



European
Commission

EUROPEAN COMMISSION
Directorate-General for
Interpretation



Getting started with OPEN SOURCE LIVE TRANSCRIPTION

Technical guide

Table of Contents

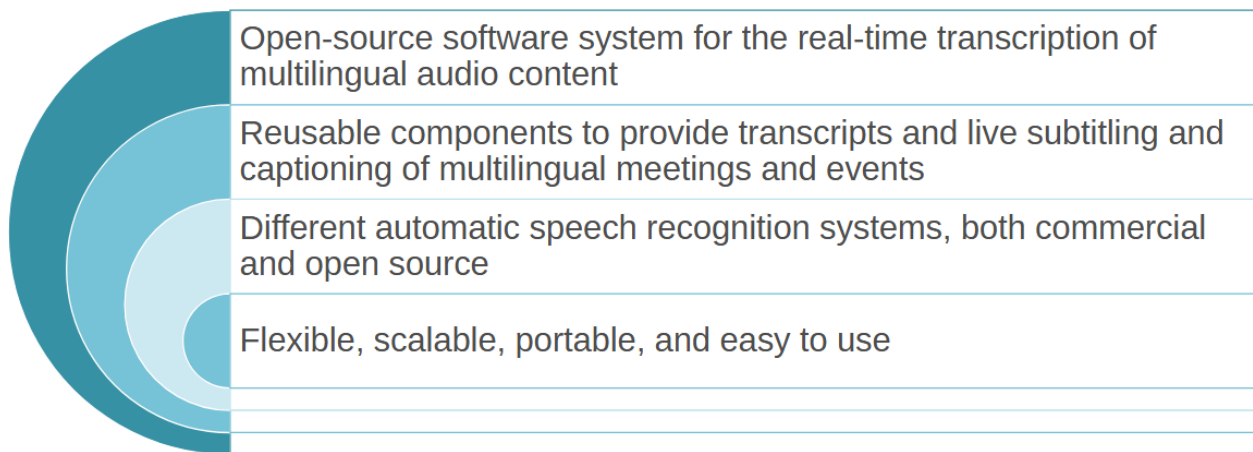
| | |
|--|----|
| Introduction to Open Source Live Transcription..... | 4 |
| Installation..... | 5 |
| Download..... | 5 |
| Configure and build..... | 6 |
| Other dependencies..... | 6 |
| Quickstart..... | 6 |
| Option 1 : Run with docker compose (straightforward)..... | 6 |
| Option 2 : Run locally (without docker)..... | 7 |
| Initialize the app..... | 9 |
| 1. Add a transcriber profile:..... | 9 |
| 2. Create and start a session:..... | 10 |
| 3. Connect to the web interface:..... | 11 |
| 4. Stream..... | 11 |
| Note on supported transports, formats and ASR:..... | 12 |
| 5. Available Automatic Speech Recognition connectors..... | 12 |
| Transcriber currently supports 2 kinds of ASR systems and is easy to extend :..... | 12 |
| The README explains how to create a custom ASR using our ASR plugin system..... | 12 |
| Open Broadcaster Software..... | 12 |
| Enabled routes..... | 13 |
| Frontend access..... | 13 |
| Session API (swagger)..... | 14 |
| Docker: How to build images..... | 14 |
| Docker: How to run..... | 15 |
| Use in production :..... | 15 |
| Tests..... | 16 |

| | |
|------------------------|----|
| Unit tests..... | 16 |
| Integration tests..... | 17 |
| System overview..... | 17 |

|

Introduction to Open Source Live Transcription

Welcome to the Live Transcription Open Source Toolbox! This software is designed to assist enterprises and organizations in effectively managing transcription sessions effectively from multiple audiovisual streams.




It perfectly suited for handling meetings in both physical or virtual setups, particularly when participants speak different languages. It bridges **any kind of audiovisual stream to multiple automatic speech recognition (ASR) providers to enable real-time transcription and deliver live closed captions.**

This software features a modular design (toolbox) and is optimized for server deployment. It is built around a session management system, accessible via a comprehensive **API** that enables session operators to interact with the platform.

The Live Transcription Toolbox provides a **real-time web interface**, designed for dynamic interaction with both ongoing and past transcription sessions.

Through this interface, operators can not only review and manage sessions but also share public, read-only links that allow anonymous participants to view transcripts in real-time.



LIVE TRANSCRIBER ONLINE

Running Session

My 2nd session
4 nov. 2023, 22:51

<<

<

1

>

>>

Audio (fr-FR)

COPY SESSION LINK

COPY CONTENT

EXPORT

09:20:17
French
(France)

Bonjour, bienvenue à notre rendez-vous quotidien pour la presse en ligne et en direct sur EBS. Nous sommes aujourd'hui le jeudi 18 mars.

09:20:26
French
(France)

J'aimerais commencer avec une mise à jour à l'agenda de la présidente demain en début d'après-midi, la présidente von der Leyen et le président du Conseil européen Charles Michel, rencontreront par visioconférence le président de la Turquie Recep Tayyip Erdoğan pour faire le point des relations entre l'Union européenne et la Turquie dans le contexte de la préparation du Conseil européen de la semaine prochaine.

puis j'appelle maintenant johannes

Stopped Session

My 1st session
4 nov. 2023, 22:49
4 nov. 2023, 22:50

<<

<

1

>

>>

de la Turquie Recep Tayyip Erdoğan pour faire le point des relations entre l'Union européenne et la Turquie dans le contexte de la préparation du Conseil européen de la semaine prochaine. puis j'appelle maintenant johannes

Installation

Download

On the target system (server) which can be your laptop for testing purpose :

```
git clone https://code.europa.eu/speech_recognition/speech-to-text
```

The project structure includes the following modules:

- **front-end**: A front-end application to use sessions, download transcripts and access live closed-captions
- **session-manager**: An API for managing transcription sessions; it also serves a front-end using Swagger client (Open API spec)

- **transcriber**: A transcription service (streaming endpoint & relay to ASR services)
- **scheduler**: A service that bridges the transcribers and subtitle delivery with the session manager, database, and message broker
- **subtitle-delivery**: Components that generate and serve subtitles (Websocket API) and downloadable transcripts through an HTTP API supporting multiple formats including VTT, SRT, TXT, Docx)
- **lib** folder: Contains generic tooling for the project as a whole, treated as another Node.js package. This allows the modules to access the tools provided by the **lib** package and use them in their implementation.

Configure and build

Other dependencies

The system depends on external dependencies including a **PostgreSQL database** and an **MQTT-compatible message broker**, which must be properly installed and configured.

Note: While the provided Docker Compose file (see quickstart below) facilitates initial setup by running these components, additional configuration tailored to your specific production needs will be required. See [Use in production :](#)

Quickstart

To quickly test this project, you can use either a local build or docker compose.

Option 1 : Run with docker compose (straightforward)

1. Create a **.env** file at the root of the project with this content. Make sure to adapt to your needs if you change some bits in the configuration (ensure correct ports for database, MQTT broker...):

```
DB_USER=myuser
DB_PASSWORD=mypass
DB_NAME=mydb
DB_PORT=5433
STREAMING_PASSPHRASE=false
STREAMING_USE_PROXY=false
```

```
STREAMING_PROXY_HOST=127.0.0.1
SESSION_API_HOST=http://localhost
BROKER_PORT=1883
DELIVERY_WEBSERVER_HTTP_PORT=8001
SESSION_API_WEBSERVER_HTTP_PORT=8002
SESSION_API_BASE_PATH=/sessionapi
FRONT_END_ADMIN_USERNAME=admin
FRONT_END_ADMIN_PASSWORD=admin
FRONT_END_PORT=8000
FRONT_END_PUBLIC_URL=http://localhost/frontend
DELIVERY_WS_BASE_PATH=/delivery
SESSION_API_PUBLIC_URL=http://localhost/sessionapi
DELIVERY_WS_PUBLIC_URL=ws://localhost
DELIVERY_PUBLIC_URL=http://localhost/delivery
DELIVERY_SESSION_URL=http://sessionapi:8002
UDP_RANGE=8889-8999
LETS_ENCRYPT_EMAIL=mail@mail.com
DOMAIN_NAME=localhost
TRANSCRIBER_REPLICAS=2
SESSION_SCHEDULER_URL=http://scheduler:8003
SCHEDULER_WEBSERVER_HTTP_PORT=8003
```

2. Run the docker-compose command:

```
make run-docker-dev
```

This compose file will compile all the docker images and launch all the containers. This will allow you to test the API and transcription.

Option 2 : Run locally (without docker)

System prerequisites

The modules are mainly written in Node.JS 20+. You might use NVM for installing it (curl -o- <https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh> | bash)

To run, modules require the following system dependency :

```
sudo apt-get install build-essential
```

```
sudo apt-get install libgstreamer-plugins-base1.0-dev
sudo apt-get install gstreamer1.0-tools
sudo apt-get install libgstreamer1.0-dev
sudo apt-get install libsrt1.5-gnutls
sudo apt-get install srt-tools
sudo apt-get install libsrt-gnutls-dev
sudo apt-get install libsrt-openssl-dev
sudo apt-get install libssl-dev
sudo apt-get install gstreamer1.0-plugins
sudo apt-get install gstreamer1.0-plugins-base
sudo apt-get install gstreamer1.0-plugins-bad
sudo apt-get install gstreamer1.0-plugins-good
sudo apt-get install gstreamer1.0-libav
```

Here are the steps to follow:

```
make install-local
```

This command will build all npm packages.

```
make run-dev
```

This command will start all the services locally. **You may need to spin docker containers for the MQTT broker and the database. You should then tune the .env accordingly.**

Initialize the app

You will interact with the running containers using a **Swagger Interface** that will help you to get schemas and payloads for your requests.

| | | | |
|----------------------|----------------------------|-----------------------------------|---|
| transcriber_profiles | | | ^ |
| GET | /transcriber_profiles | Get all transcriber configs | 🔒 |
| POST | /transcriber_profiles | Create a new transcriber config | 🔒 |
| GET | /transcriber_profiles/{id} | Get a transcriber config by ID | 🔒 |
| PUT | /transcriber_profiles/{id} | Update a transcriber config by ID | 🔒 |
| DELETE | /transcriber_profiles/{id} | Delete a transcriber config by ID | 🔒 |
| healthcheck | | | ^ |
| GET | /healthcheck | Check the health of the service | 🔒 |
| sessions | | | ^ |
| GET | /sessions | Get all sessions | 🔒 |
| POST | /sessions | Create a new session | 🔒 |
| GET | /sessions/active | Get all active sessions | 🔒 |
| GET | /sessions/terminated | Get all terminated sessions | 🔒 |
| GET | /sessions/{id} | Get a session by ID | 🔒 |
| DELETE | /sessions/{id} | Delete a session by ID | 🔒 |
| PUT | /sessions/{id}/start | Start a session by ID | 🔒 |
| PUT | /sessions/{id}/stop | Stop a session by ID | 🔒 |

1. Add a transcriber profile:

Log in to the session API is available via the following link:

<http://localhost/sessionapi/api-docs/>. In the POST /transcriber_profiles section, add the following json:

```
{
  "config": {
    "type": "microsoft",
    "name": "microsoft_custom_fr",
    "description": "microsoft custom fr",
    "languages": [
```

```
{
  "candidate": "LANG (fr-FR BCP_47 code)",
  "endpoint": "ENDPOINT"
},
{
  "key": "KEY",
  "region": "REGION",
  "endpoint": "ENDPOINT"
}
}
```

2. Create and start a session:

- In the session API POST /sessions, create a new session with the following json:

```
{
  "name": "test session",
  "channels": [
    {
      "name": "test channel",
      "transcriberProfileId": 1
    }
  ]
}
```

- Retrieve the session id from the request return
- Start the session using the PUT sessions/IP/start endpoint specifying the session id
- Retrieve your channel's streaming endpoint via GET sessions/ID

```
{
  "id": "b43e2b32-4156-4463-a4e4-c76113376f6e",
  "status": "active",
  "name": "string",
  "start_time": "2024-04-24T06:16:44.315Z",
}
```

```

"end_time": null,
"errored_on": null,
"createdAt": "2024-04-24T06:15:50.218Z",
"updatedAt": "2024-04-24T06:16:44.315Z",
"channels": [
  {
    "transcriber_id": "15e3e05a-da71-4e26-a6a1-ce4da05d4cd7",
    "languages": [
      "en-GB",
      "fr-FR"
    ],
    "name": "string",
    "stream_endpoint": "srt://localhost:8889?
mode=caller&passphrase=unoo890qfklpxyezxo3",
    "stream_status": "active",
    "transcriber_status": "ready",
    "closed_captions": null,
    "closed_caption_live_delivery": "1950b1a9-a762-4da9-a2c3-d7e0d76a8b3e",
    "closed_captions_file_delivery": null,
    "createdAt": "2024-04-24T06:15:50.271Z",
    "updatedAt": "2024-04-24T06:16:50.701Z",
    "transcriberProfileId": 1
  }
]
}

```

3. Connect to the web interface:

- Go to: <http://localhost/frontend/admin.html>
- The user/password combination will be admin/admin by default , but can be adapted though the .env file.
- Now select your session

4. Stream

You are now ready to receive real-time transcriptions. In order to do this, you must send your SRT stream to the streaming endpoint. You should use the following command to stream the file “fr.mp3”:

```

gst-launch-1.0 filesrc location=./fr.mp3 ! decodebin ! audioconvert !
audioresample ! avenc_ac3 ! mpegtsmux ! rtpmp2tpay ! srtsink
uri="srt://localhost:8889?mode=caller&passphrase=unoo890qfklpxyezxo3"

```

You can now see the transcriptions appearing in real time.

You might use your local microphone to stream directly using the following command:.

```
gst-launch-1.0 alsasrc ! audioconvert ! audioresample ! avenc_ac3 !  
mpegtsmux ! rtpmp2tpay ! srtsink uri="srt://localhost:8889?  
mode=caller&passphrase=unoo890qfklpxyezxo3"
```

You should adapt the command to use the correct audio source.

Note on supported transports, formats and ASR:

The system currently supports **SRT protocol** for input streams.

As the Transcriber component leverages a **Gstreamer transcoding pipeline**, any kind of audiovisual streams are supported (aac, mp3, ogg...). For optimal bandwidth efficiency, we recommend prioritizing audio streams over video streams whenever possible.

This method effectively saves bandwidth while still ensuring the delivery of high-quality audio.

5. Available Automatic Speech Recognition connectors

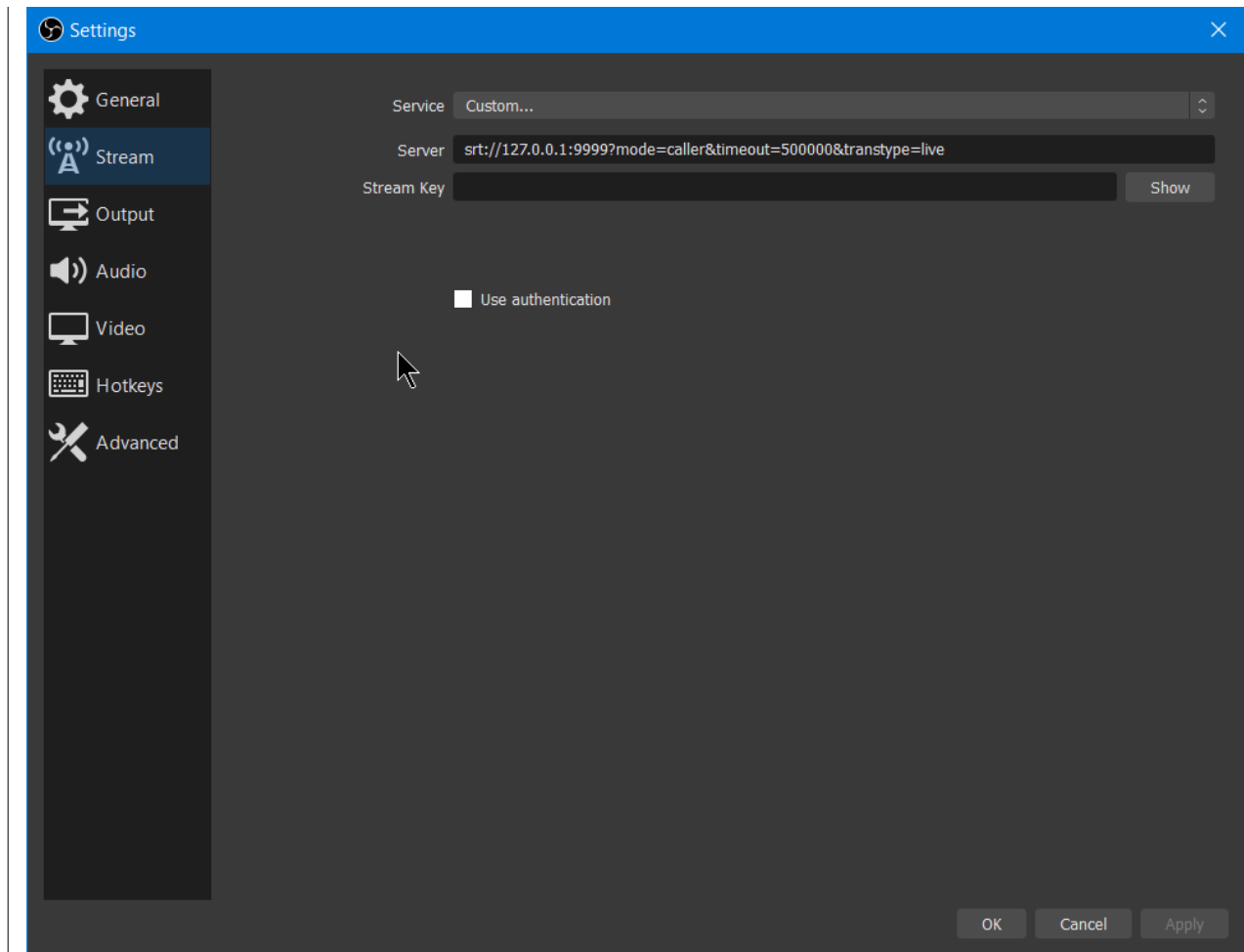
Transcriber currently supports 2 kinds of ASR systems and is **easy to extend** :

- Microsoft Azure Speech Service [Link](#)
- LinTO STT (Kaldi, Whisper V3) [Link](#)

The README explains how to create a custom ASR using our ASR plugin system.

Open Broadcaster Software

Latest version(s) of [OBS](#) supports SRT streams natively. You can create a stream and connect the appropriate **streaming endpoint** once a session is started (GET session by Id)



In this scenario, you want to use `srt://localhost:8889?mode=caller&passphrase=unoo890qfklpxyezxo3`

As the “Server” in your OBS configuration.

Enabled routes

Once the service is launched, several routes are accessible:

Frontend access

- <http://localhost/frontend/admin.html> → The frontend URL allows viewing of the sessions and receiving transcriptions. Also allows to download transcript in a variety of formats, including time-coded subtitle files (.vtt, .srt)



EUROPEAN COMMISSION

Automatic transcription

Test OBS Session - Fr-FR (fr-FR)
03 November 2023 11:10

| | |
|--------------------------|--|
| 11:11 French (France) | Financer l'achat des statuts par des gens en inscrivant le nom des donateurs dessus et ça leur a permis de faire l'acquisition de son statut supplémentaire. C'est minimum 7000€ la statue et aujourd'hui y en a 177 et en 2017, rénovation de la totalité du projet pour encore 3000000 d'euros. Maintenant, voyons l'impact sur le tourisme. |
| 11:12 French (France) | C'est pas possible, on doit forcément être à la pire date de l'année. Je sais pas. On m'avait prévenu que c'était mort, mais là. |
| 11:12 French (France) | Non? Et en 2023, en fait, les 30 ans du quartier. Ils veulent encore agrandir l'exposition en allant chercher 7000000 d'euros supplémentaires par financement participatif. C'est beau, certes, mais. |

- <http://localhost/frontend/user.html> → This frontend URL is similar to the admin one but only allows viewing a single session. The URL for this view is generated from the admin view.

Session API (swagger)

- <http://localhost/sessionapi/api-docs/> → This route allows access to the Swagger interface for configuring/using sessions.

Docker: How to build images

All components are dockerized following the same process. In every component folder, you'll find a Dockerfile and its associated docker-entrypoint.sh. To compile the Docker image of a component, you must position yourself **at the root of the cloned repository** and not in the component folder, then launch the command:

```
docker build -f [COMPONENT]/Dockerfile .
```

For example, to build the **Transcriber**, launch the command:

```
docker build -f Transcriber/Dockerfile.
```

You can compile images in this way for the following components:

- Delivery
- Scheduler
- Session-API
- Transcriber
- front-end
- migration

In practice, for local testing, there is no need to manually compile these images because Docker Compose will do it for you.

Docker: How to run

In order to launch the Docker containers, 3 Docker Compose files are provided:

- `docker-compose.yml` -> It is used for a secure HTTPS production deployment.
- `docker-compose-dev.yml` -> It is used for local deployment in order to perform manual tests.
- `docker-compose-test.yml` -> It is specifically used in integration tests launched by the `integration-test.sh` script.

To make use of Docker Compose, it is recommended to refer to the quickstart section which guides you step by step through the complete launch of the service.

Use in production :

The complete suite of services is designed to function as a standalone, production-ready system, and is bundled with a `docker-compose.yml` file tailored for this purpose. This setup includes a **Traefik reverse proxy** for seamless integration, with **automatic Let's Encrypt SSL certificate issuing**. This scenario starts with this command

```
make run-docker-prod
```

Important Note:

When deploying this configuration, it is essential to meticulously configure the environment variables and `docker-compose.yml` file. Please consult the annotations in the `.envdefault` and the Docker Compose file for detailed guidance on customization. Pay particular attention to the following aspects:

```
##### Inbound / Pulled streaming #####
# transcriber might listen to an UDP port directly (like using
# network mode host in docker)
# if transcriber is behind a proxy, use those configs to
# correctly reach the transcriber container. Also make sure to
# set UDP_RANGE to a fixed port like 8889-8889 so that the
# transcriber container is reachable.
# i.e: using network_mode as host, keep this default values
# and every transcriber spawned will listen to a random port
# within the range on the host machine
# i.e : deploying transcriber behind a proxy, you'll want to
# fix the UDP port of transcriber to some value (e.g. 8889), set
# the proxy address as STREAMING_PROXY_HOST (e.g.
# mystreamingdomain.com) and port accordingly to your needs.
STREAMING_PROXY_HOST=false # Transcriber host for inbound
streaming (false to disable)
```

Kubernetes

K8s deployments are not covered within this documentation.

Tests

Unit tests

The code being entirely event-driven, it is difficult to test it in a unitary way. For this reason, the unit tests have focused on very specific points. Specifically, the unit tests concern the circular buffer of the transcriber and are located in `Transcriber/tests.js`.

Integration tests

In order to comprehensively test the project, integration tests have been added and are entirely carried out in a bash script named `integration-test.sh` at the root of the project. These tests validate several parts of the code:

- The correct launch of all services
- The creation of a session and the enrollment of the transcriber
- The start of the session and the initiation of the transcriber's pipeline
- Resilience when a transcriber crashes and recovers
- The streaming of SRT (SubRip Text) streams
- The recording of closed captions in the session
- The stopping of the session

After creating the `.envtest` file (as documented at the beginning of `integration-test.sh`), these tests can be simply run with `./integration-test.sh`.

System overview

