EUROPEAN COMMISSION
Directorate-General for Interpretation

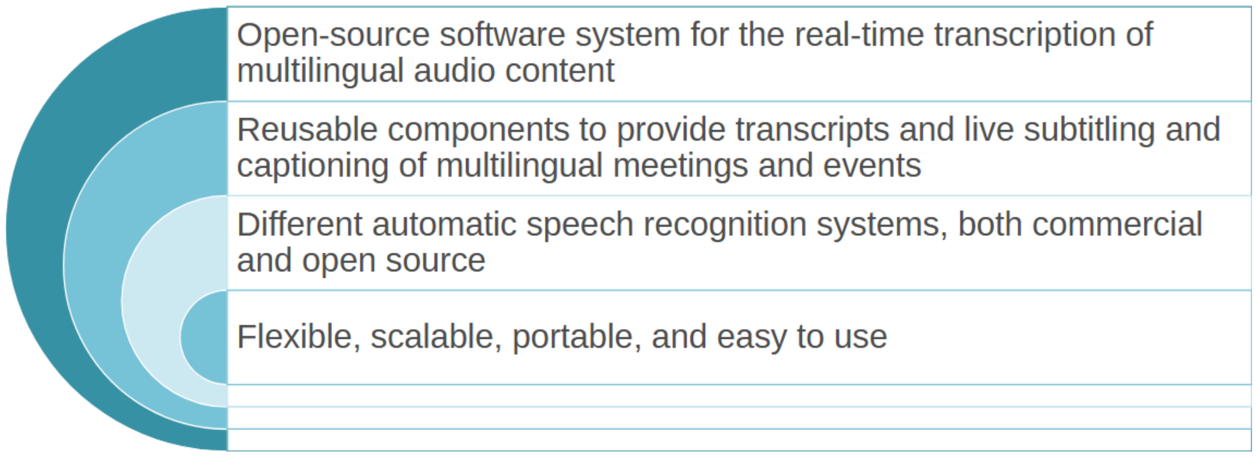
Getting started with OPEN SOURCE LIVE TRANSCRIPTION

Technical guide

Table of Contents

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.


Open-source software system for the real-time transcription of multilingual audio content
Reusable components to provide transcripts and live subtitling and captioning of multilingual meetings and events
Different automatic speech recognition systems, both commercial and open source
Flexible, scalable, portable, and easy to use

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 does not offer a web interface as it is. However, when combined with LinTO Studio, it 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. We will detail the integration with LinTO Studio later in this document and present its web interface.

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:

- Session-API: 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
- lib folder: Contains generic tooling for the project as a whole, treated as another Node.js packageT 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=test
STREAMING_USE_PROXY=false
STREAMING_PROXY_HOST=127.0.0.1
STREAMING_PROTOCOL=SRT,WS,RTMP
STREAMING_WS_TCP_PORT=9012
STREAMING_PROXY_WS_TCP_PORT=9012
STREAMING_RTMP_TCP_PORT=9013
STREAMING_PROXY_RTMP_TCP_PORT=9013
SESSION_API_HOST=http://localhost:8002
BROKER_PORT=1883
SESSION_API_WEBSERVER_HTTP_PORT=8002
TRANSCRIBER_REPLICAS=1
```

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
```

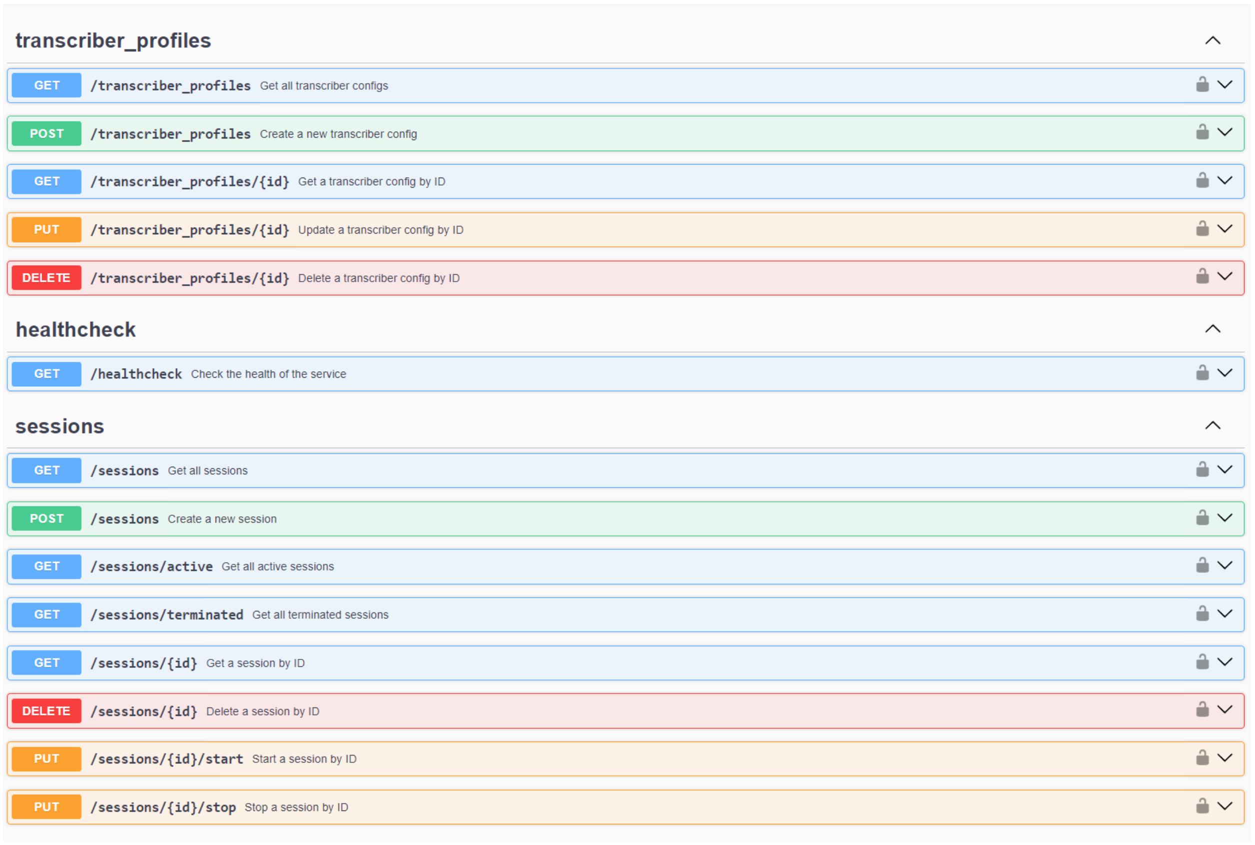
This command will build all npm packages.

```
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": "ready",
  "name": "string",
  "start_time": "2024-04-24T06:16:44.315Z",
  "end_time": null,
  "errored_on": null,
  "owner": null,
  "organizationId": null,
  "public": false,
  "createdAt": "2024-04-24T06:15:50.218Z",
  "updatedAt": "2024-04-24T06:16:44.315Z",
  "channels": [
    {
      "keepAudio": false,
      "diarization": false,
      "index": 0,
      "languages": [
        "fr-fr"
      ],
      "translations": null,
      "name": "string",
      "stream_endpoints": {
      "srt": "srt://localhost:8889?streamid=37af8315-8b6f-4fb7-ab52-
09a6449f54cc,0&mode=caller&passphrase=testtesttesttesttest",
            "rtmp": "rtmp://localhost:9013/37af8315-8b6f-4fb7-ab52-09a6449f54cc/0",
            "ws": "ws://localhost:9012/37af8315-8b6f-4fb7-ab52-09a6449f54cc,0"
          },
      "stream_status": "inactive",
      "transcriber_id": null,
      "audioFile": "",
      "createdAt": "2024-08-13T09:16:20.310Z",
      "updatedAt": "2024-08-13T09:23:10.623Z",
      "transcriberProfileId": 1,
      "sessionId": "37af8315-8b6f-4fb7-ab52-09a6449f54cc"
    }
  ]
}
```

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 in the logs.

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

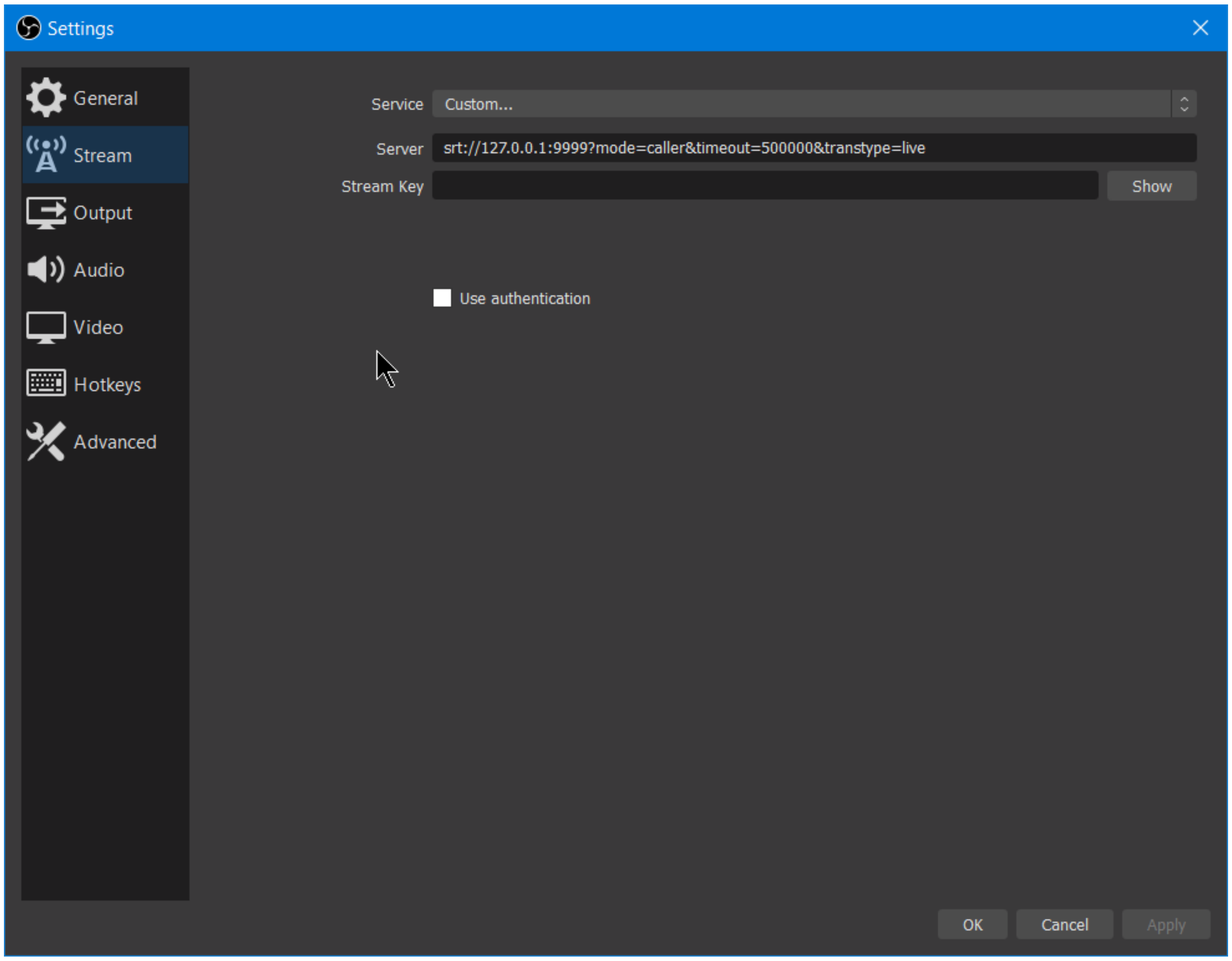
Transcriber currently supports 2 kinds of ASR systems and is **easy to extend** :

- 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 in the section **Custom ASR**.

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)



Service: Custom...
Server: srt://127.0.0.1:9999?mode=caller&timeout=500000&transtype=live
Stream Key:
Use authentication

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, the API documentation is accessible:

Session API (swagger)

- http://localhost:8002/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:

- Scheduler
- Session-API
- Transcriber
- 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:

- compose.yml -> This is the base compose file defining the common properties of the different services.

- compose.prod.yml -> It is used for a secure HTTPS production deployment.
- compose.override.yml -> It is used for local deployment in order to perform manual tests.
- 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.

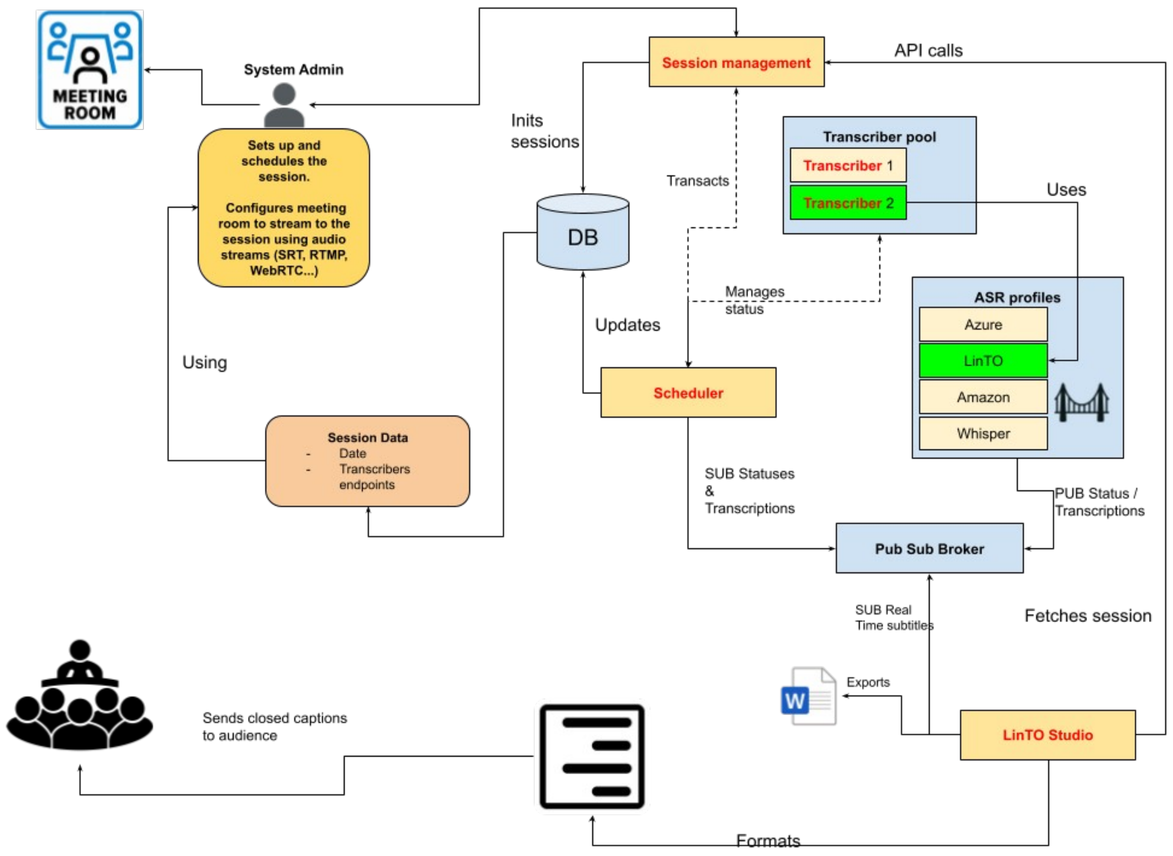
Integration tests

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



Intégration with LinTO Studio

LinTO Studio is an open Source AI driven recording, transcription and media management solution. It has been specially adapted to be compatible with the toolbox and to integrate functionalities such as transcription, the session and channel system, as well as the concept of transcriber profiles.

Installation

To simplify usage, everything has been done to make the developer's life easier. For this purpose, the following elements have been added to the toolbox repository:

- a Git submodule located in the linto-studio folder at the root of the project

- a compose.linto-studio.yml and compose.linto-studio-override.yml file allowing the use of the linto-studio compose files without modification and adapting them to the context of the toolbox
- an environment file .envdefault.linto.docker allowing the configuration of LinTO Studio's environment variables from the Toolbox
- a Makefile target allowing everything to be easily launched with a single command

To initialize the submodule if you have already cloned the repository:

```
git submodule update --init --recursive
```

To clone the repository directly with the linto-studio module:

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

And now, start all the services with a single command:

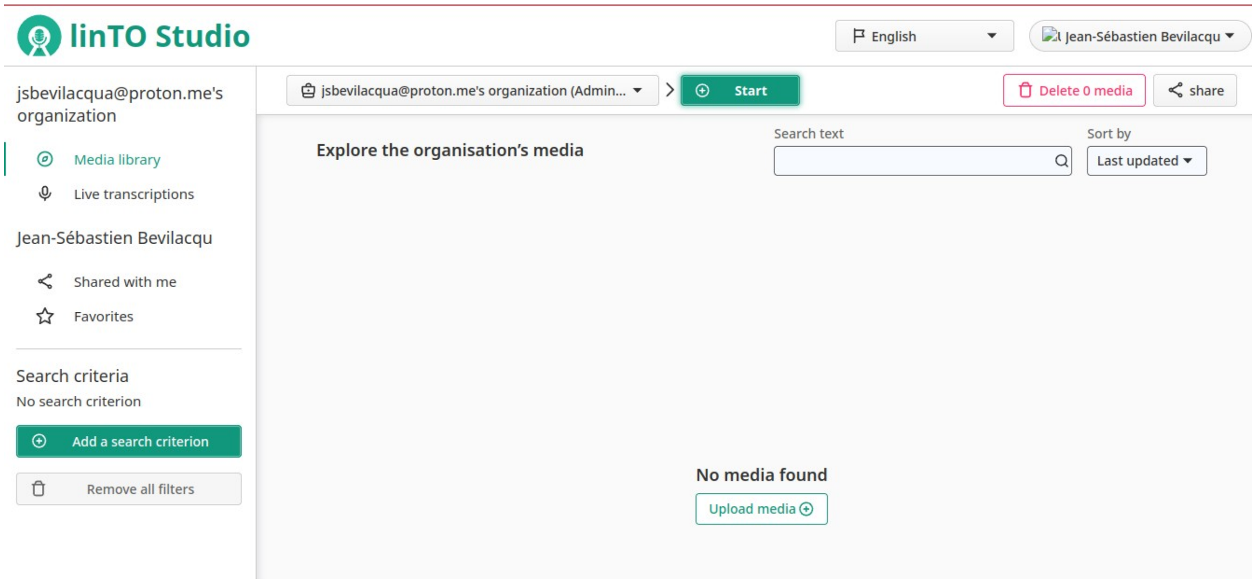
```
make run-docker-dev-linto-studio
```

You can now connect to LinTO Studio at the URL http://localhost:8003

Web Interface

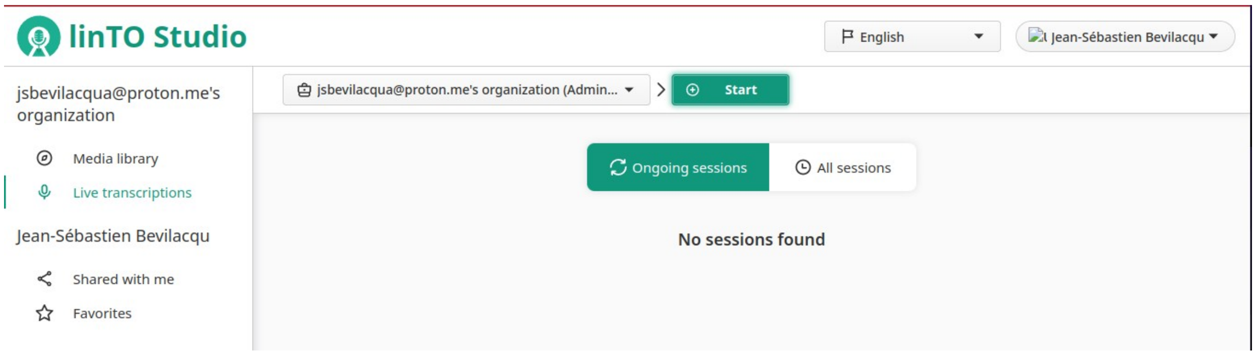
The web interface is intuitive and easy to access. First, create an account and connect to the web interface. Here is an example that allows you to view real-time transcription.

1. Welcome to the homepage



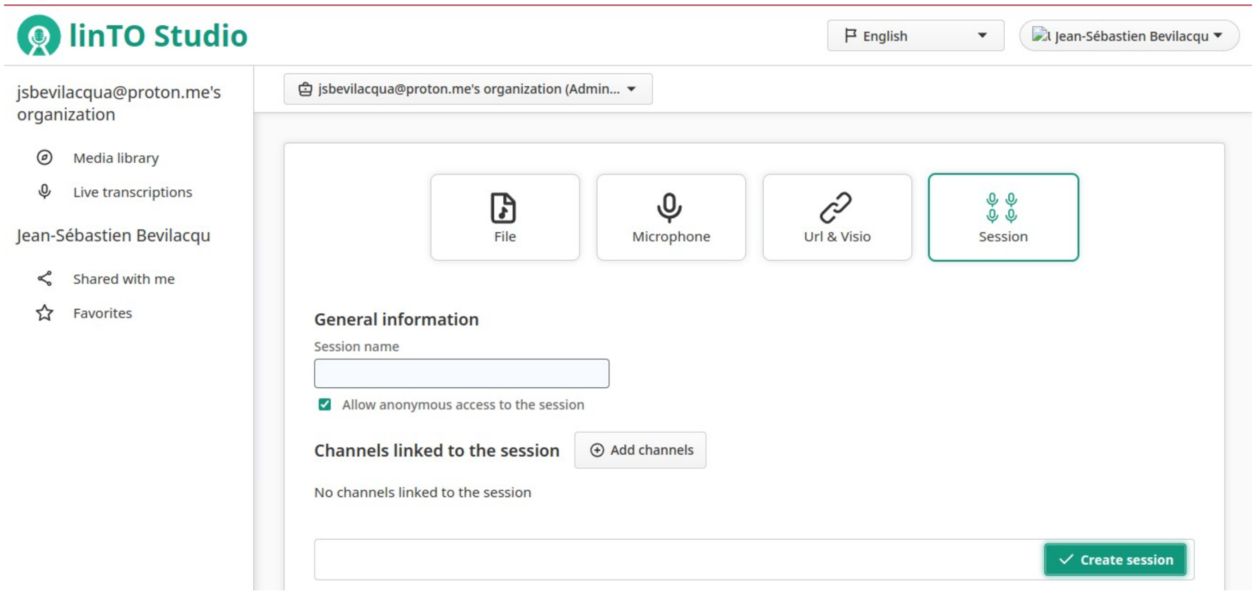
Here, you will find the web interface with all its elements. The detailed usage of the complete interface is beyond the scope of this documentation. We will focus only on the part related to the Toolbox.
Click on "Live transcriptions" in the left-hand menu.

2. List session view



We arrive at the Sessions view. Since we haven't created any sessions yet, it's empty. We will now create our first session. To do this, click on the "Start" button.

3. Start session view



General information

Session name

✓ Allow anonymous access to the session

Channels linked to the session

No channels linked to the session

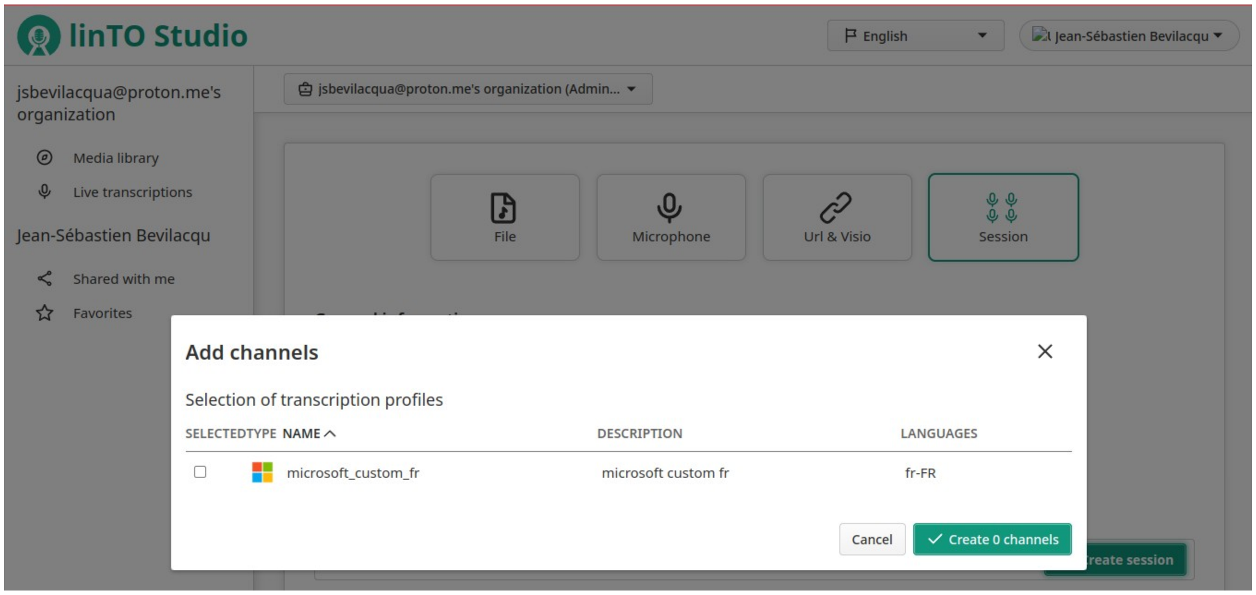
To create a new session, select the "Session" object type (the 4th square). We can now specify a name for the session, set its visibility, and add channels to it.

We have not yet created a Transcriber Profile, and the web interface currently does not allow us to do so. Transcriber Profiles are essential for creating channels. Therefore, we will create a Transcriber Profile using curl.

```
curl -X POST http://localhost:8005/v1/transcriber_profiles \ -H "Content-Type: application/json" \ -d '{ "config": { "type": "microsoft", "name": "string", "description": "string", "languages": [ { "candidate": "string", "endpoint": "string" } ], "key": "string", "region": "string", "endpoint": "string" } }'
```

Replace the various values with your configuration parameters.
For a more intuitive interface, you can make this request via the Swagger UI available here:
http://localhost:8005/api-docs/#/transcriber_profiles/post_transcriber_profiles
.

4. Channel Selection



linTO Studio

English

Jean-Sébastien Bevilacqu

jsbevilacqua@proton.me's organization

jsbevilacqua@proton.me's organization (Admin...

Media library

Live transcriptions

Jean-Sébastien Bevilacqu

Shared with me

Favorites

File · Microphone · Url & Visio · Session

Add channels

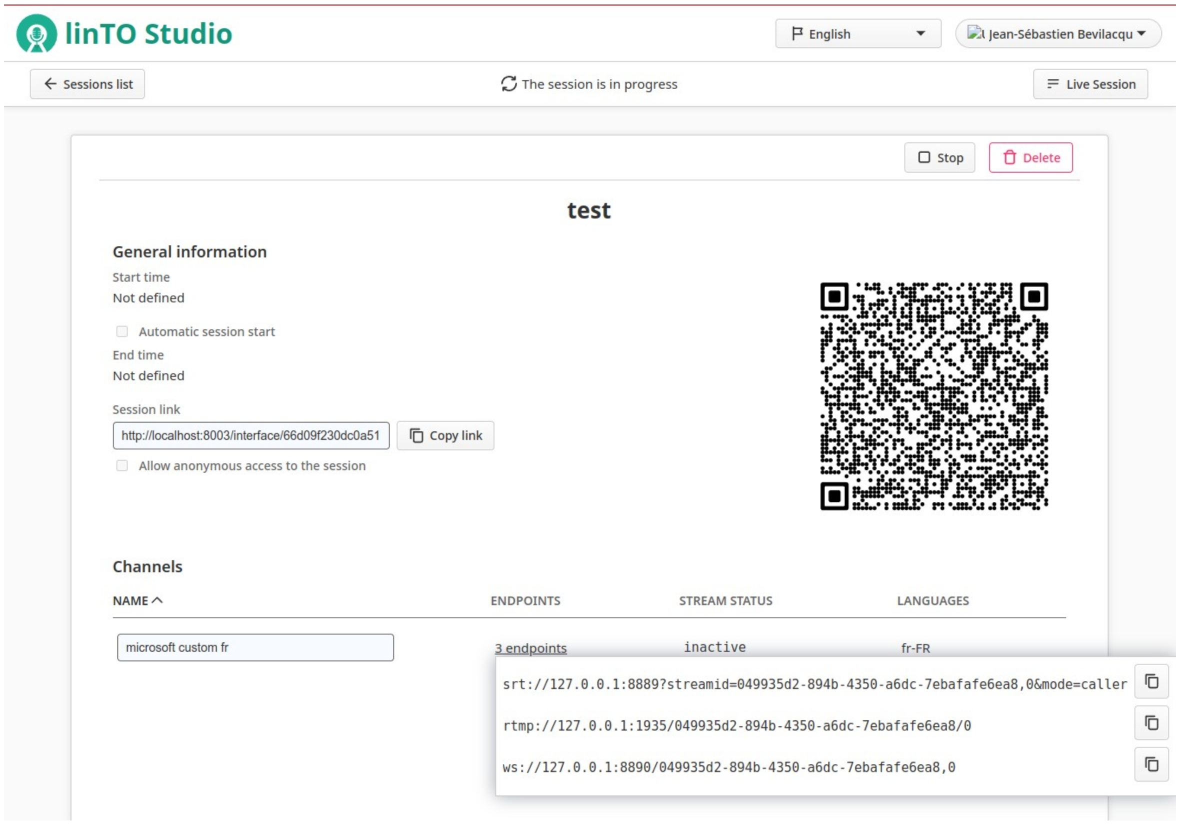
Selection of transcription profiles

SELECTED	TYPE	NAME	DESCRIPTION	LANGUAGES
☐		microsoft_custom_fr	microsoft custom fr	fr-FR

Cancel · Create 0 channels

After clicking on the "Add Channels" button, you will see the different available transcriber profiles. You should see your newly created transcriber profile. Select it and create your session.

5. Detail session view



test

General information

Start time
Not defined

Automatic session start

End time
Not defined

Session link
http://localhost:8003/interface/66d09f230dc0a51

Allow anonymous access to the session

Channels

NAME	ENDPOINTS	STREAM STATUS	LANGUAGES
microsoft custom fr	3 endpoints	inactive	fr-FR

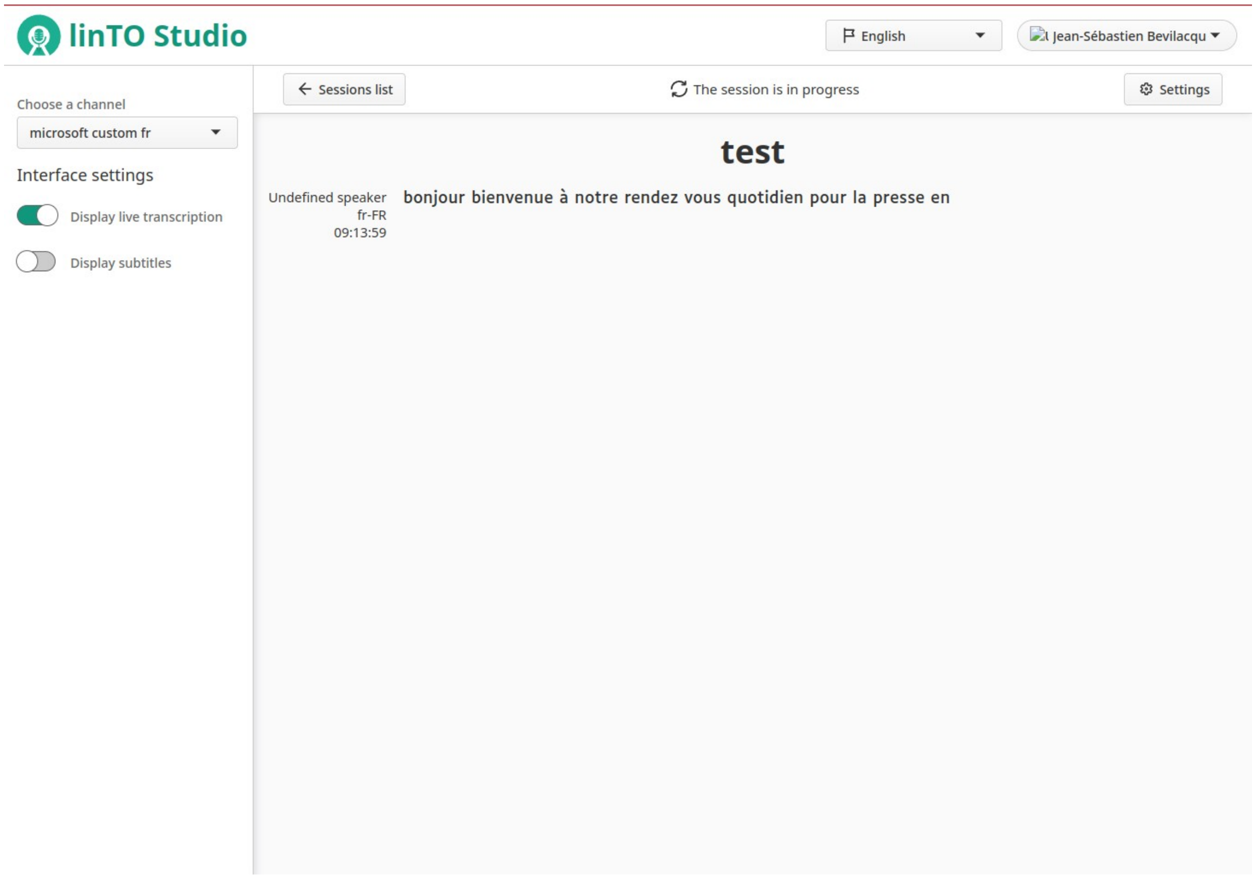
```
srt://127.0.0.1:8889?streamid=049935d2-894b-4350-a6dc-7ebafafe6ea8,0&mode=caller
rtmp://127.0.0.1:1935/049935d2-894b-4350-a6dc-7ebafafe6ea8/0
ws://127.0.0.1:8890/049935d2-894b-4350-a6dc-7ebafafe6ea8,0
```

You will then arrive at the session view. Here, you can find the link to the session, along with its channels and endpoints. A QR code is also generated for direct access to the session, which can be easily shared. As you can see, it's possible to send a stream via SRT, RTMP, or WebSocket. In our example, we will send a stream using SRT.
In the header, you can see that the session is live. Click on the "Live Session" button to view real-time transcriptions.

We will now send an SRT stream using the following command:

```
gst-launch-1.0 filesrc location=./fr.mp3 ! decodebin ! audioconvert !
audioresample ! avenc_ac3 ! mpegtsmux ! rtpmp2tpay ! srtsink
uri="srt://127.0.0.1:8889?streamid=049935d2-894b-4350-a6dc-
7ebafafe6ea8,0&mode=caller"
```

6. Real-Time Transcription



Undefined speaker
fr-FR
09:13:59
bonjour bienvenue à notre rendez vous quotidien pour la presse en

The transcriptions appear in real-time, and your platform is now fully operational!