# Audion Voice AI - руководство пользователя

**Содержание**

Это руководство описывает текущий рабочий workflow Audion Voice AI Live и Audion Voice AI Studio.

# 1. Выбор издания

Используйте **Audion Voice AI Live**, если вам нужен компактный дистрибутив с API Live и локальными моделями. Это основной вариант для ноутбука, live-диктовки и обычной файловой транскрибации.

Используйте **Audion Voice AI Studio**, если у вас есть CUDA-машина и нужны быстрые локальные прогоны больших моделей, faster-whisper, PyTorch и GPU diarization.

# 2. Установка

## Доустановка моделей распознавания

В раздачу не входят веса моделей — около 7 ГБ: кэш GigaAM, whisper.cpp с моделями large-v2 и large-v3-turbo; с pyannote для разделения по говорящим (только NVIDIA) около 10,6 ГБ. У весов свои лицензии, отдельные от лицензии программы, а условия pyannote принимаются лично на HuggingFace — поэтому скачивает их сам пользователь, под своей учётной записью.

Пока модели не установлены, транскрибация не запустится: окно откроется, а распознавать будет нечем.

При первом запуске программа сама предложит докачать недостающее: окно «Докачать модели и движки» перечисляет модули с объёмом загрузки: GigaAM, whisper.cpp CUDA с моделью Turbo, модель Large V2 и, на машине с NVIDIA, слой разделения по говорящим; вместе около 10,6 ГБ. Кнопка «Скачать и установить» ставит их по очереди; ход виден на странице «Обслуживание», и когда всё скачано, все режимы там горят зелёным. «Позже» откладывает вопрос до следующего запуска, «Больше не предлагать» скрывает окно навсегда. Модули по-прежнему ставятся вручную на той же странице.

То же можно сделать вручную: запустите `builder_main.cmd` и выберите по очереди:

| № | Пункт меню | Что ставит |
|---|---|---|
| 08 | `GIGAAM ONNX` | основная модель распознавания русской речи |

| № | Пункт меню | Что ставит |
|---|---|---|
| 09 | `WHISPER.CPP CUBLAS` | движок whisper.cpp с ускорением CUDA |
| 10 | `WHISPER.CPP LARGE V2` | основная модель для файлов на русском |
| 11 | `CUDA / PYANNOTE` | разделение по говорящим — необязательно |

Первые три шага обязательны. Одиннадцатый нужен только если вы размечаете диалоги по участникам; для него потребуется принять условия модели на HuggingFace.

Откройте `builder_main.cmd` или вкладку `Обслуживание` в GUI.

Рекомендуемый порядок:

1. Запустить программу. Окно «Докачать модели и движки» при первом запуске ставит всё рекомендованное одной кнопкой; дальнейшие пункты нужны для ручного ремонта или проверки.
2. `builder_main.cmd` нужен только если приложение ещё не собрано: он проверяет структуру папок и ставит Python runtime.
3. FFmpeg входит в раздачу. `Переустановить` на странице `Обслуживание` нужно только после смены драйвера NVIDIA: сборка подбирается под драйвер.
4. Live dependencies (микрофон, потоковая диктовка) GUI ставит сам при запуске из `wheelhouse\live`, без сети. Ручная кнопка остаётся для ремонта.
5. `Кэш зависимостей` (`wheelhouse`) входит в раздачу и показывается как `Установлен`. `Переустановить` нужно только если папку удалили: он заново скачает wheels для GigaAM/ONNX Runtime.
6. Нажать `Проверить` в карточке `Проверка микрофона`: тест проверяет устройство записи Windows по умолчанию, затем отдельное устройство связи и нативные частоты 44,1/48 кГц. Аудио не сохраняется.
7. `GigaAM ONNX pack`: onnx-asr, провайдер ONNX Runtime, модели GigaAM v3 и Silero VAD для нарезки длинных записей.
8. `whisper.cpp pack`: Live ставит CPU-сборку, Studio — CUDA/cuBLAS; модель Turbo идёт в комплекте.
9. Studio: `Модель whisper.cpp Large V2` — основная модель для файлов в режиме CUDA. `Диаризация на GPU` (torch + pyannote) — по желанию и только на NVIDIA.
10. Запустить verify/smoke-проверки базовых модулей.

Вкладка `Обслуживание` показывает найденный GPU, рекомендуемый профиль, ход операций, скорость и ETA. Установочный вывод автоматически направляется в левый `Журнал`; отдельного пустого терминала справа нет. Строки помечаются как `Рекомендуется`, `По желанию` или `Не нужно`; нерекомендуемые действия остаются читаемыми и доступными.

`builder_main.cmd` и GUI используют те же install-скрипты. Builder нужен для первичной portable-сборки, recovery и ручного обслуживания; обычный пользователь после запуска GUI устанавливает модули во вкладке `Установки`.

## 3. API-ключи

API-режимы читают ключи из файлов в `config`.

```
config/
  api_key_*.txt
```

Reset App не удаляет эти файлы. Cleanup также должен защищать рабочие конфиги.

## 4. Главное окно

Основные вкладки:

- `Live` - диктовка, overlay, API/Local источники, модели и OpenAI cleanup.
- `Файлы` - очередь файлов, модели, язык транскрипции, постобработка/очистка, субтитры и экспорт.
- `Настройки` - тема, язык приложения, tray, интеграции Notion/Obsidian, глобальные параметры и сброс настроек.
- `Обслуживание` - рекомендуемый профиль, runtime, payloads, модели и служебные операции.

Слева находятся `Журнал` и очередь. Журнал остаётся пустым до диктовки, ошибки или запуска обслуживающей операции: состояние Live и `Right Alt + F12` показываются в tooltip кнопки микрофона и в строке состояния. Справа находятся рабочие настройки текущего сценария.

Правый клик по компактной верхней плашке открывает полупрозрачное меню частых действий. Первые пункты - `Последние диктовки` и `Вставить последнюю диктовку`; ниже находятся режим Live-диктовки, режим файловой записи, папки `Input` / `Output`, окно программы, настройки и выход. Быстрое окно overlay показывает 20 последних записей. Пункт tray `Все диктовки` открывает прокручиваемую историю до 200 завершённых Live-диктовок; после

достижения лимита самые старые записи удаляются автоматически. Нажатие по двухстрочной карточке повторно вставляет её текст в ранее активное окно, а круглые кнопки справа копируют или удаляют запись. В Live-режиме готовая плашка показывает микрофон; в файловом режиме - красный символ Record и подпись `Начать запись`. Во время файловой записи кнопка паузы исключает этот интервал из WAV, не закрывая микрофон, Stop сохраняет файл, а Cancel удаляет временную запись. У плашки и tray нет всплывающих подсказок: действия подписаны или используют стандартные транспортные символы.

Масштаб overlay в свежей установке по умолчанию равен 70%. Последующие изменения сохраняются. Колесо мыши прокручивает страницу настроек и не меняет случайно значение масштаба под указателем.

Крайний левый kebab `⋮` виден уже рядом с готовым значком микрофона. Overlay можно перетащить до начала записи, не запуская захват голоса.

Горячая клавиша `Right Alt + F12` активируется автоматически вместе с приложением. Для локального STT доступна отдельная настройка **Загружать локальную модель заранее**: она ускоряет первую диктовку, но заранее занимает RAM/VRAM. На готовность плашки, tray и горячей клавиши эта настройка не влияет.

## 5. Язык приложения и язык транскрипции

Это разные настройки.

- **Язык приложения** меняет язык интерфейса.
- **Язык транскрипции** сообщает движку, на каком языке запись.

Для неизвестного языка оставьте `Авто`. Если запись точно русская или английская, лучше выбрать конкретный язык.

Для Live-диктовки используется отдельный **Основной язык**. Значение по умолчанию — `Раскладка окна`: при начале диктовки приложение читает текущую Windows-раскладку именно целевого окна, куда затем будет вставлен текст (`ru` или `en`). `Русский`, `Английский` и `Авто` служат явным переопределением. Общий **Словарь терминов** виден и на вкладке `Live`, и в файловых операциях независимо от выбранного API/Local-режима. В него через запятую вводятся только точные написания имён, названий, аббревиатур и форматов. Словарь передаётся Live и файловым STT как keyterms/prompt, если выбранный движок поддерживает подсказки, а при включённом оформлении защищает написание терминов. GigaAM ONNX не принимает prompt/hotwords, поэтому словарь не меняет его сырое распознавание. Отдельное поле **Описание записи** предназначено для свободного текста о теме, участниках, цели и важных

фактах; дублировать там словарь не нужно. Стартовый словарь содержит градостроительные сокращения `ИАС УГРТ`, `ФГИС ТП`, `КИПРР`, `ГП`, `ПТП`, `ПЗЗ`, `ЗОУИТ`, `КРТ`, `МНГП`, `РНГП`, `СТП`, `ССЭР`.

## 6. Файловая транскрибация

1. Откройте вкладку `Файлы`.
2. Добавьте файлы или папку в очередь отдельными круглыми кнопками рядом с `Добавить…`. Чекбоксы в первом столбце позволяют запустить только нужные файлы. Если галочек нет, Start обрабатывает всю очередь; массовое удаление и очистка временных данных по-прежнему могут использовать обычное выделение строк.
3. Выберите файловый движок: OpenAI, Local Models или CUDA в Studio.
4. Выберите язык транскрипции.
5. Для OpenAI выберите профиль транскрипции; для локального режима - модель и backend.
6. Включите нужные форматы экспорта.
7. Нажмите старт.

По умолчанию результаты сохраняются рядом с исходником. Это удобнее для больших архивов: готовый файл лежит там же, где запись.

Папка добавляется со всеми вложенными папками: поддерживаемые файлы ложатся в очередь плоским списком, а расшифровки — рядом с каждым файлом в его папке. Полный список форматов (17 аудио и 18 видео) показывают подсказки кнопок `Файлы…`, `Папку…` и подписи `Форматы`. Выбранные через пикер или drag-and-drop внешние файлы не копируются в проект: очередь хранит их исходные пути и обрабатывает их на месте. В `input` автоматически создаются только WAV-файлы, записанные самим Audion. Кнопки `Input` и `Output` занимают оставшуюся ширину строки и открывают соответствующие проектные папки.

## 7. Поддерживаемые форматы

Приложение опирается на FFmpeg, поэтому лучше использовать распространённые форматы:

- audio: WAV, MP3, M4A, AAC, FLAC, OGG, OPUS;
- video: MP4, MOV, MKV, WEBM, AVI.

Если файл экзотический, предварительно конвертируйте его в WAV, MP3 или MP4.

## 8. API workflow

OpenAI подходит для быстрой качественной транскрибации и очистки текста. В Live модель OpenAI не выбирается вручную: Realtime использует `gpt-realtime-whisper`, batch fallback использует `gpt-4o-mini-transcribe`. xAI и ElevenLabs сейчас используются как фиксированные realtime Live-провайдеры.

- корректный API-ключ нужного провайдера;
- выбранный профиль OpenAI для файлов;
- выбранная модель постобработки;
- prompt очистки, если включена генерация чистого текста.

Профили OpenAI для файлов:

- `Быстро / экономно` -> `gpt-4o-mini-transcribe`;
- `Макс. точность` -> `gpt-4o-transcribe`;
- `С диаризацией` -> `gpt-4o-transcribe-diarize`.

Каждый профиль имеет tooltip с объяснением. Кнопка обновления моделей остаётся полезной для post-processing/cleanup моделей.

## 9. Local Models workflow

Local Models - локальный режим без отправки аудио во внешний API. Он нужен для приватной работы, длинных записей и машин, где уже установлен подходящий GPU/CPU backend.

Локальные модели:

- GigaAM - предпочтительный локальный выбор для русской версии интерфейса;
- whisper.cpp - CPU fallback в Live и CUDA/cuBLAS GPU pack в Studio;
- backend: авто, CUDA, DirectML или CPU fallback;
- GigaAM Live держит модель в памяти до полного выхода из приложения;
- длинные записи GigaAM режет на отрезки до 25 секунд в самых тихих точках (по Silero VAD, который ставится вместе с GigaAM ONNX pack; без него по энергии сигнала). Ничего не выбрасывается, тихая речь с эхом тоже идёт в модель; каждый отрезок получает время начала и конца;
- выгрузка/порог буфера относится к whisper.cpp live-сценариям.

Перед использованием установите нужные runtime/payloads и скачайте модели на странице `Обслуживание` (или согласитесь в окне первого запуска). Wheels для GigaAM/ONNX Runtime лежат в `wheelhouse` раздачи: `GigaAM ONNX pack` ставит из них `onnx-asr` и ONNX Runtime provider, прогревает `gigaam-v3-e2e-ctc` / `gigaam-v3-e2e-rnnt` в `models\huggingface` и докачивает Silero VAD. На Windows auto-маршрут выбирает DirectML как лёгкий универсальный backend; в Studio на NVIDIA используется CUDA.

В Studio режим `Транскрипция + диаризация` запускает `pyannote` вторым проходом для локальных файловых движков, включая GigaAM. Для GigaAM приложение автоматически использует более короткие чанки (`diarization.gigaam_chunk_seconds`, по умолчанию 45 секунд), потому что GigaAM отдаёт текст на уровне чанка, а не word timestamps. Live-режим pyannote не использует.

### 9.1. Установка локальных backend

Audion выбирает backend по факту загрузки provider DLL/runtime, а не только по названию видеокарты.

- **DirectML**: основной лёгкий Windows fallback для Live и запасной путь для Studio. Устанавливается через `GigaAM ONNX pack` как `onnxruntime-directml`; внешний SDK не нужен. Нужен актуальный драйвер NVIDIA/AMD/Intel.
- **CUDA**: путь Studio для NVIDIA. Сначала установите актуальный NVIDIA driver и Studio runtime/payloads через `Установки` / `builder_main.cmd`, затем запускайте `GigaAM ONNX pack`. Для ONNX Runtime нужен `onnxruntime-gpu`, а CUDA/cuDNN/MSVC DLL должны быть доступны процессу; Audion дополнительно вызывает `onnxruntime.preload_dlls()`.
- **TensorRT**: не используется в текущем профиле проекта. Если ONNX Runtime показывает TensorRT provider, Audion не выбирает его как рекомендуемый backend.

## 10. CUDA workflow в Studio

CUDA доступен только в Studio. Он рассчитан на NVIDIA GPU и покрывает GigaAM ONNX CUDA, whisper.cpp CUDA/cuBLAS и faster-whisper/pyannote.

В карточке `CUDA` есть профиль Faster-Whisper:

- `Качество` - режим по умолчанию. Используется обычный Faster-Whisper/CTranslate2 без batched inference. Он спокойнее грузит GPU, лучше подходит для грязной речи, тихих вступлений, служебных реплик и последующей диаризации, потому что таймлайн обычно получается подробнее.

- `Скорость` - включает BatchedInferencePipeline с `batch_size=16`. Этот режим заметно плотнее загружает CUDA/GPU и ускоряет длинные файлы или очередь файлов, но сильнее зависит от VAD, может склеивать сегменты крупнее и иногда пропускать тихие пограничные фразы. Используйте его, когда нужно быстро прогнать материал и GPU можно отдать задаче.

Типичный сценарий:

1. Установить `whisper.cpp pack` (CUDA/cuBLAS) и `Модель whisper.cpp Large V2`; окно первого запуска делает это само.
2. Установить `Диаризация на GPU` (CUDA/pyannote), если нужна разметка по говорящим.
3. Запустить verify.
4. Turbo остаётся быстрым профилем для сравнения с large-v2.
5. Выбрать Studio GPU-движок: GigaAM CUDA, whisper.cpp cuBLAS или faster-whisper CUDA.
6. Для speaker labels выбрать `Транскрипция + диаризация`.
7. Запустить короткий smoke на небольшом файле.
8. После этого запускать многочасовые записи.

На RTX 5070 smoke подтвердил рабочие целевые профили Studio GigaAM CUDA и Studio whisper.cpp CUDA/cuBLAS. На машине без NVIDIA GPU CUDA smoke может проверить только импорт модулей.

## 11. Live-диктовка

Live-диктовка запускается из вкладки `Live`, кнопкой записи над логом или из tray.

Режимы:

- **API models** - OpenAI batch, OpenAI Realtime, xAI Realtime или ElevenLabs Realtime. OpenAI-модель выбрана программой и не показывается как каталог.
- **Local Models** - GigaAM или whisper.cpp с выбранным backend.

Сначала выберите источник (`API models` или `Local Models`), затем конкретный provider или локальную модель. Неактивная карточка остаётся затемнённой, чтобы экран не выглядел пустым и при этом не подталкивал к неправильным настройкам.

Если включено `При запуске`, диктовка стартует вместе с приложением.

## 12. Overlay

Overlay показывает текущую живую расшифровку поверх других окон и заранее занимает нативную геометрию полного контроллера шириной 420-680 px. В idle маска оставляет видимым и интерактивным только центральный овал 120×12 px. При наведении маска снимается и сразу появляется полноразмерная Record-плашка с неподвижной иконкой микрофона строго по центру. После клика иконка мягко гаснет примерно за 220 мс, а элементы записи появляются в тех же границах без изменения размера или позиции окна. Минимальная и стандартная высота — 52 px.

В настройках можно:

- включить или выключить overlay;
- изменить высоту;
- выбрать поведение при live-диктовке.

## 13. Очистка live-текста

Для длинной диктовки можно включить `Чистка Live`.

Стандартная очистка работает консервативно и прежде всего восстанавливает пунктуацию, регистр, границы предложений и абзацы. Она не должна пересказывать текст, переставлять мысли или заменять слова синонимами; допускается удаление только очевидных филлеров, фальстартов и случайных повторов.

Параметры:

- модель очистки;
- prompt очистки;
- количество предложений, после которого запускается очистка.

Это удобно для длинного голосового набора: приложение периодически превращает поток фраз в аккуратный текст.

## 14. Экспорт

Экспорт доступен из GUI и tray.

Форматы:

- Markdown;
- TXT;
- JSON;
- SRT;
- WebVTT.

Действия:

- сохранить лог в Markdown через файловый пикер;
- экспортировать транскрипт;
- отправить материал в Notion;
- отправить материал в Obsidian.

## 15. Tray

Tray нужен для быстрых действий, когда основное окно закрыто или спрятано.

Сворачивание окна скрывает приложение в tray. Кнопка закрытия завершает приложение полностью, останавливает фоновые процессы и удаляет значок из tray.

Возможности:

- показать или скрыть приложение;
- старт/стоп live;
- экспортировать лог в Markdown;
- отправить результат в Notion/Obsidian.

Если tray-поведение не нужно, отключите его в `Настройки`.

## 16. Настройки и сохранение

После перезапуска сохраняются:

- тема;
- язык приложения;
- чекбоксы;
- заполненные поля;

- выбранные модели;
- списки моделей;
- настройки live;
- настройки экспорта;
- настройки tray, интеграций и глобального поведения приложения.

Селекты не меняются колесом мыши, чтобы случайная прокрутка не ломала параметры.

## 17. Reset App

`Reset App` возвращает дефолтные настройки программы. Используйте его, если интерфейс или workflow были случайно настроены неправильно.

Reset App не должен удалять:

- API-ключи;
- установленные runtimes;
- payloads;
- скачанные модели;
- рабочие файлы пользователя.

## 18. Cleanup

`cleanup_project.cmd` очищает то, что можно восстановить на другой системе: временные build-артефакты, runtime, `Tools`, модели, `install\download`, `install\wheels` и рабочие payload-папки.

Удаляются также содержимое `input`, `output`, `logs`, `report`, `workspace` и `release`. Это ожидаемо: `input` - временная рабочая зона для файлов, а не архив пользователя.

Защищены рабочие конфиги, API-ключи, install-скрипты, `system_core`, `Docs`, `tests` и важные корневые файлы проекта. После cleanup пустая структура восстанавливается через `install\init_folders.cmd`.

GUI-каталог и `builder_main.cmd` синхронизированы. На RTX 5070 проверены целевые профили: Live GigaAM DirectML, Live whisper.cpp CPU fallback, Studio GigaAM CUDA и Studio whisper.cpp CUDA/cuBLAS.

## 19. Рекомендации

- Для коротких задач используйте API models.
- Для приватных и длинных задач пробуйте Local Models.
- Для больших архивов на NVIDIA GPU используйте Studio и CUDA.
- Храните результат рядом с исходником.
- Не смешивайте язык приложения и язык транскрипции.
- Сначала запускайте smoke на коротком файле, потом многочасовые записи.

## 20. Если что-то пошло не так

- Проверьте лог в левом окне.
- Убедитесь, что FFmpeg установлен.
- Проверьте API-ключи для OpenAI, xAI или ElevenLabs.
- Для Local Models проверьте наличие модели и runtime.
- Для CUDA проверьте NVIDIA driver и наличие `whisper.cpp pack` CUDA/cuBLAS с моделью Large V2; PyTorch нужен только диаризации.
- Длинная запись в GigaAM режется на отрезки до 25 секунд автоматически. Если отрезок всё же падает с ошибкой DirectML, обновите драйвер видеокарты или переключите backend на CPU.
- Studio на машине без NVIDIA: выключите `Транскрипция + диаризация`, иначе после расшифровки конвейер остановится на отсутствующем pyannote.
- Если модели ставились вручную и матрица `Готовность режимов` не зелёная, откройте `Обслуживание`: строка с `Не установлен` показывает, чего не хватает.
- Нажмите Reset App, если проблема похожа на сломанную конфигурацию UI.

## Чек-лист пакетной обработки в Studio

Просмотрите полный список источников, папку вывода, язык, движок, модель, диаризацию, временные метки и настройки экспорта. Перед пакетом прогоните один показательный файл и проверьте качество текста, границы говорящих, тайминг и имена файлов.

Для CUDA убедитесь в драйвере NVIDIA, совместимом стеке PyTorch/runtime, наличии модели и свободной VRAM. При откате на CPU или DirectML ожидайте другую скорость и, возможно, другие возможности backend. Зафиксируйте фактический профиль, на котором принят пакет.

## API и локальная обработка

API-маршрутам нужны сеть, ключ, модель, квота и согласие с политикой данных. Локальным маршрутам нужны установленные backend и модели. Не смешивайте ошибки провайдера с ошибками декодирования, FFmpeg, модели и формата вывода.

Не отправляйте чувствительный звук в API, если задача требует обработки без сети. Расшифровки, субтитры, карты говорящих, отчёты и журналы могут оставаться чувствительными даже после удаления исходного звука.

## Наблюдение за пакетом

Следите за ходом по каждому файлу и за временем появления результатов. Тихая загрузка модели или ожидание ответа провайдера сами по себе не означают зависание. Перед остановкой задачи проверьте дочерний процесс, загрузку GPU/CPU, журнал и последний завершённый файл.

Не допускайте, чтобы параллельные задачи писали в один и тот же вывод. Оставляйте достаточно места на диске под декодированный звук и временные сегменты.

## Восстановление

Сохраните упавший файл, журнал, выбранный движок и модель, профиль железа и параметры. Повторяйте только упавшие файлы, если имена результатов стабильны. `Reset App` нужен при испорченных настройках интерфейса, а не при отсутствующих зависимостях или несовместимых компонентах CUDA.

## Интерфейс и конфигурация как справочник

Декларативные настройки GUI и карты конфигурации — структурированный источник сведений о доступных движках, моделях, полях, умолчаниях, подсказках и действиях backend. Руководство превращает их в производственную последовательность: подготовить пакет, выбрать локальную или API-обработку, оценить нагрузку на железо и приватность, следить за файлами по отдельности и принимать расшифровки с доказательствами.

При проверке релиза сравнивайте видимые элементы управления, сохранённые настройки, сформированный запрос к движку, отчёты по файлам, экспорт и документацию. Новые параметры пакета должны говорить, на что влияют: декодирование, сегментация,

распознавание, диаризация, чистка, экспорт, параллельность, стоимость или хранение. Условные элементы должны объяснять, когда появляются и какой запасной путь срабатывает без локальной модели или компонента CUDA.