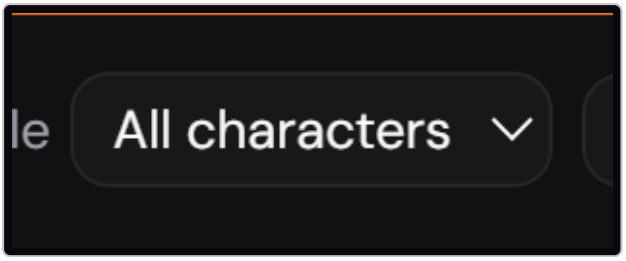
An actor does not need to set everything up themselves: if the adapter already prepared the project, **Ouvrir un projet** (open a project) in the **Fichier** menu is enough. Otherwise, **Vidéo** loads the film from disk (or **URL** / **YT** from an address), and **Script** imports the text — SRT, DetX, Excel or Word.



The menu that opens an already saved project.

02 Pick the role

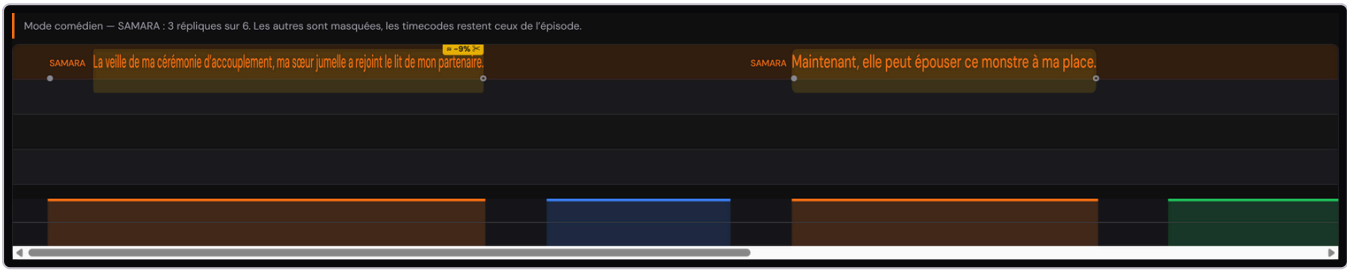
The **Rôle** menu lists the characters present in the episode. It is filled from the script: no prior setup is needed.



The Role menu, showing the episode's character list.

03 What the filter changes, and what it does not

The band now shows only the chosen character's lines, and a banner states which one. **The timecodes do not move.** That is deliberate: a band compressed onto one role's lines would no longer match the picture, and the recording would land off. So the actor sees the others' silences, those are their cues.



The band filtered to a single character: their lines alone, at the episode's timecodes.

04 Export the role sheet

📄 **Feuille de rôle** gives the character's lines in film order, timecoded, with the word count, useful when an actor is paid by the line.

The Role menu set to a character, and the Role sheet button.

05 Record the whole band in one go

🎬 **Enregistrer la bande** (record the band), in the dedicated bar above the band, starts the video and the microphone together: the video's own sound is muted automatically so it does not bleed into the take, and every line records at its timecode, with no need to stop at each one.

> This bar is **always visible** as soon as Rythmo opens. Buttons grey out rather than disappear, with a tooltip explaining why — two more discreet placements were tried before this one, and a real actor never noticed either of them.

The recording toolbar mid-session: Pause, Redo this block and Listen back to the band.

06 Pause without losing anything

⏸ **Pause** stops right where you are, without sending anything. During the pause, you can ▶ **Écouter** (listen to) a line already recorded or ↺ **Refaire ce bloc** (redo this block), which rewinds to the start of the line under the playhead and stops, ready to re-record. ▶ **Reprendre l'enregistrement** (resume) continues exactly where you left off.

The recording toolbar while paused: Resume, Redo this block and Finish and send now enabled.

07 Finish and send

✓ **Terminer et envoyer** (finish and send), available once paused, closes the session and delivers **one single file** for the whole band, every line placed at its timecode — not one take per line to stitch back together. The file is named after the character, the project and the take number.

> No accidental upload mid-session: **Pause** only pauses, this button alone delivers. So you can breathe, listen back, redo a block, then resume, without risking delivery of an incomplete band.

The Finish and send button, active once the band is paused.

Correct a word across the whole episode

A misspelled name, a series-bible term changed mid-season, a form of address to revise on one role only.

`2 minutes`

01 Search

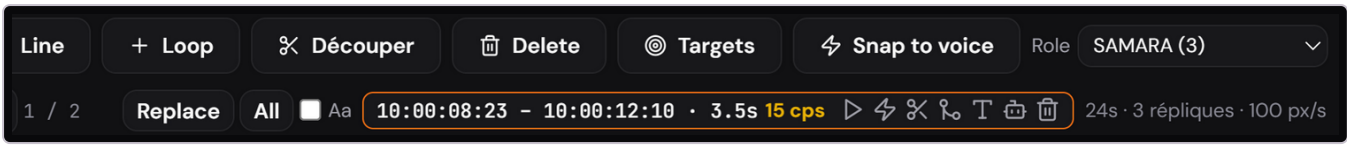
Type into **Rechercher…**. The counter shows position and total; ◀ ▶ move between occurrences, and **Enter** / **Shift+Enter** do the same without leaving the keyboard. **Aa** makes the search case-sensitive.

The Search and Replace fields, the occurrence counter and the case option.

02 Limit it to one character

The search **follows the Rôle filter**. If a character is selected, only their lines are scanned, including by **Tout** (all). That is what lets you revise one role's form of address without touching the others.

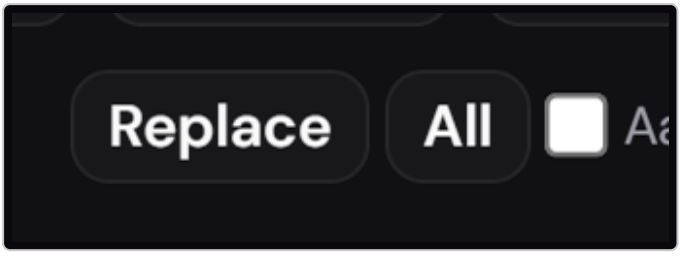
> It is also this tool's trap: a replacement that appears to have done nothing is almost always a **role filter left active**. Switch back to "all characters" before concluding.



The active Role filter next to the search fields: search only walks that character's lines.

03 Replace

Remplacer handles the current occurrence; **Tout** handles the whole episode, within the chosen role. A global replacement counts as **a single undo step**: **Ctrl+Z** reverses all of it.



The Replace and All buttons, and the match-case option.

This tutorial online : `https://axeho.studio/en/tutorials-rythmo-band.html`

Getting started with the detection and adaptation

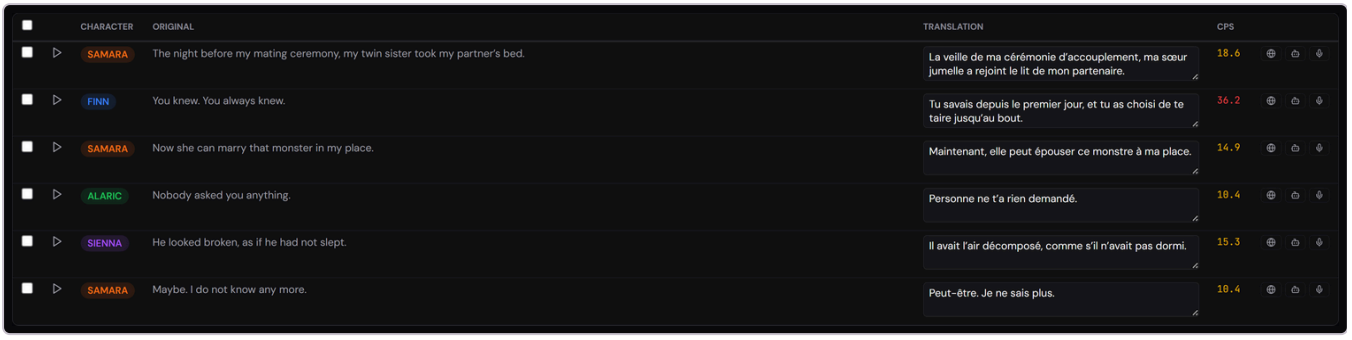
Adapt a line that fits the mouth

The adapter writing the target version, who wants to know while writing whether it will hold.

`10 minutes`

01 Write in the adaptation column

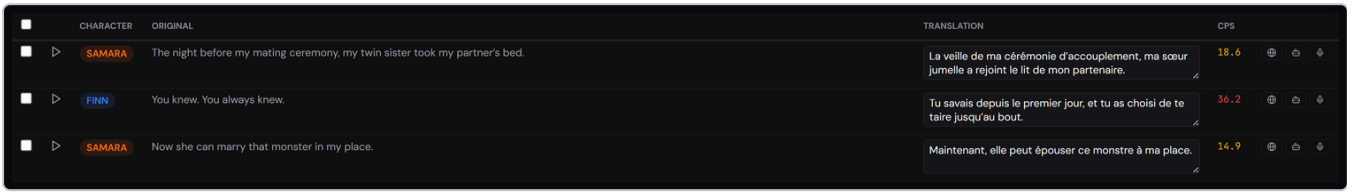
The original and the adaptation sit **side by side**, each with its timecode. You write while looking at the original, not from memory.



The Translation view: original and adaptation side by side, with each line's rate.

02 Read the rate while writing

The **characters per second** figure updates on every keystroke. **Green**: it fits. **Yellow**: there is room left. **Red**: the actor will not manage it. Thresholds depend on the line's length, one speaks faster over three seconds than over half a second.

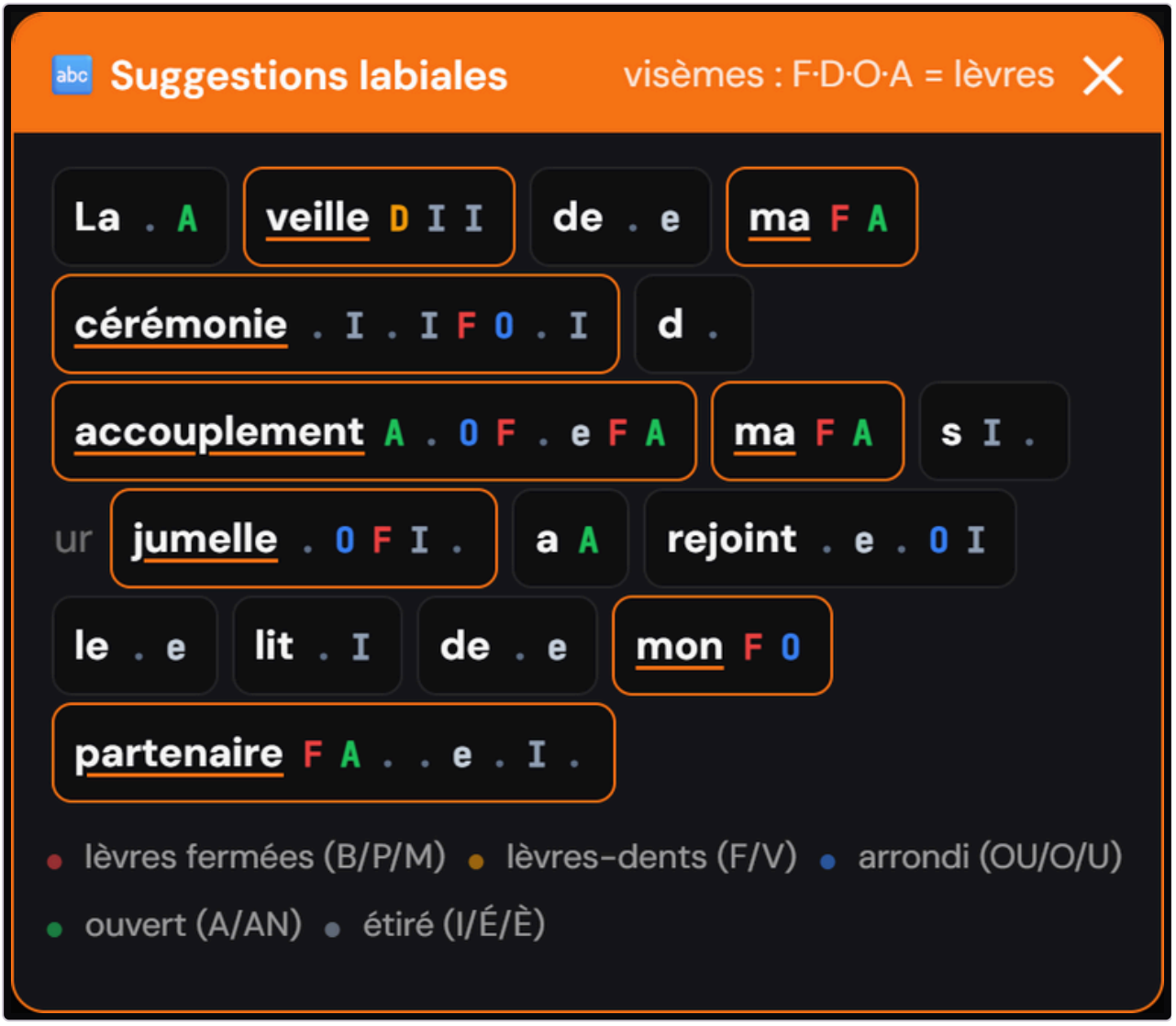


The CPS column: each line's rate, green, amber or red depending on its window.

03 **See the labials**

The consonants that close the mouth (**B, P, M**) and those that bite it (**F, V**) are marked on the text. That tells you whether your line lands where the actor's lips close.

> The 77,288-entry lexicon is **inside the application**. This step works in the booth, on a train, or on a programme that is not allowed to leave the studio.



The lip-cue window: each word with its visemes, and the mouth-shape legend.

04 **Ask for another word**

Double-click a word to get synonyms and expressions **filtered by viseme**: only those that keep the same mouth shape. That is where the tool saves time, hunting for a labial synonym by hand is the slowest gesture in the trade.

Synonyms proposed for a word, filtered to the same mouth shape.

This tutorial online : `https://axeho.studio/en/tutorials-lip-detection.html`

Getting started with the transcription

Transcribe an episode with no script

A programme arrives with no text, or with a script that no longer matches the cut.

`5 minutes of setup, then the wait`

01 **Point at the source**

Transcription works on the **audio**, not the picture. Paste the media address, or pick it under **Médias**. An audio-only file is enough, and faster than a whole video.

The address field for the audio to transcribe, and the local file button.

02 **Pick the language, or don't**

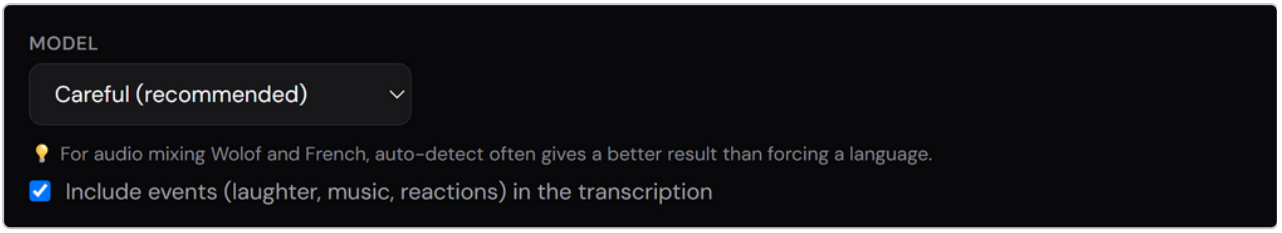
Auto-detect is the recommended setting, and genuinely so: on audio mixing two languages (Wolof and French in the same scene) it often beats forcing one.

> Forcing a language only helps when detection gets it wrong, which happens on very short programmes.

The Language menu, on automatic detection.

03 **Set the level and the events**

Careful is slower and more accurate; **Standard** is enough for a first pass. The **Include events** box adds laughter, music and reactions, useful in dubbing, where they take real space on the band.

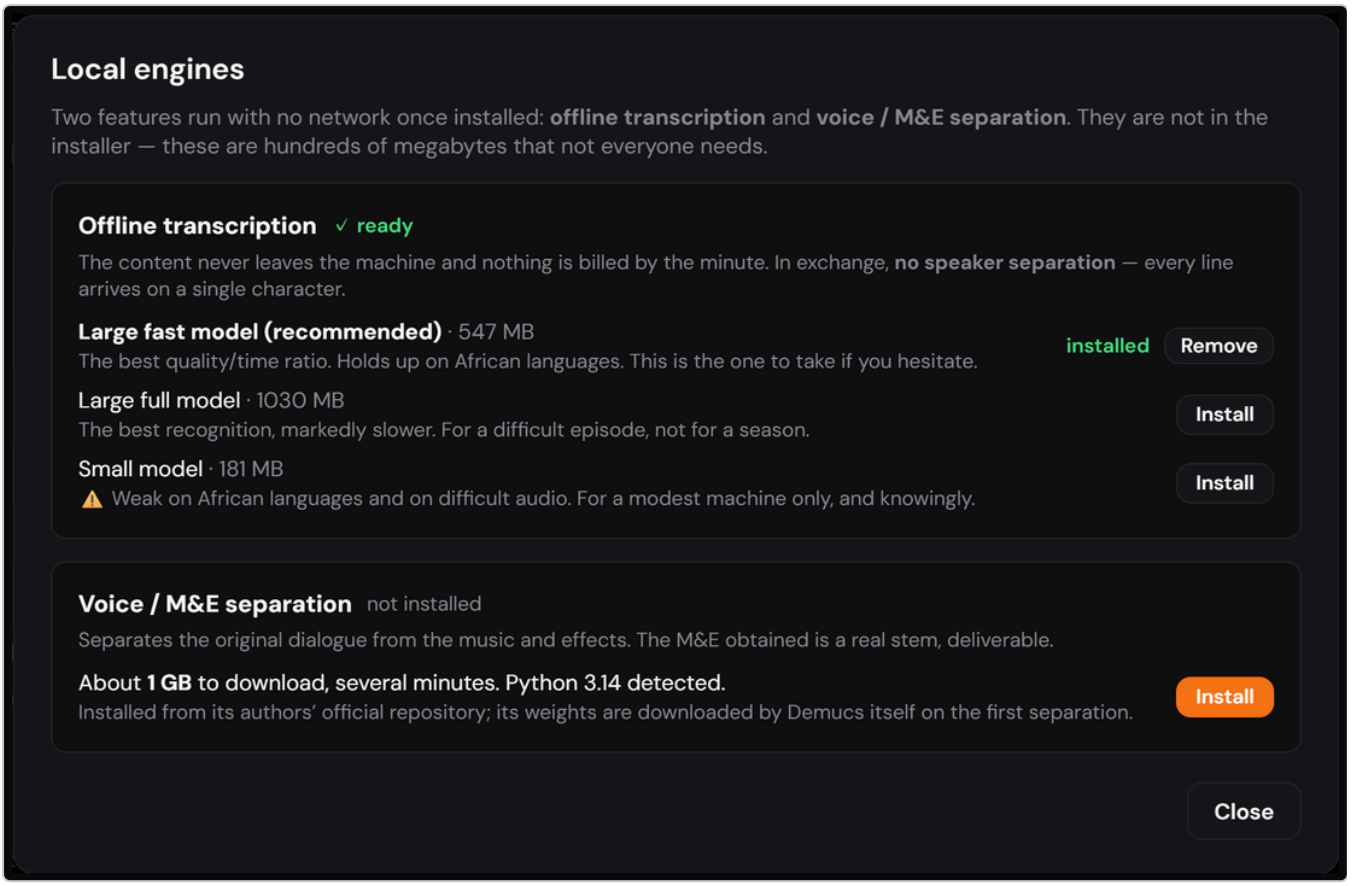


The model choice, Careful or Standard, and the events checkbox.

04 Work with no network

An **offline engine** installs from **Moteurs locaux**, in one click. The content no longer leaves the machine and nothing is billed by the minute: that is what an embargoed programme, a confidentiality clause, or a booth with no connection calls for.

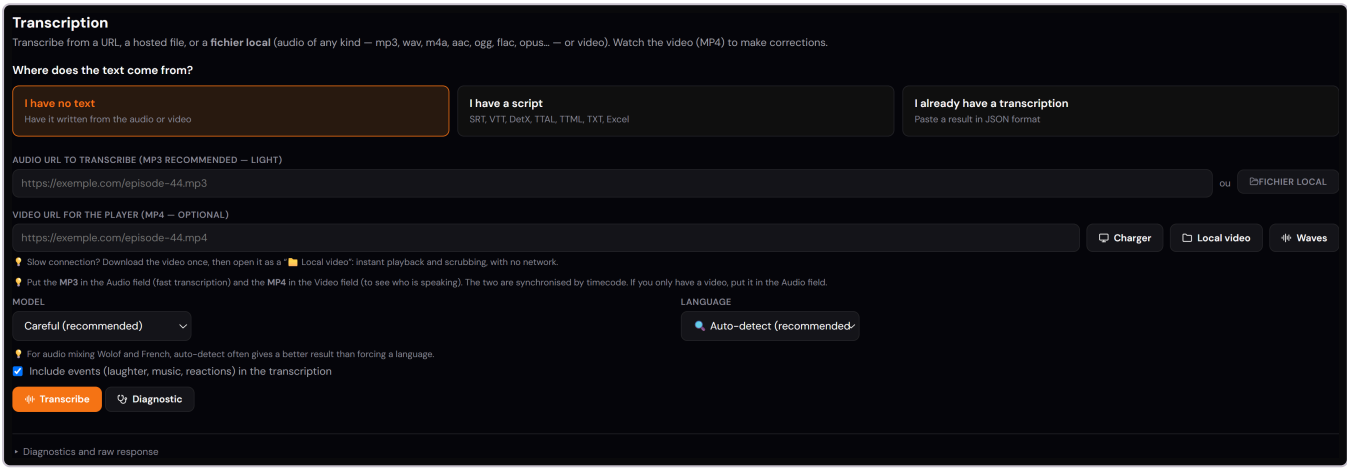
> In exchange, **no speaker separation**: every line arrives on a single character, to be split afterwards. And it is the machine's processor doing the work — expect longer than the programme's own duration.



The Moteurs locaux window: offline transcription and voice / M&E separation, to install on the machine.

05 Start it, and let it run

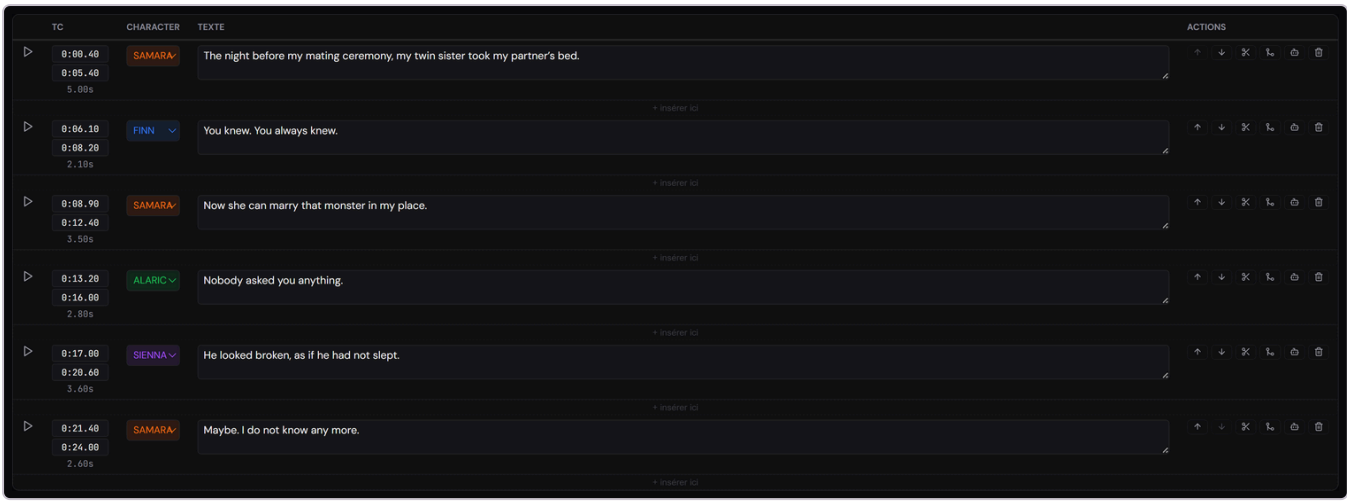
The screen names the current phase and the elapsed time, and estimates what remains from your previous runs. You can interrupt at any point.



The transcription screen: source, language and fidelity before starting the job.

06 **Collect the text**

The lines arrive **timecoded** in the script, ready to adapt. That is the starting point for everything else: detection, the band, the voices.



The transcribed lines, timecoded and assigned to their character, in the script editor.

This tutorial online : `https://axeho.studio/en/tutorials-transcription.html`

Getting started with the conforming

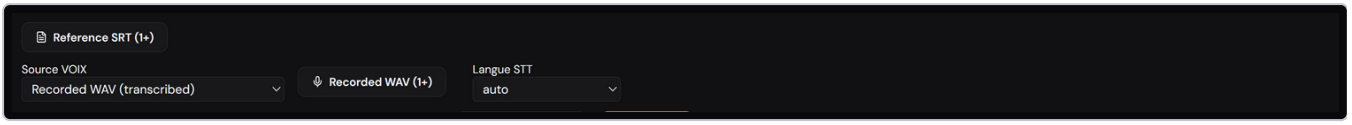
Conform after the studio comes back

What was recorded no longer matches the written text: the floor changed lines.

`10 minutes per episode`

01 **Bring the two versions together**

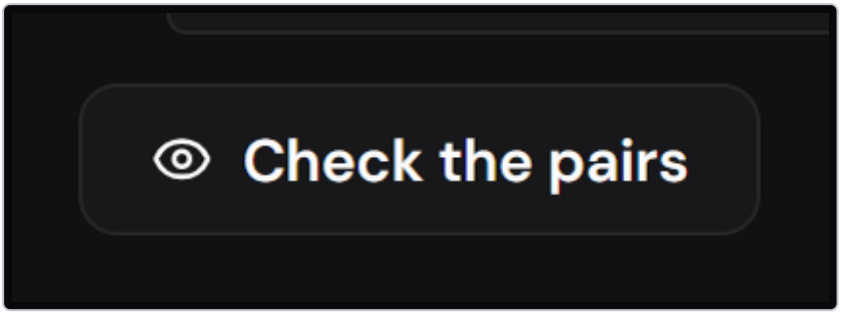
On one side the **reference SRT**, the original text. On the other, the **recorded WAV files** from the studio. Several can be loaded at once.



Picking the reference SRT, the VOICE source and the transcription language.

02 **Let it pair them**

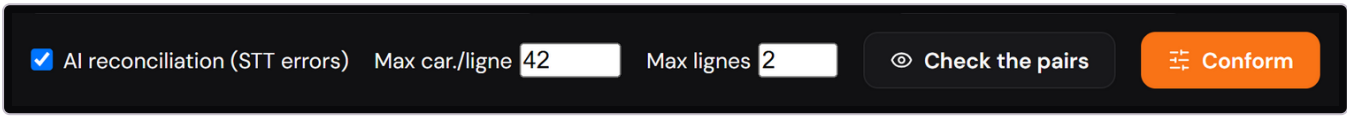
The tool matches each recording to its reference. That is the job nobody wants to do by hand across forty lines.



The Check pairs button, which shows the matching before the batch starts.

03 **Run it and re-read**

Conforming transfers by **timecode** and flags the gaps. Those gaps are what to look at: the rest is already correct.



The formatting rules, the reconciliation option and the Conform button.

04 **Collect the result**

The **voice SRT** and the **final DetX** come out matching what was actually said, ready for delivery.

The result header and the download button for the conformed SRT.

This tutorial online : `https://axeho.studio/en/tutorials-conforming.html`

Getting started with series

Keep a bible across a whole series

Whoever runs a long series, where the same names and choices must come back identical.

`15 minutes to set up, then a few minutes per episode`

01 **Load the episodes**

Choose the .srt files takes several episodes at once. The dashboard then shows the **status of each episode**: what is adapted and what is not.

Picking the series' .srt files, all in one go.

02 **Have the story analysed**

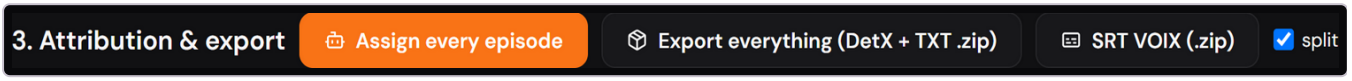
Analyse the story collects proper nouns, recurring terms and adaptation choices, and files them in the series bible.

> This step calls the AI, so it uses your own access and budget.

The Analyse the story button and the number of episodes used as a sample.

03 **Propagate across the series**

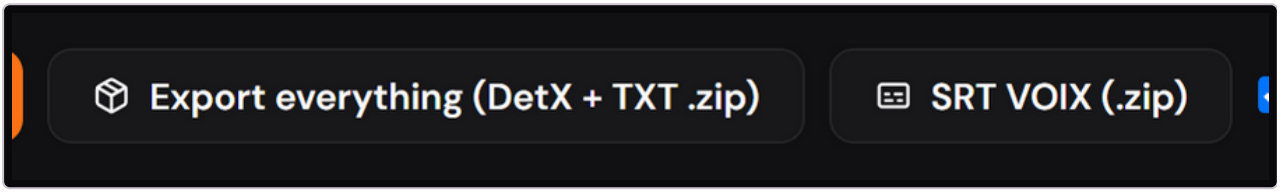
Assign all episodes applies the bible to the whole run. That is what stops a character being addressed informally in episode 12 and formally in episode 40.



The Assign all episodes button, next to the series exports.

04 **Export the batch**

Export everything produces a .zip with the series' DetX and TXT files. **SRT VOIX** exports the batch meant for recording.



The batch exports: DetX and TXT as a .zip, and the VOICE SRTs.

This tutorial online : `https://axeho.studio/en/tutorials-series.html`

Getting started with voices

CONTENTS

Cast the voices and generate an episode

The dubbing director who wants to hear a mock-up before calling the actors in.

20 minutes

01 Cast the project

Load the voices pulls the list available on your account, then you assign one voice per character. The casting belongs to the project: it carries across episodes.

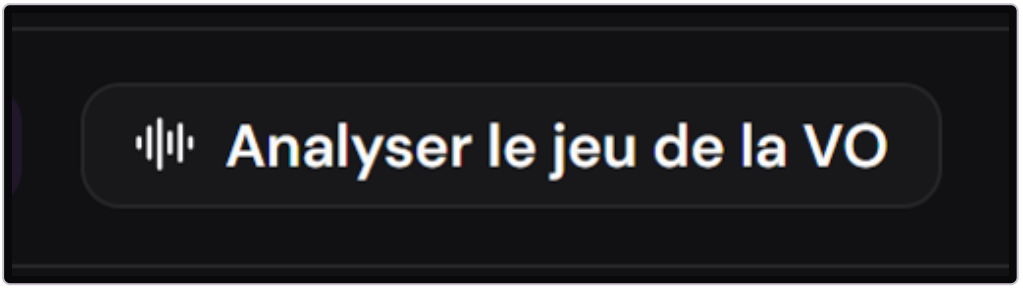
> You connect **your own credentials**. Costs, quotas and voice rights stay yours, which is also what leaves you in control of the provider.

The project casting: one voice per character, with each one's line count.

02 Listen to the original performance

Analyse the original performance measures, line by line, the intensity, pitch, rate and intonation of the original actor. Those measurements are then attached to the tag request: the AI stops guessing the emotion from the meaning of the words alone.

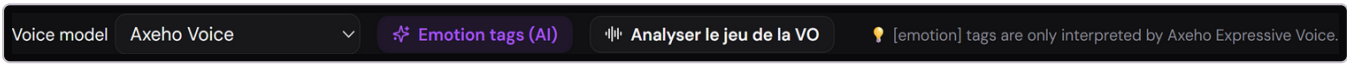
> The analysis is **local**, with no network and no cost. Each line is compared to the average **of the same character**: a bass voice is not "low", it is normal for that actor.



The button that analyses the original performance.

03 Set the performance intent

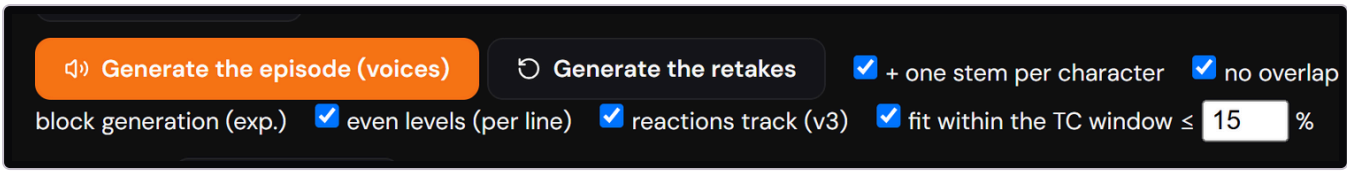
Emotion tags suggests performance markers line by line. You can also insert them by hand wherever the reading has to change.



The emotion tags button and the analysis of the original performance.

04 **Generate the episode**

Generation produces clips **aligned to the project's timecodes**, plus separated stems. Each run is dated and kept: you can compare two versions instead of overwriting the last one.

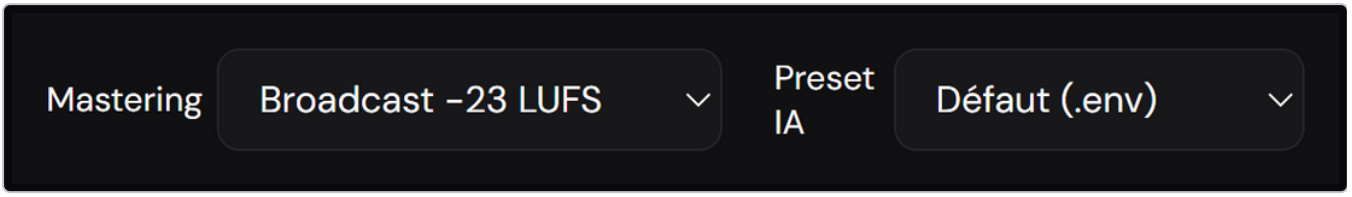


The episode generation button and the track options.

05 **Master for the broadcaster**

Four loudness targets: **broadcast −23**, **streaming −27**, **web −16**, **micro-drama −15 LUFS**. It applies to the full mix and to every stem, and the broadcaster no longer has to reject the delivery over a level that is not theirs.

> The mastered file is written **next to** the raw one, never over it: you compare the two, and keep the one you send.



The mastering target and processing preset menus.

Adjust the lips on screen

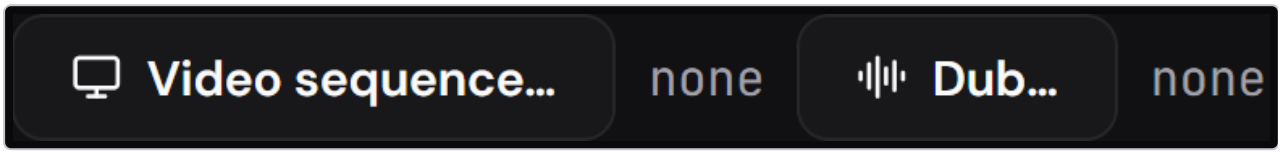
An advert, a corporate film, a training module: the picture has to match the dub.

`5 minutes of setup, then the processing`

01 **Point at the sequence and the dub**

A **video sequence** and the **dub** that goes with it. The feature works on a shot, not on a whole episode.

> **Not for fiction.** The actor's performance is the product; distorting it damages the very thing you are paid to deliver. Axeho's reference sync remains the one in the text, words chosen to land where the mouth closes.



Picking the video sequence and the dub file.

02 **Process only the part that matters**

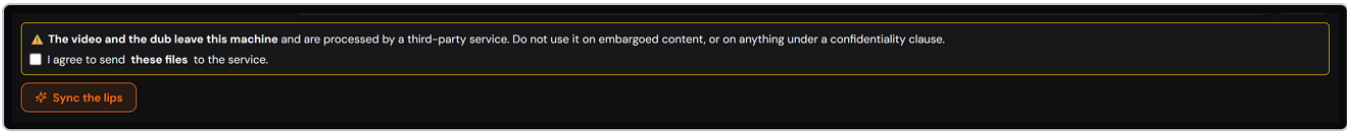
Bound the passage in seconds. Less data sent, so cheaper, and a better result: the service works on what matters instead of crossing the whole shot.

The start and end bounds of the excerpt to process, in seconds.

03 **Accept the upload, then start**

The video and the dub leave this machine and are processed by a third-party service. The box is ticked each time, for **those files**: it is a decision you take, not a setting you forget.

> Everything else on this page works on the machine. This option is the only one that leaves, which is why it asks for explicit consent.



The third-party upload warning, its consent box and the start button.

This tutorial online : `https://axeho.studio/en/tutorials-voices.html`

Getting started with subtitles and delivery

CONTENTS

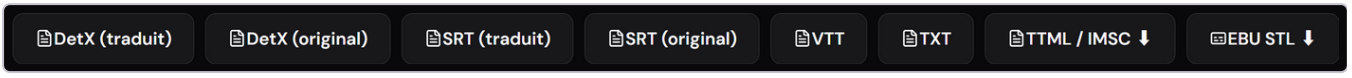
Export what the client expects

The end of the chain: every recipient expects a different format.

2 minutes

01 **Pick the right format**

DetX goes to the booth. **SRT** and **VTT** go to subtitling. **TXT** is for proofreading on paper, still the fastest medium for client sign-off.

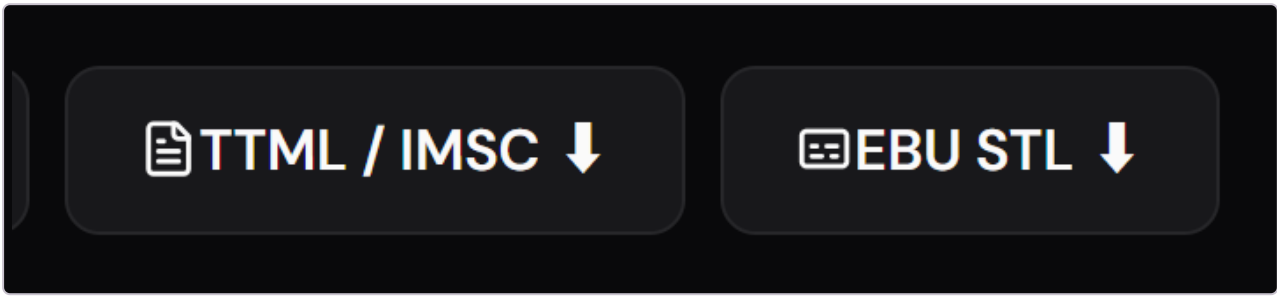


The export formats: DetX, SRT, VTT, TXT, TTML and EBU STL.

02 **Export in the broadcasters' formats**

EBU STL for European channels, with the French teletext conventions — an export that went through CANAL+'s own check. **TTML** for the platforms, switching to **IMSC** when the language requires it.

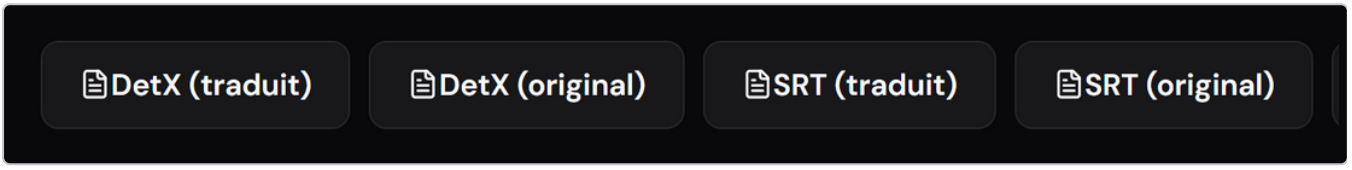
> Duration and reading-rate constraints are **checked and reported**, never silently fixed: a subtitle that runs too fast shows up in the report, and you decide what gets cut.



The TTML / IMSC and EBU STL exports, next to the everyday formats.

03 **Original or adapted**

DetX and SRT export in **both versions**. Clients often ask for both: one to check against, one to work from.



DetX and SRT, each in an adapted and an original version.

04 Export a single role's band

For an actor in a home studio, the **role sheet** gives their lines in film order, timecoded, with the word count, useful when they are paid by the line.

The Role menu and the Role sheet button, in the rythmo band.

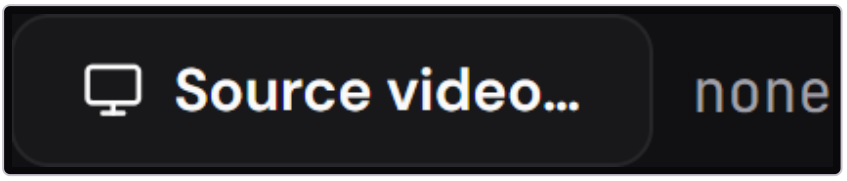
Render the dubbed video

The client wants to watch the episode, not open a DetX.

`A few seconds per episode`

01 Choose the picture

The **source video** is the one from the edit. It is **not re-encoded**: the video stream comes out identical to what went in, which is why an episode renders in seconds rather than half an hour.

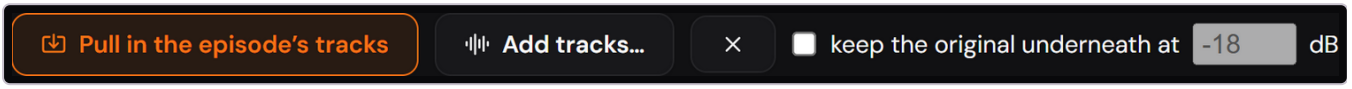


The button that picks the render's source video.

02 Pull in the episode's tracks

Pull in the episode's tracks fetches the voice from the last generation and the separated M&E, where they were filed. Everything stays editable, and you can add tracks by hand.

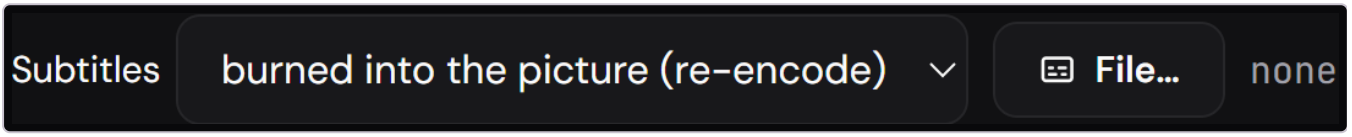
> **Keep the original underneath** is for checking, never for delivery: you then hear both versions at once, which is exactly what a broadcaster rejects.



Pulling in the episode's tracks, adding tracks by hand, and the option to keep the original underneath.

03 Subtitles: burned in or embedded

Embedded track leaves the picture untouched and can be switched off in the player. **Burned into the picture** re-encodes, so it takes longer, but the text shows up everywhere — including in editing software, which ignores text tracks.



The render's subtitle menu: none, embedded track, or burned into the picture.

04 **Render**

A name, a container, an output folder. The file produced is the one you show the client — and it is also the one to run through **quality control** before sending it.

The button that starts the dubbed video render.

This tutorial online : `https://axeho.studio/en/tutorials-delivery.html`

Getting started with quality control

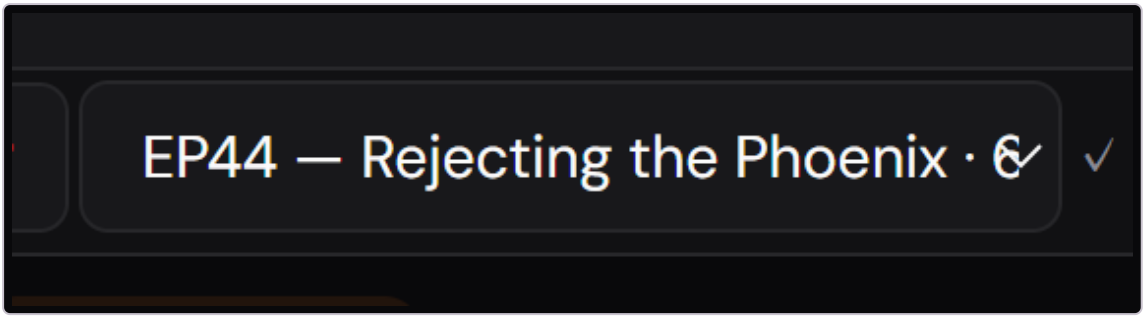
Check an episode before sending it

The file is ready. You want to know what the client's QC will find, before they do.

`2 minutes per episode`

01 Open the episode's project

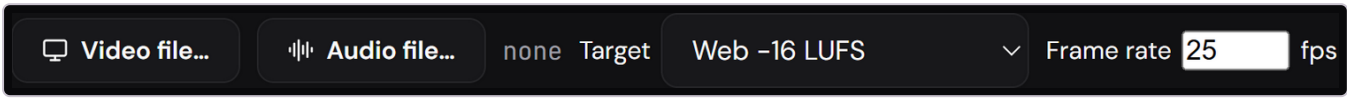
The script is the reference: without it you can measure levels, but neither completeness nor synchronisation. Axeho says so rather than silently skipping the checks.



The menu that opens an already saved project.

02 Point at the deliverable

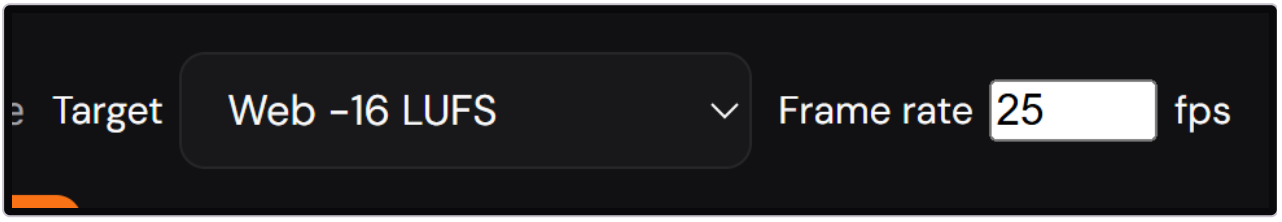
The video render, the mix, or a disputed stem. The check accepts picture as readily as audio, the video file is precisely what nothing used to verify.



Choosing the file to check, video or audio.

03 Choose the target and the frame rate

The loudness target depends on the broadcaster. The frame rate expresses the sync tolerance in **frames**: that is the unit of the trade, a millisecond means nothing to anyone.



The loudness target and the reference frame rate.

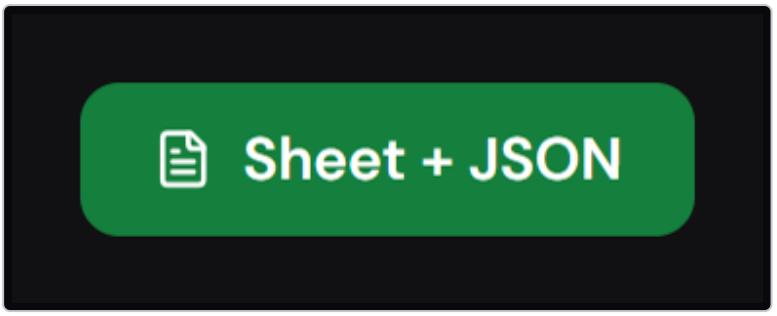
04 Run everything

One button chains the three checks then issues the sheet. Missing lines and the most-offset lines are clickable: you go and listen to them without hunting.

The button that chains the three checks and produces the report.

05 **Attach the sheet**

The sheet ships with the delivery. It depends on no software: it is a document that opens anywhere and prints to PDF.



The Sheet + JSON button, which produces the document shipped with the delivery.

This tutorial online : `https://axeho.studio/en/tutorials-quality-control.html`

Getting started with project management

CONTENTS

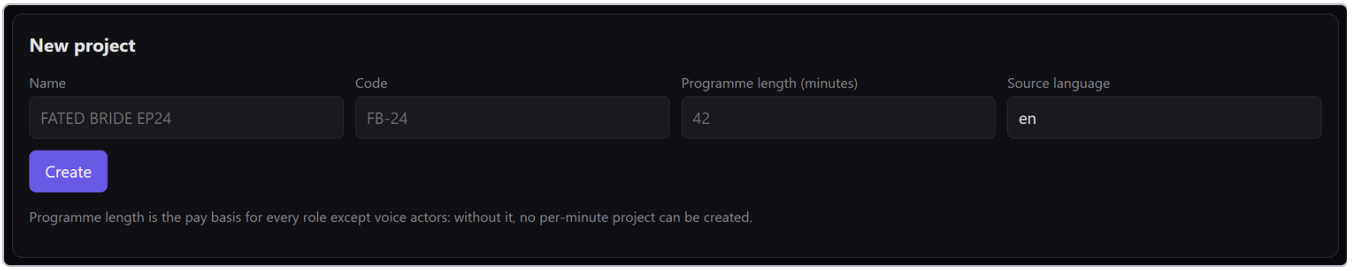
Create a project and assign it

An episode comes in. It has to exist in the console before it can be handed out.

5 minutes

01 Create the project

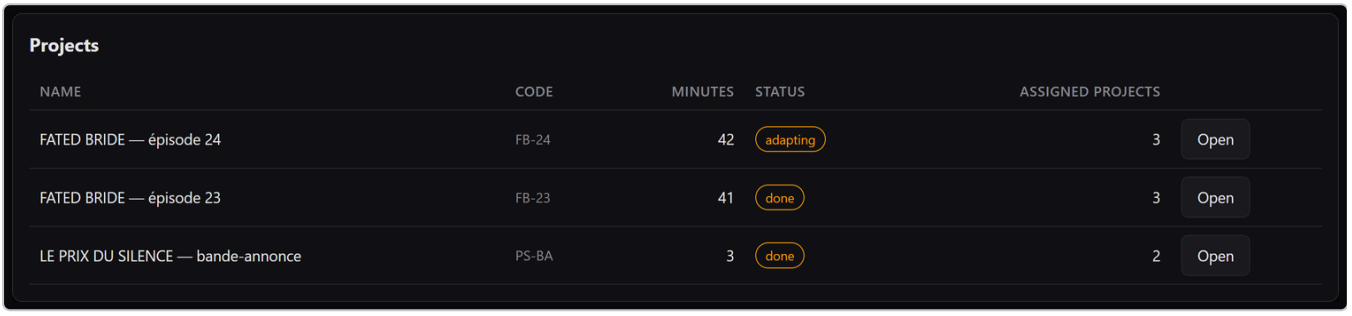
A name, a code, and above all the **programme duration in minutes**. That is the pay base for every role **except voice actors**, who are paid by the line: without it, no per-minute assignment can be made.



The creation form: name, code, programme duration and source language.

02 Find the project in the list

The list shows each project's state and how many **assigned projects** hang off it. **Open** unfolds the sheet: that is where everything happens.

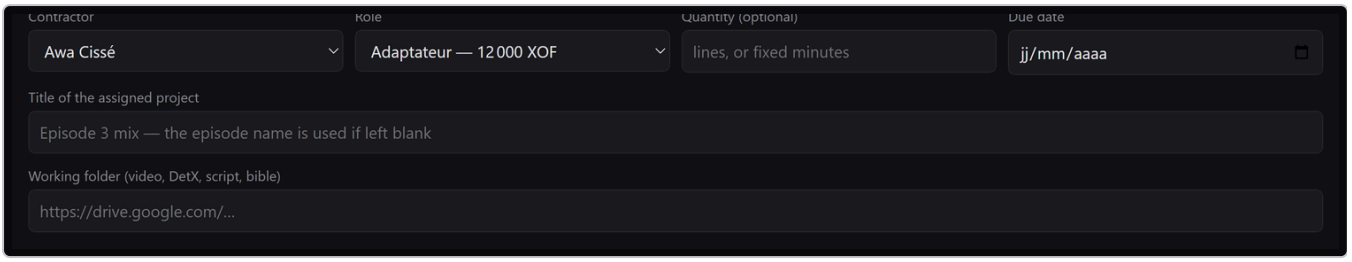


The project list: minutes, state, number of assignments.

03 Hand out the work

A contractor, a **role**, a due date, and the working folder — video, DetX, script, series bible. The amount comes from the role's rate and the programme duration; **quantity** is only entered for a line count or imposed minutes.

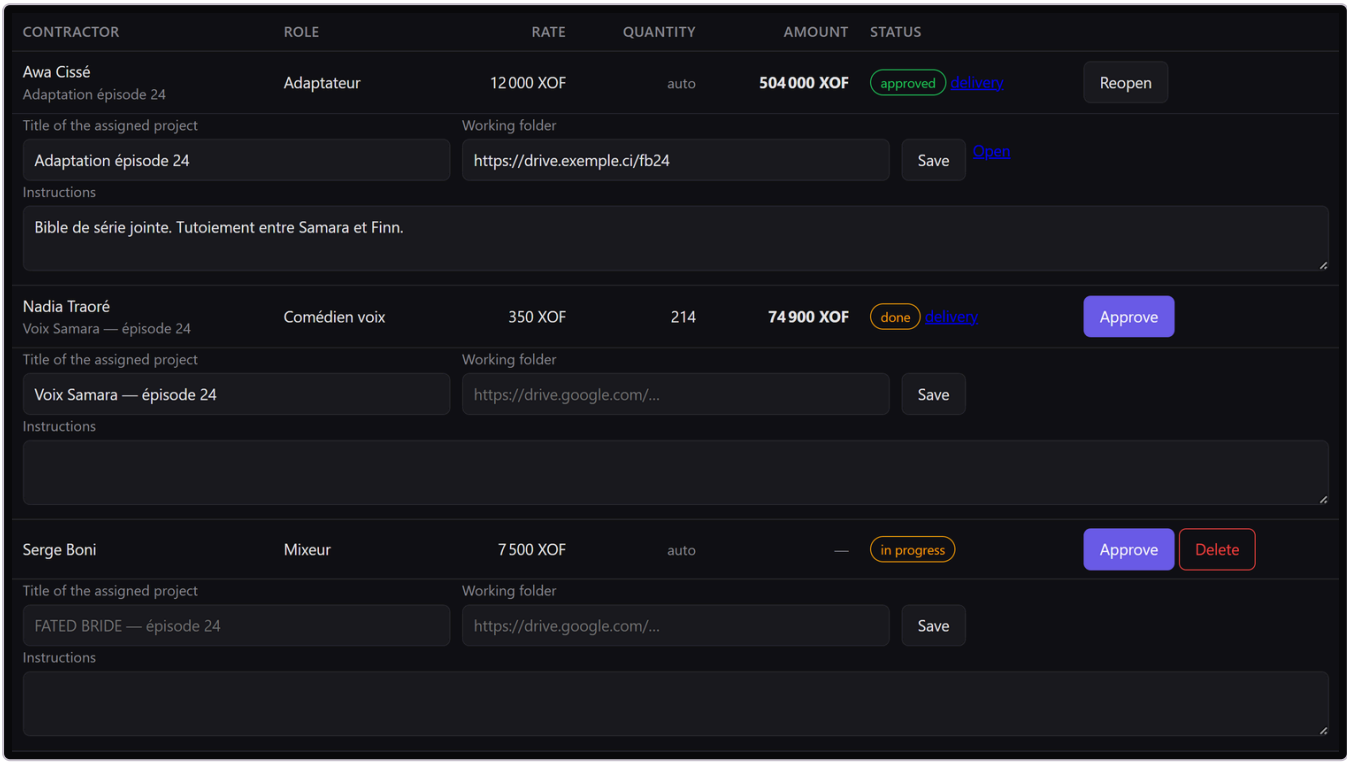
> The contractor's usual role only **pre-fills** the field. The assignment's role is what decides the pay: an adapter lending their voice on one episode is paid by the line there, not by the minute.



The assignment form: contractor, role, quantity, deadline and working folder.

04 Track what has been handed out

Each row carries its rate, quantity, amount and state. The **title** and the **instructions** stay editable until the work is approved: that is what the contractor sees on their side.



The assignment table: three contractors, three states, and each one's instruction sheet.

Invite a contractor

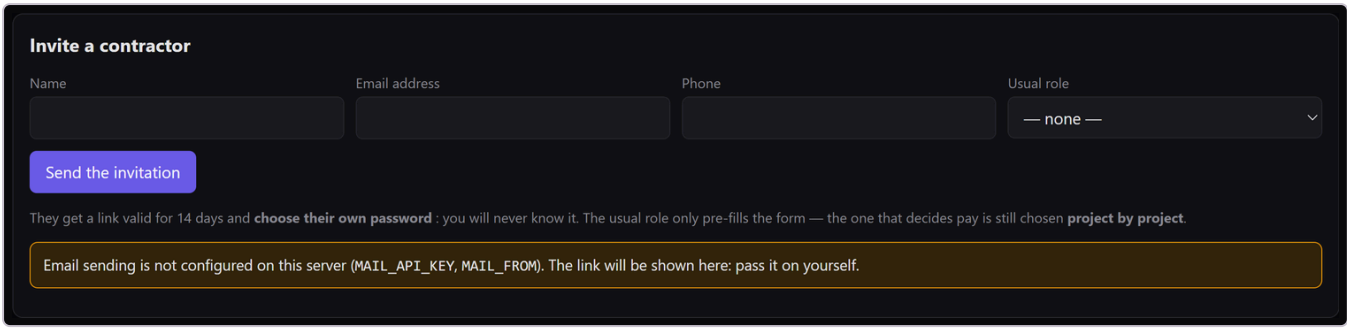
A voice actor, a mixer, an adapter: they need their own access before receiving any work.

`2 minutes`

01 Send the invitation

A name, an address, a usual role. They get a link valid for **14 days** and **choose their own password**: you will never know it.

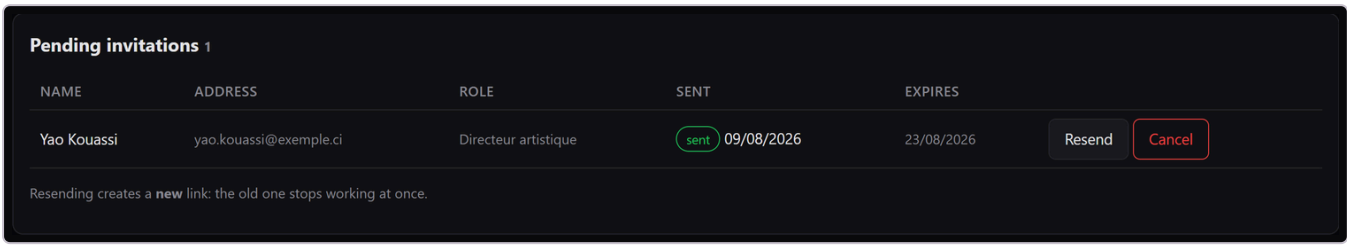
> If e-mail sending is not configured on the server, the console **shows the link** rather than swallowing it: you pass it on yourself. An invitation that fails silently is a person left waiting.



The invitation form: name, address, phone and usual role.

02 Track pending invitations

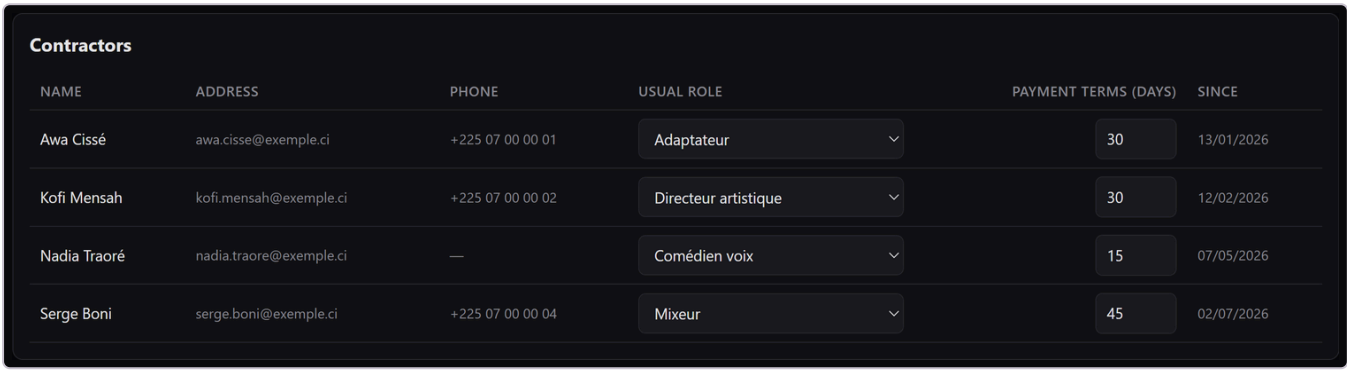
Sent, to be passed on, or failed: the state is stated. **Resend** mints a **new** link, and the old one stops working at once.



The pending invitations list, with their send date and expiry.

03 Set the usual role and payment terms

The **payment terms** belong to each contractor: they set the due date of their invoices, and therefore what shows up in your upcoming outgoings.



The contractor list: usual role and payment terms, editable in place.

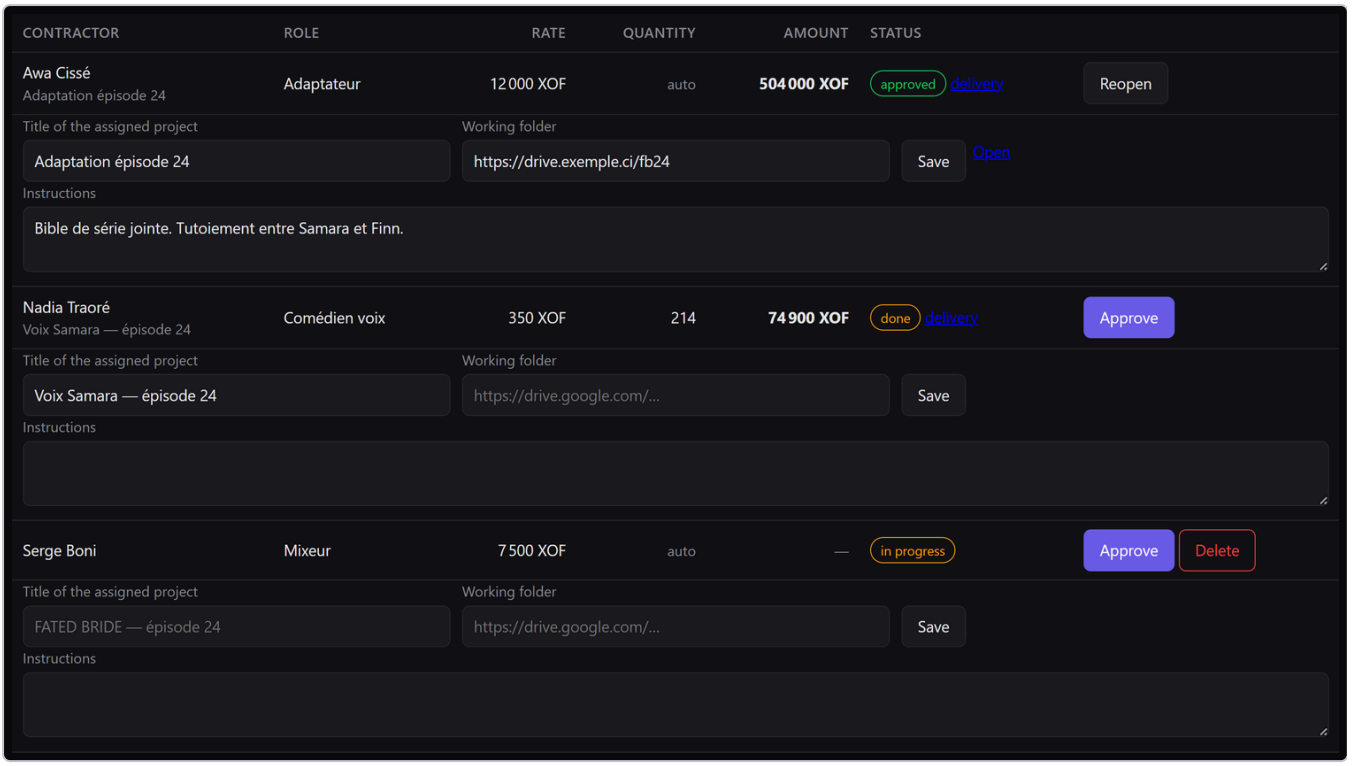
Approve the work, then invoice

The work has been delivered. It has to be frozen, then paid.

3 minutes

01 Approve what came back

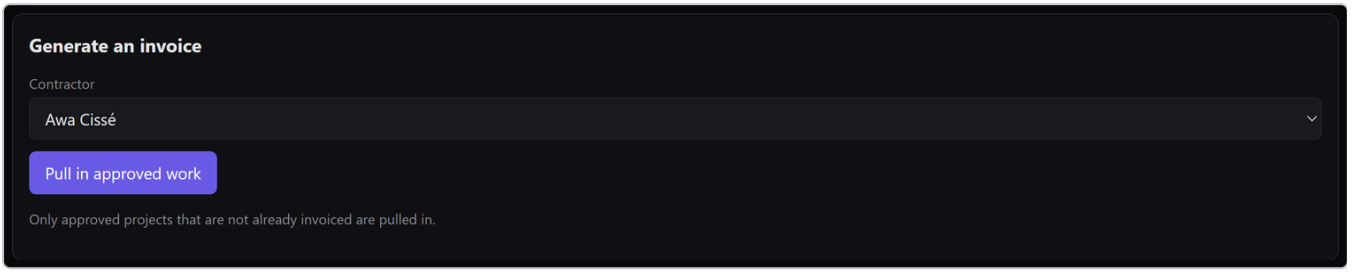
Approving freezes the amount. Changing it afterwards requires **reopening** the assignment — a sum owed is never corrected quietly, and that is what makes the console hold up between you and your contractor.



The assignment table: approved, completed, in progress — with the Approve and Reopen buttons.

02 Generate the contractor's invoice

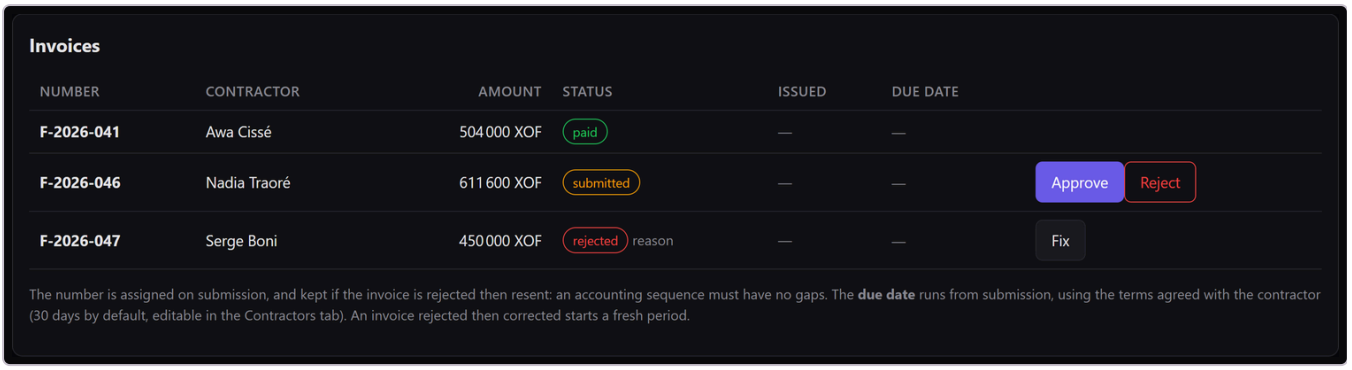
The invoice is built **from the approved work**: nothing to retype, nothing to recount. Each line points back to the assignment that justifies it.



The form that generates an invoice from a contractor's approved work.

03 Track the invoices

Submitted, paid, rejected — and the **reason** for a rejection stays attached to the invoice. A rejection with no written reason gets re-argued by phone, and nobody remembers the same thing.



The contractor invoice list, with each state and the reason for a rejection.

Read the dashboard

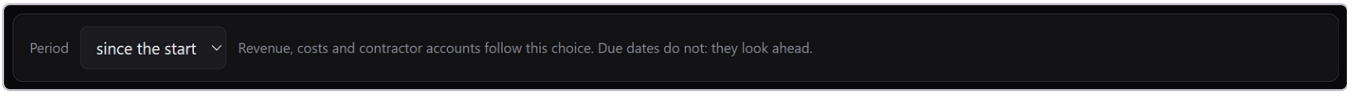
Knowing what the studio earned, what it owes, and which project actually pays.

5 minutes

01 Choose the period

Week, month, year, or since the beginning. Revenue, costs and contractor accounts follow this choice; **deadlines do not** — they look ahead.

> An all-time figure says nothing about the current financial year, and still passes for a result.



The dashboard period selector.

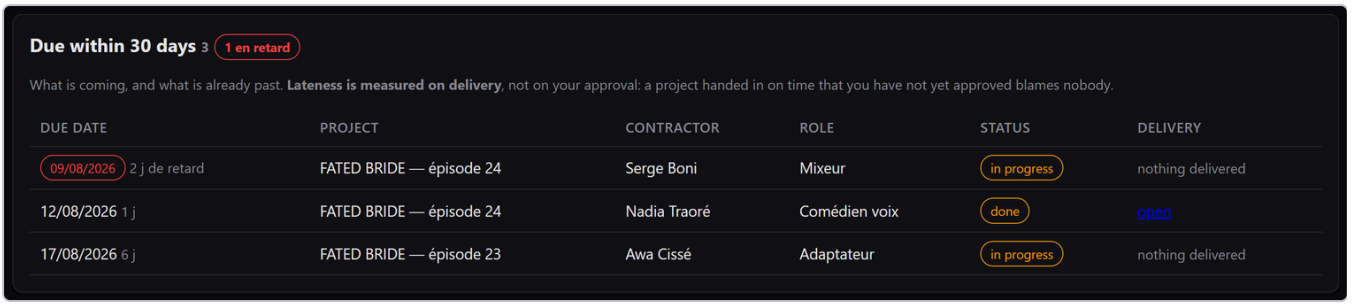
02 Read the overview

Billed and **collected** are deliberately separate: an issued invoice is not cash. On the right, what you still owe contractors, and what you have already paid them.

The five indicators: invoiced, still to collect, costs, result, owed to contractors.

03 Watch the deadlines

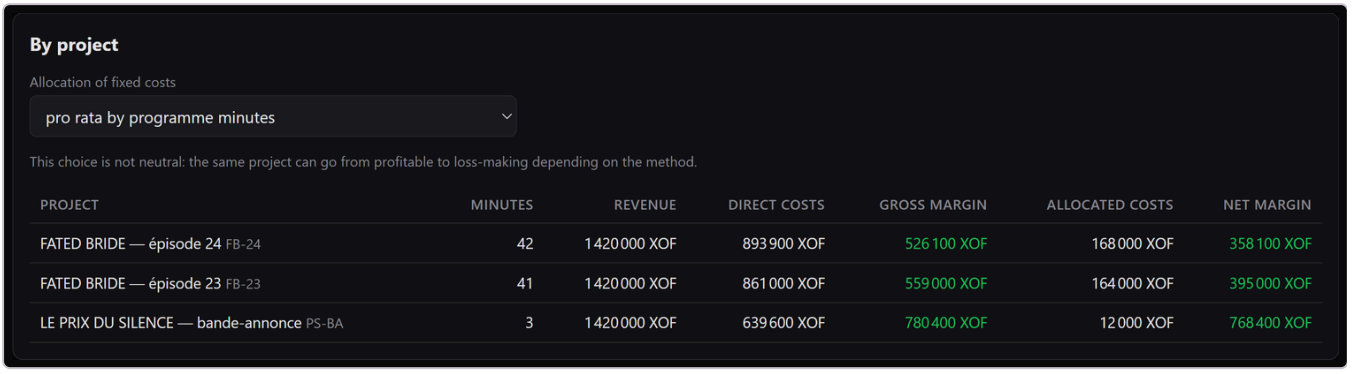
Thirty days ahead, plus what has already gone by. **Lateness is measured on delivery**, not on your approval: work delivered on time that you have not yet approved accuses nobody.



The thirty-day deadlines, one of them late, with each state and delivery.

04 See what each project earned

Revenue, direct costs, gross margin, then fixed costs allocated — **pro rata by minutes**, pro rata by cost, or not at all. That choice is not neutral: the same project swings from profitable to loss-making depending on the method.



The per-project table: minutes, revenue, direct costs, gross margin, allocated costs, net margin.

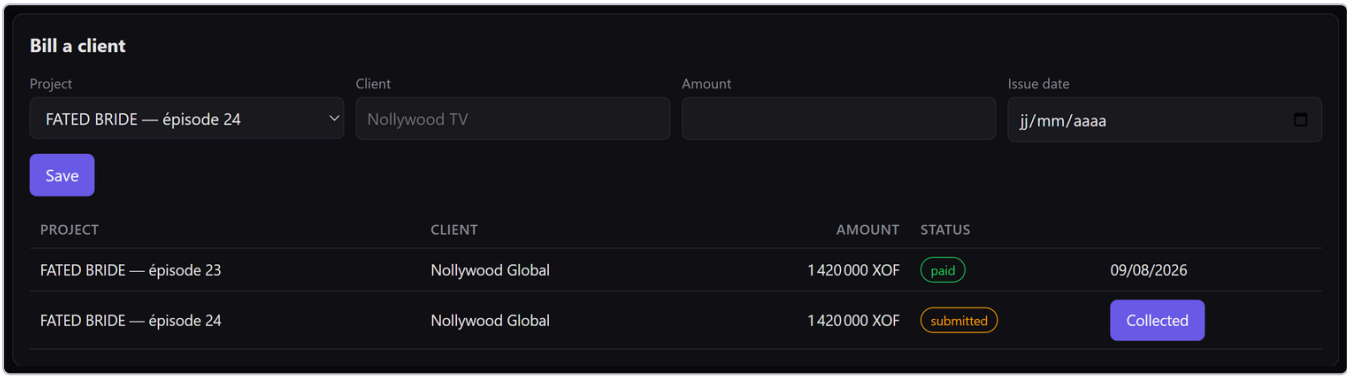
Record revenue and costs

What the client pays, what the studio spends. Without both, the dashboard says nothing.

`5 minutes`

01 Invoice a client

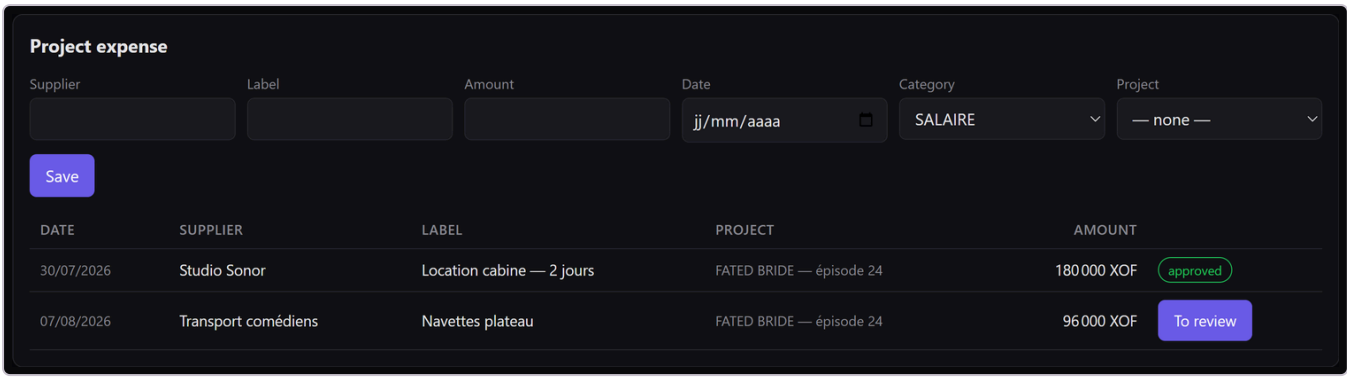
A project, a client, an amount, an issue date. The invoice stays **issued** until you mark it **collected**: that is what separates, on the dashboard, what is billed from what is in the bank.



The client invoicing form and the list of issued invoices.

02 Enter an expense

Supplier, label, amount, category — and **the project, if the expense belongs to one**. An expense attached to a project counts in its direct costs; with no project, it stays a studio overhead.

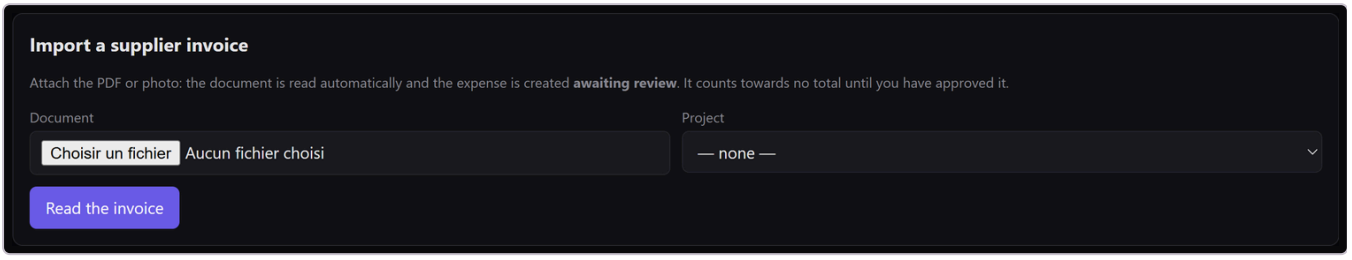


Entering an expense, and the expense list with one awaiting review.

03 Import a supplier invoice

Drop in the PDF or the photo: the document is read and the expense created **pending review**. It **counts in no total** until you have approved it.

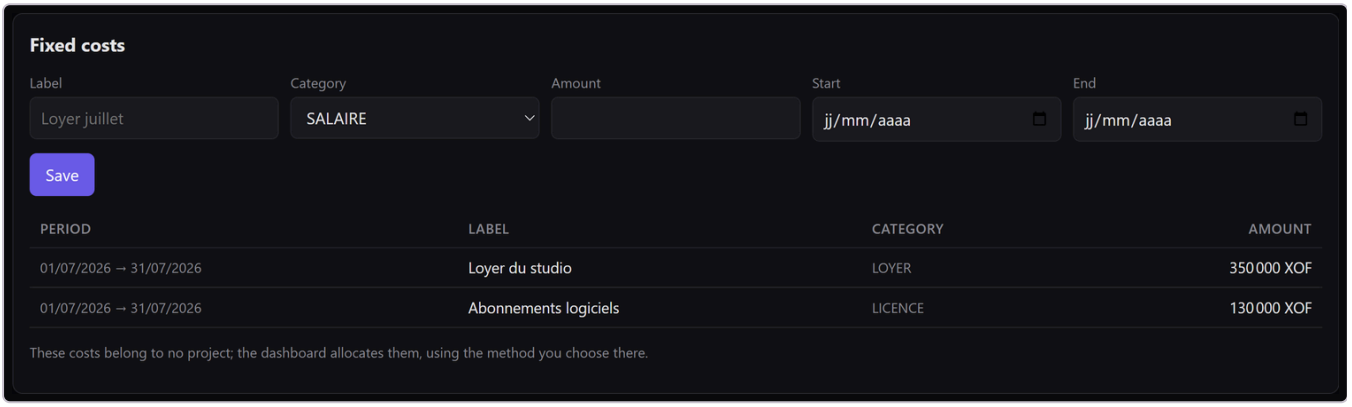
> That is deliberate: automatic reading sometimes misreads a digit, and a wrong cost travels all the way to the project's margin. A visible queue beats a silently wrong total.



Importing a supplier invoice: the document, the project, and the read button.

04 Set the fixed costs

Rent, subscriptions, salaries: what runs over a **period**, not over a project. These are the amounts the dashboard then allocates across projects — pro rata by minutes, by cost, or not at all.



Entering fixed costs, and their list, period by period.

Set the rates for each role

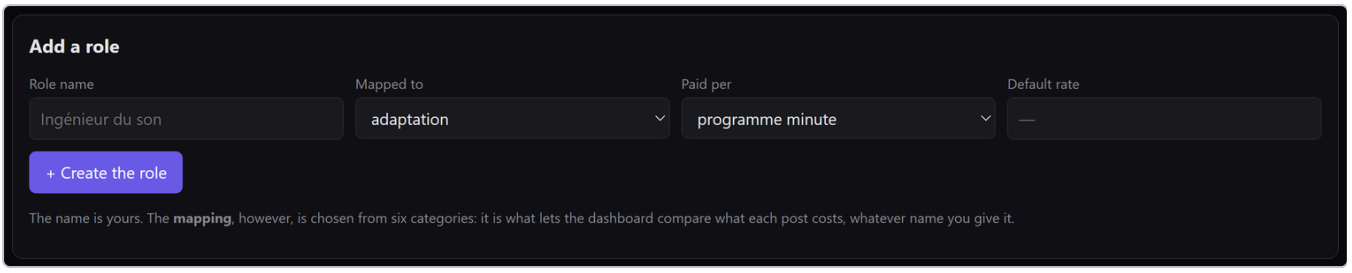
Set once what each role costs, so it never has to be retyped per project.

`3 minutes`

01 Create a role

The **name is yours** — “Sound engineer” if that is what you call it. The **category**, however, is picked from six: it is what lets the dashboard state what adaptation costs, whatever name you gave it.

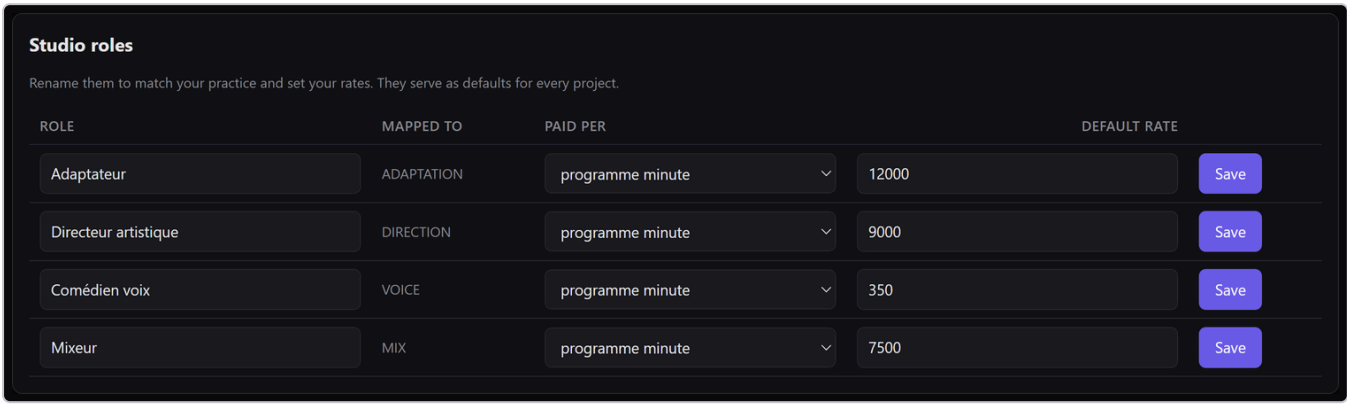
> **Paid by** decides the maths: the *programme minute* for almost every role, the *line* for voice actors, the *flat fee* when the work counts in neither.



Creating a role: name, category, pay basis and default rate.

02 Adjust the rates

These rates are the **defaults** for every assignment: they can still be changed project by project without touching the role. Renaming a role does not change its category, so it does not disturb the dashboard’s comparisons.



The studio’s role list: editable name, category, basis and rate.

Connect the studio and track licences

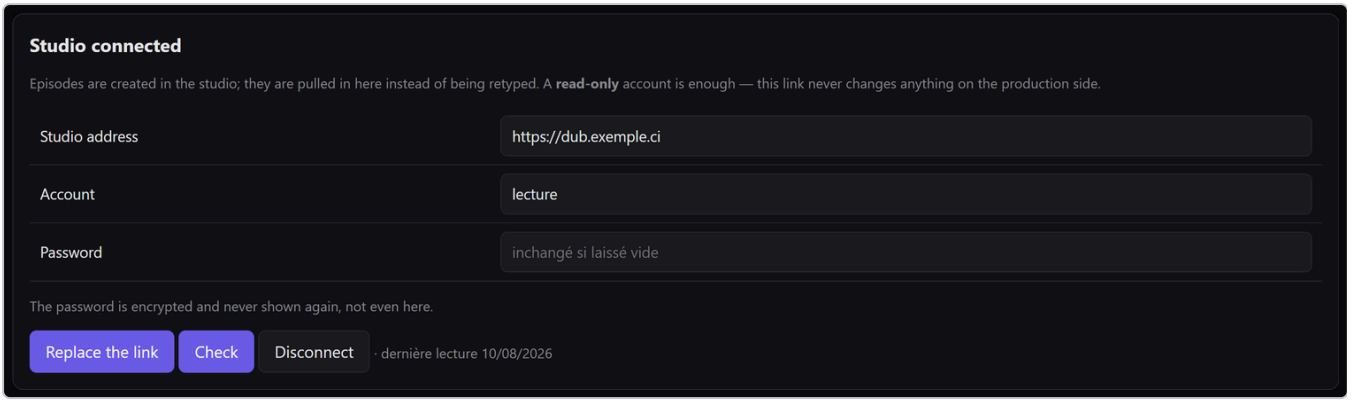
Stop retyping episodes, and know where the licence pool stands.

4 minutes

01 Connect the production studio

Episodes are created in the studio; they are **pulled in here** instead of being retyped. A **read-only** account is enough: this link never changes anything on the production side.

> The password is **encrypted and never shown again**, not even here. If the application key changes, the console says so and asks for it again rather than failing silently on every read.

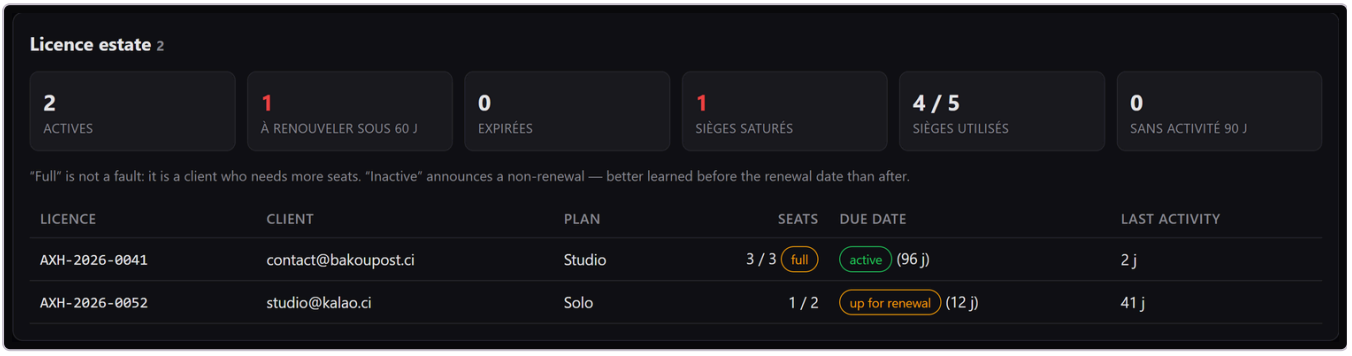


The connector form: studio address, read-only account and password.

02 Read the licence pool

How many licences are sold, which are nearing expiry, which are saturated. **Nothing is copied here**: the pool is read on demand from the licence server, which stays the only source.

> “Saturated” is not a fault: it is a client who needs more seats. “No activity” announces a non-renewal. Better learned before the expiry date than after.



The licence pool: active, to renew, saturated seats, and the per-licence detail.

Contractor side: receive and deliver

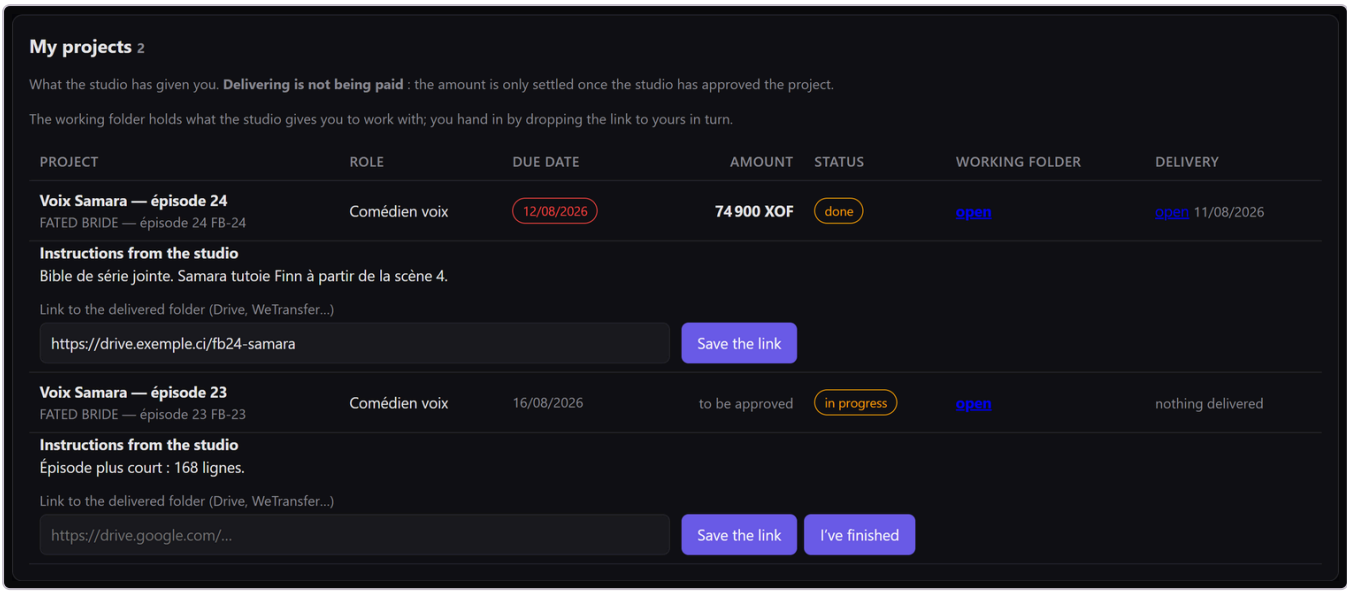
The invited voice actor, mixer or adapter: what they see, and what is expected of them.

2 minutes

01 See what you have been given

A contractor has only **two tabs**: their work and their invoices. The rest of the console is none of their business, and it is removed rather than left there to refuse them.

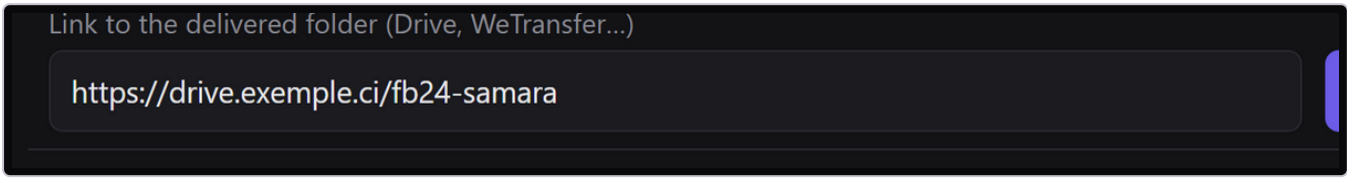
> The **working folder** holds what the studio provides — video, DetX, script, series bible — and the **instructions** say what to know before starting.



The contractor's view: two assignments, their deadline, amount and the studio's instructions.

02 Submit the delivery

You deliver by pasting the **link to the delivered folder**, then marking **I've finished**. **Delivering is not being paid**: the amount is only fixed once the studio approves the work.



The delivery link field and its save button, on the contractor's side.

This tutorial online : `https://axeho.studio/en/tutorials-project-management.html`