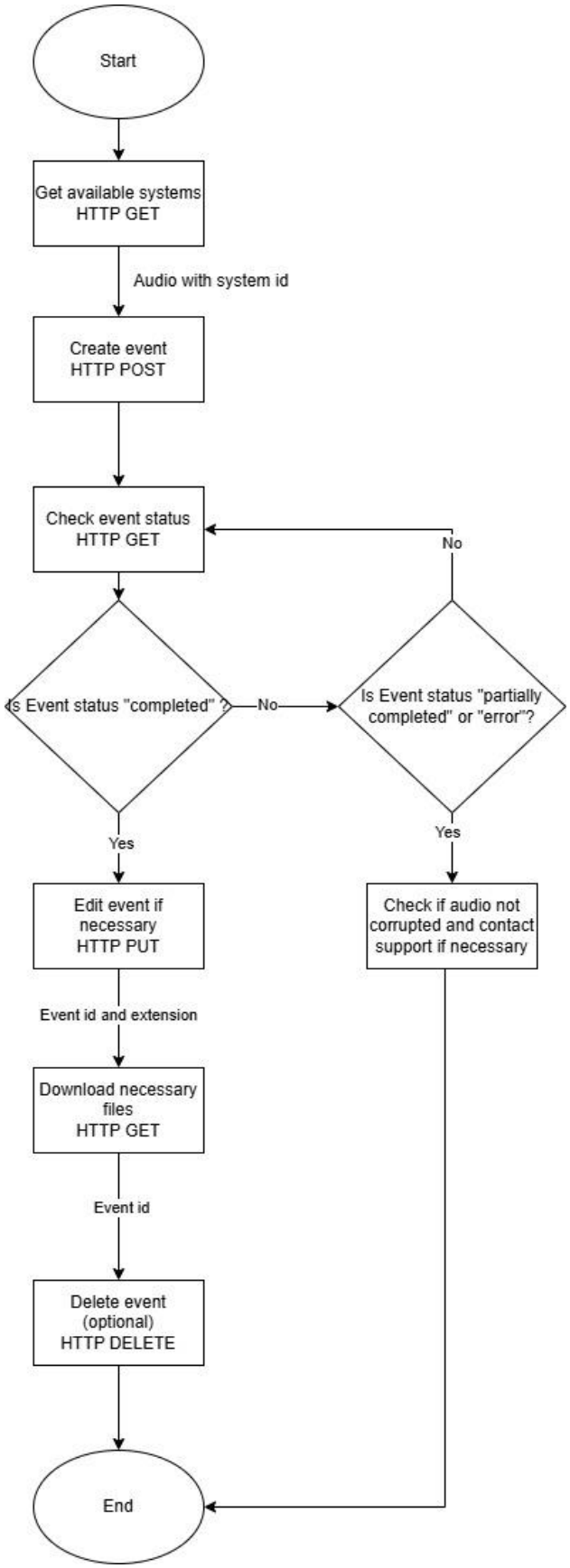
Offline ASR

This document outlines the workflow of offline event processing workflow. The system handles audio events with associated metadata, processing them through various stages, including creation, status checking, potential editing, file downloading, and optional deletion. Here is a short description of simple scenario for transcribing speech to text.

All of the requests mentioned below should contain authorization headers (Api key or token).

All data models mentioned in this document can be found in swagger . Note, that in swagger, method descriptions might contain some additional params for advanced use cases, but here is described scenario for only speech transcription.

Below is simple diagram that displays workflow to get from audio file to its transcript:


Start
Get available systems
HTTP GET
Audio with system id
Create event
HTTP POST
Check event status
HTTP GET
No
Is Event status "completed" ?
No
Is Event status "partially completed" or "error"?
Yes
Yes
Edit event if necessary
HTTP PUT
Check if audio not corrupted and contact support if necessary
Event id and extension
Download necessary files
HTTP GET
Event id
Delete event (optional)
HTTP DELETE
End

Check available systems

You can make request to check available engines. Make request, select system you want to use and remember systemId which will be used for other requests.

Method: *HTTP GET* {apiUrl}/systems?systemType=Offline

Query params: *systemType* - which type of systems should be returned. To return systems only for offline speech recognition, set this param to 'Offline'

Create Event

This request initiates the audio event processing pipeline. The request body contains the audio data and its corresponding metadata. Upon successful creation, the system returns created event object. **EventId from response will be used in other request.**

Method: *HTTP POST* {apiUrl}/event?system={system}

Query params: *system* = System id for selected asr engine.

Formdata: *audio*: File in binary format

Check Event Status

This request retrieves the current status of an event. The event ID is provided as a query parameter. Response contains event object with all necessary event information.

Method: *HTTP GET* {apiUrl}/event/{eventId}

Params: *eventId* - Event Id for which you want to check status.

Edit Event (Optional)

It is possible to edit event, by modifying its json content. Usually it is done through UI. Modifying through api is risky and can lead to unexpected errors, so be careful to keep json in correct format.

Method: *HTTP PUT* {apiUrl}/event/{eventId}

Params: *eventId* - Event Id which you want to update

Request payload
Event json. Check model in swagger.

Download Necessary Files

This requests allows to download transcription in different formats. Currently supported formats: _txt, vtt, srt, docm, json.

Method: *HTTP GET* {apiUrl}/event/{eventId}/srt?edited=true&diarization=false

Params: *eventId* - event id for which you want to download files.

Query params:
edited - set this to true/false based on if you want to get edited version of document. If file is not

edited, but value set to true - you will get original file.
diarization - whether speakers should be included in transcript. Currently supported for docm.

Delete Event (Optional)

This request permanently deletes an event.

Method: *HTTP DELETE* {apiUrl}/event/{eventId}

Params: *eventId* - event id which you want to delete.

List all files

To return all files associated with user, you can call list method.

Method: *HTTP GET* {apiUrl}/event/list