API Reference Guide

AudioCodes Speech-to-Text Transcription Engine for Voice.AI

ATS Speaker Recognition API

Version 0.12

Table of Contents

Notice

Information contained in this document is believed to be accurate and reliable at the time of printing. However, due to ongoing product improvements and revisions, AudioCodes cannot guarantee accuracy of printed material after the Date Published nor can it accept responsibility for errors or omissions. Updates to this document can be downloaded from https://www.audiocodes.com/library/technical-documents.

This document is subject to change without notice.

Date Published: March-27-2024

Customer Support

Customer technical support and services are provided by AudioCodes or by an authorized AudioCodes Service Partner. For more information on how to buy technical support for AudioCodes products and for contact information, please visit our website at https://www.audiocodes.com/services-support/maintenance-and-support.

Stay in the Loop with AudioCodes

Abbreviations and Terminology

Each abbreviation, unless widely used, is spelled out in full when first used.

Related Documentation

Document Name
LTRT-26001 AudioCodes Automatic Speech Recognition – WebSocket API (v0.49)
LTRT-26002 AudioCodes Speech – LVCSR WebSocket API (v0.59)
LTRT-26003 AudioCodes Speech REST API (v0.5)

Document Revision Record

LTRT	Description
26004	Initial document release for Version 0.11.
26013	Added different APIs to documentation for Version 0.12.

Documentation Feedback

AudioCodes continually strives to produce high quality documentation. If you have any comments (suggestions or errors) regarding this document, please fill out the Documentation Feedback form on our website at https://online.audiocodes.com/documentation-feedback.

1 Introduction

This document describes how to interact with AudioCodes' Speaker Segmentation technology suitable for various use cases offline. The API exposes technology that provides dual functionality, allowing enrollment using previous voiceprints and/or dominant speaker segmentation as input and output speaker segmentation and updated voiceprints. The technology operates via WebSocket-based protocol API, which is explained below. The document provides details of LVCSR session interaction including the parameters that govern the process.

Speaker enrollment and segmentation WebSocket endpoint

```
ws[s]://<server IP>:<port>/api/v1/speech: SREnrollAndSegment
```

2 AudioCodes Speaker Recognition Application guidelines

The following guidelines provided primarily focus on the Enroll & Segment APIs and similarity APIs. Additionally, the diarization REST and WebSocket APIs are for more generic transcription services use case. The Enrollment API and Segmentation API address less common use cases.

2.1 Introduction & Considerations

This section highlights the main features of the *Speaker Recognition* (SR) technology.

- **Multi-speaker enrollment**: The SR server is capable of enrolling multiple speakers from a single audio recording, without the need to fragment the audio into each speaker separately.
- **Enhancement of external speaker segmentation**: The SR enrollment API exports an enhanced version of the given external speaker segmentation, if given, (e.g., MSFT Dominant Speaker), which is better aligned to the audio on a speaker-dependent basis.
- **Multiple speakers on the same device**: The case of speaker enrollment in the presence of multiple speakers in the same audio channel, and/or when external audio/video is shared, is supported.
- **Verification of enrollment speech**: A layer of speaker verification is activated during the enrollment process, to ensure the right speaker ID is assigned to each voiceprint, whenever possible.
- **Automatic enrollment of anonymous speakers**: Voiceprints are also created for speakers who the SR server is unsure about their identity.
- **Enrollment-and-Segmentation**: Support in segmentation cut-through, allowing speaker enrollment and segmentation in a single API call.
- **Voiceprint management**: APIs for voiceprint management, enabling the users of the application to control the health and resolve issues with their voiceprints.

2.1.1 General Considerations

The SR server and the I/O of the full system are different and therefore it is necessary to provide some mediation utilities to overcome the following:

- **Audio file support**: The system uses raw audio files (header-less) and there is a necessity to stream them to the SR server.
- **Voiceprint storage**: Due to data privacy (GDPR, etc.), the system must keep the speaker voiceprints in a secure place, and not on the SR server. A voiceprint is textual file, and its expected size is at most 4kB (not constant). The voiceprint size may be larger in future releases.
- **External speaker segmentation**: Some conferencing system (e.g., MSFT Teams) provide additional metadata about speaker or voiced segmentation that can be used in the speaker enrollment and segmentation processes. Per enrollment, the API allows (optionally) to suggest the speaker ID per segment, where segments that do not belong to any speaker now being enrolled are ignored.
- **Segmentation units**: Speaker segmentation is given in units of 10msec frames. The application must adjust to this unit, in all APIs including external segmentation to enrollment/segmentation and in segmentation results.
- **Stateless server**: The SR server is stateless, and all information is encapsulated inside the voiceprints the application manages and stores.

2.2 Enrollment and Segmentation Flow Charts

This section describes graphically the application guidelines for handling the meeting recordings and initiating enrollment and segmentation processes. Definitions are given in this section; details are specified in the rest of the sections.

2.2.1 I/O Diagram

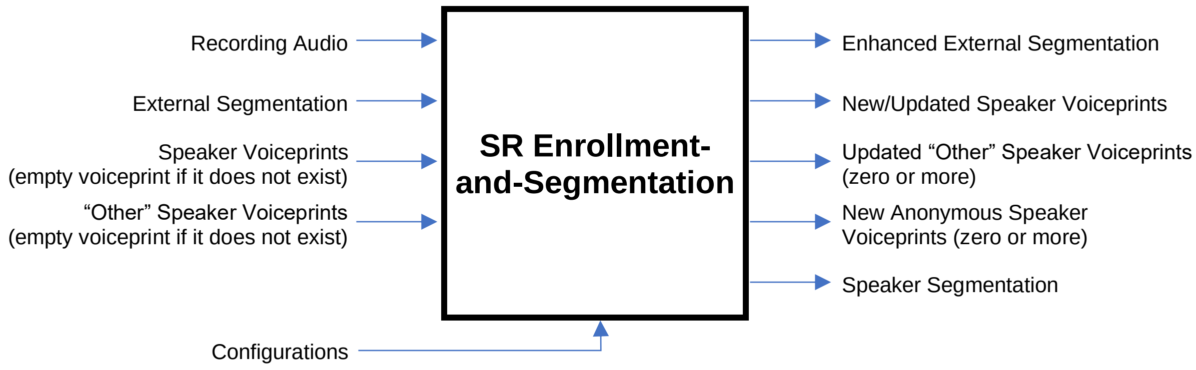


Figure 1 - SR Enrollment-and-Segmentation I/O diagram

2.2.2 Definitions

- A (normal) speaker is a participant that is connected to the meeting.
  - A speaker may or may not speak during the meeting.
  - A conference room entity (e.g., Gilboa Conference Room) is also considered as a normal speaker. **Its ID must start with an asterisk** (*) and its voiceprint **must** be empty.
- An "other" speaker is a scheduled participant (an invitee) that did not connect to the meeting (see section 2.3, "Enrollment Management" for more details).
- A voiceprint may be **empty**, **zero**, or **non-zero**. An **empty** voiceprint is a voiceprint that contains only metadata (e.g., speaker ID), and no data. A **zero** voiceprint is a non-empty voiceprint with a zero matureness score (see section 2.3, "Enrollment Management"). Conversely, a **non-zero** voiceprint is a non-empty voiceprint with non-zero matureness score. The SR server treats empty voiceprints and zero voiceprints equally.
- The external speaker segmentations specify which audio segments have speech activity.
  - In case of single-speaker enrollment, this input is optional, and the segmentation may or may not indicate the speaker ID for each segment. If speaker IDs are indicated, then only the segments that match the ID of the enrolling speaker are taken into consideration.
  - In case of multi-speaker enrollment, this input is obligatory, and the segmentation **must** indicate the speaker ID (or the conference room ID) for each segment.
- Configurations: see next subsection.

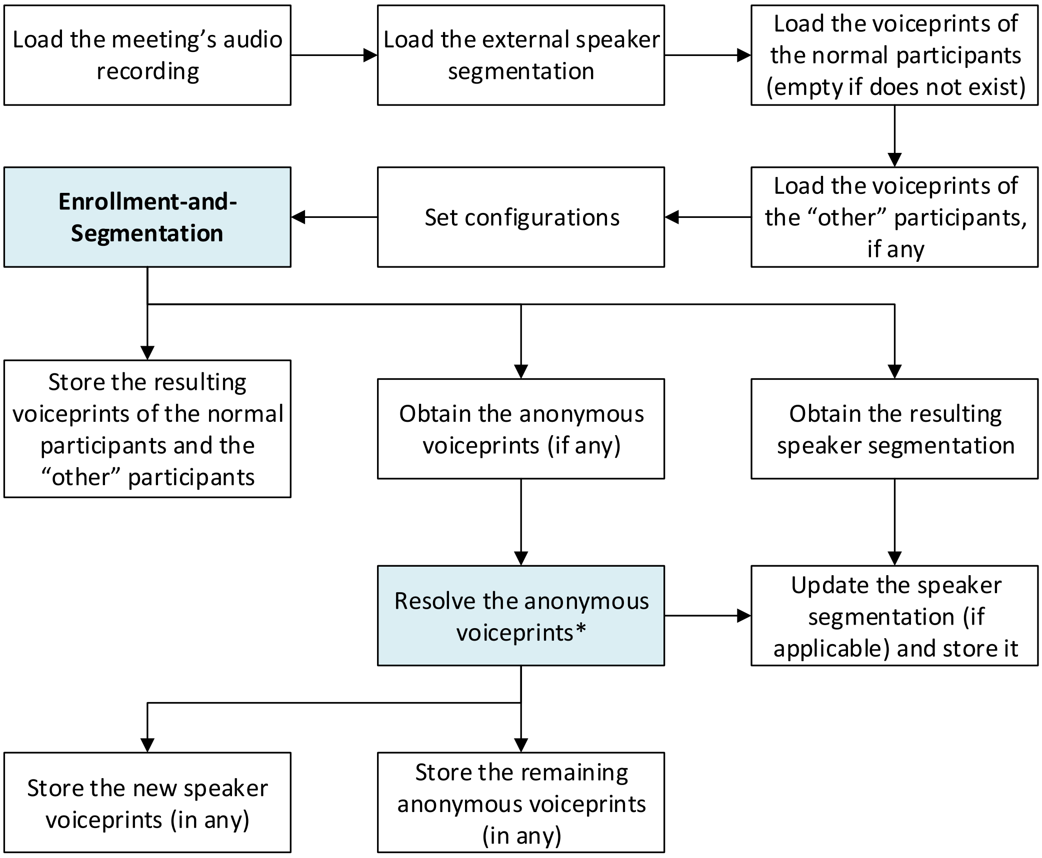
2.2.3 Configurations

The applicable configurations are:

- Automatic ID Assignment Mode: enable / disable. See corresponding section for details.

2.2.4 Application Flow

The chart below shows a simplified application flow for handling the meeting recordings.



* Refer to see section 2.5.2, "Resolving Anonymous Voiceprints and Figure 4 therein.

Figure 2 - Enrollment-and-Segmentation application flow

2.3 Enrollment Management

The following is a list of application guidelines regarding the speaker enrollment process.

- **Online meetings**: Speaker enrollment is recommended for online meetings only, where most speakers use their personal device to remotely connect to the meeting.
- **Conference room meetings**:
  - Generally, the case of conference room meetings is not supported: Speaker enrollment from conference room meetings is **not** recommended, and the recognition accuracy of the SR server in this case is not assured.
  - The SR server automatically detects conference room meetings by searching for an asterisk sign (*) at the beginning of the speaker ID. When detected, updating of the speaker voiceprints is prevented.
  - The recommended way for the application to handle conference room meetings is to request Enrollment-and-Segmentation. The application **must** ensure that the conference room is included as a speaker in the list of participants and that its ID starts with an asterisk (*). External speaker labels with speaker IDs **must** be provided, even if all speech segments are marked with the ID of the conference room. Participants who did not connect to the meeting using their personal device are indicated as "other" speakers (see 'Scheduled participants' bullet at the bottom of this list).
- **Company meetings**: The case of a company meeting held physically, e.g., in an auditorium in the presence of hundreds of employees, was not evaluated; the SR performance is unpredictable. The case of online company meeting was not evaluated as well, but the SR server is expected to retain the same recognition accuracy rate as in any other online meeting.
- **Multi-speaker enrollment**: Usually, there are multiple speakers in a recording. There is **no need** to fragment a recording into speaker-specific fragments to control the enrollment process and digest each speaker separately. The SR server is capable of multiple-speaker enrollment in a single API call.
- **ENROLLED state**: A speaker voiceprint can be in one of two states: ENROLLING and ENROLLED. The current recommendation is to always keep enrolling the speaker, so its voiceprint follows the specific speaker voice changes over time. Therefore, ENROLLED state is **not** reflected; only ENROLLING state is reflected.
- **Voiceprint matureness**: The enrollment API provides a metric of the voiceprint matureness in the form of a score ranging from 0-100, as specified in the table below.

  Table 1: Voiceprint Matureness

Score	State	Quality
0	ENROLLING	N/A
1-25	ENROLLING	Poor
26-50	ENROLLING	Fair
51-75	ENROLLING	Good
76-99	ENROLLING	Excellent

  We term a voiceprint with a matureness score of 0 a **zero voiceprint**, and a voiceprint with a matureness score greater than 0 a **non-zero voiceprint**.
- **Scheduled participants** (denoted in the APIs as "other" speakers): One possible scenario is that a scheduled participant did not join the meeting using his/her own device, but instead physically joins a colleague who did connect to the meeting. To cope with this scenario, indications about scheduled participants that did not connect to the meeting are provided to the SR server during the enrollment process.

- **Contamination by multiple speakers**: It is absolutely forbidden to mix between speakers during the enrollment process. In such a case, it is recommended to enroll from scratch or from the point the voiceprint was corrupted.

2.4 Speaker Segmentation Guidelines

The recommendation is to initiate an **Enrollment-and-Segmentation** process for every meeting, to achieve the best accuracy the SR server can provide. Initiating an enrollment process and then initiating a separate segmentation process is possible, but it results in a lower recognition accuracy, with respect to the recommended method.

2.5 Voiceprint Management

Generally, the SR server creates and updates the speaker voiceprints automatically, whenever possible. The application is responsible only for storing the voiceprints.

In addition, an enrollment process may yield anonymous voiceprints when the SR server is unsure of the speaker identity. This section gives application guidelines to resolve anonymous voiceprints or cases of speaker voiceprint corruption.

2.5.1 Automatic ID Assignment Mode

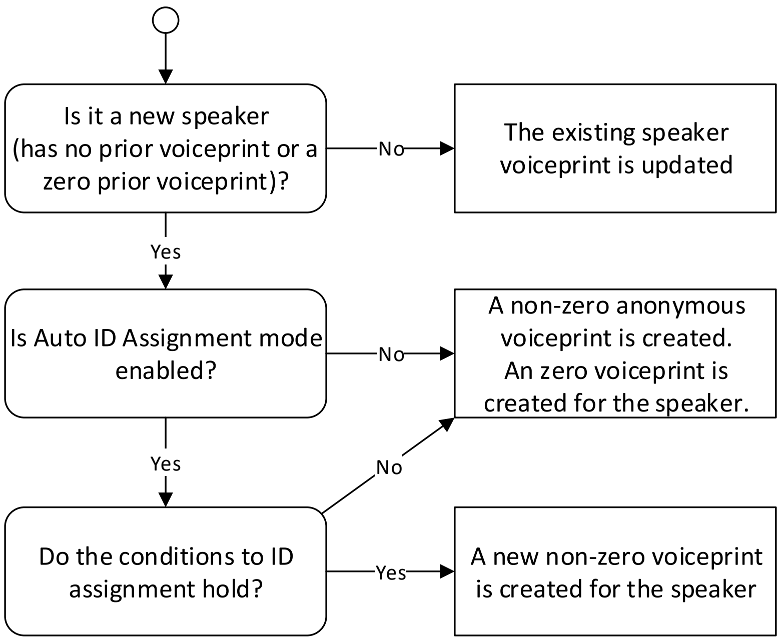


Figure 3 – SR server flow for assigning a speaker ID to a voiceprint

The *Automatic ID assignment* mode can be configured to either **enable** (default) or **disable**.

This mode is relevant **only** for "new" speakers, for which there is no prior voiceprint or only a zero voiceprint (zero matureness score). When **enabled**, and the SR server encounters a "new" speaker, it may create a non-zero voiceprint for him/her if certain conditions hold:

- This is an online meeting.
- All scheduled participants joined the meeting.
- Only a single speaker is detected in the speech segments of the designated speaker.

If these conditions do **not** hold, or if the Automatic ID assignment mode is **disabled**, then **a zero voiceprint is always created for the "new" speaker**. Instead, an anonymous voiceprint is created, and this voiceprint needs to be resolved (see next section: Resolving Anonymous Voiceprints). In other words, a speaker ID is assigned to a voiceprint only if the speaker has a non-zero prior voiceprint and he/she was positively identified.

Under rare conditions, the Automatic ID assignment mode can cause a failure, assigning a voiceprint with the wrong speaker ID. Thus, the application has the option to disable this mode to mitigate this possibility (see section 2.5.1.1, "Use Cases").

Figure 3 above gives a graphical representation of the above explanations.

2.5.1.1 Use Cases

An organization in which there are **only online meetings**, and where each participant is remotely connected to the meeting with his/her own personal device, may find the Automatic ID Assignment mode very useful. The speaker voiceprints are automatically created and updated, with only a rare need for manual intervention.

On the other end, an organization in which **all meetings are physically** held in conference rooms, should work only with the manual mode. Anonymous voiceprints should be constantly resolved by manual means, until all employees and regular guests[1] are enrolled (have voiceprints).

In between, **a hybrid organization** that have both kinds of meetings, can choose between automatic mode and manual mode as suited. Claims for and against each mode should consider what is the most common type of meeting and how much additional resources can be invested in manual operations.

[1] Regular guest is a person, out of the organization, that has recurring interactions with the organization. One-time guests, on the other hand, will always require manual ID assignment, regardless of the ID assignment mode (automatic or manual), unless leaving them as anonymous is acceptable.

2.5.2 Resolving Anonymous Voiceprints

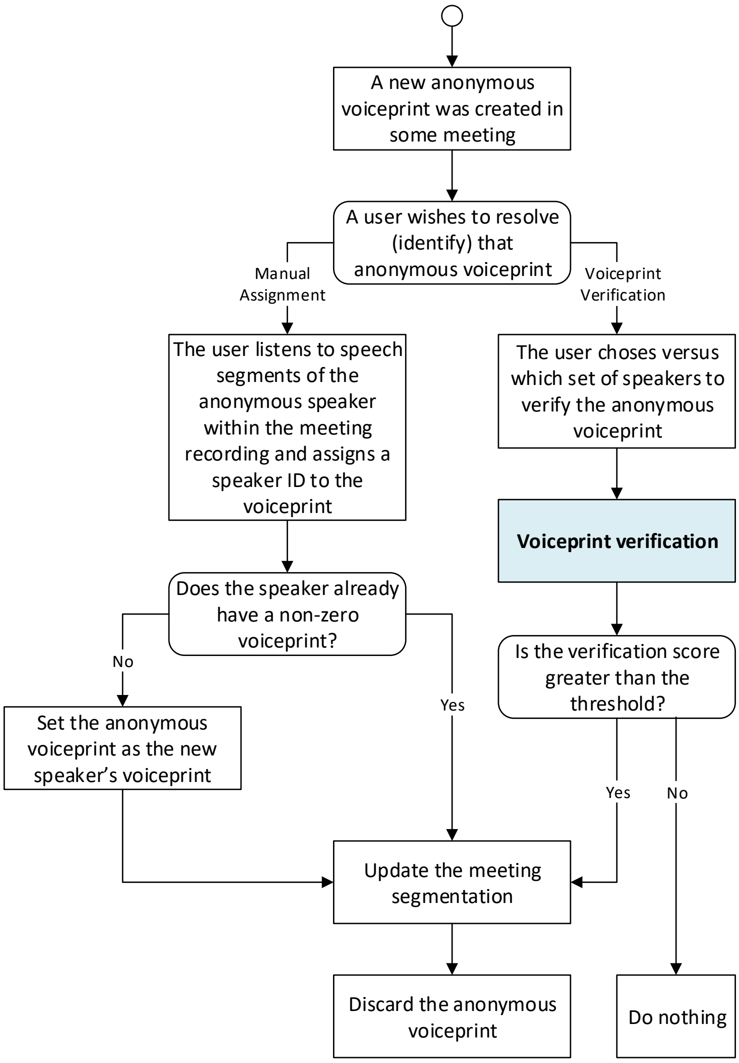


Figure 4 – Flow for resolving anonymous voiceprints

Anonymous voiceprints are meeting-specific; they are created when the SR server is unsure of the speaker identity. The user has the option to resolve them by assigning a speaker ID to the anonymous voiceprint(s). This can be done manually, by an API which is out of the scope of the SR server, or by verifying the anonymous voiceprint versus an existing speaker voiceprint (see section 4, "Speaker Recognition Similarity Measure - WebSocket API").

Manual assignment: In the first option, the user directly assigns a speaker ID to an anonymous voiceprint, e.g., after listening to the meeting recording. As a result of this action, the speaker segmentation of the corresponding meeting is updated. In addition, if the speaker has no existing voiceprint (or only a zero voiceprint), then this voiceprint is set as his/her voiceprint (if the speaker already has a non-zero voiceprint, then the anonymous voiceprint can be simply discarded).

Voiceprint verification: In the second option, the user verifies one or many anonymous voiceprints versus existing speaker voiceprints with a dedicated API. This is done by scoring the voiceprints and comparing the scores to some threshold to make a decision. If the verification is successful, the meeting's segmentation is updated accordingly (afterwards, the anonymous voiceprint can be simply discarded).

The user that initiates these actions does not have to be the true speaker behind the anonymous voiceprint, nor the meeting owner. It is up to the application to decide who has the privileges to initiate such actions.

2.5.2.1 Example Use Case

An example use case: Two speakers are using the same device to connect to an online meeting (a host speaker who owns the device, and another speaker), but none of them has a prior voiceprint. Assuming both spoke enough, two anonymous voiceprints are born from that meeting, as the SR server cannot know who of which is truly the host speaker. On some later meeting, a voiceprint is created for the host speaker. The user (or the application) can utilize the new voiceprint to verify who of the two anonymous speakers in the first meeting is the host speaker.

2.5.3 Resolving Corrupted Voiceprints

A speaker's voiceprint may be corrupted when built on speech segments of the wrong speaker, on speech segments from multiple speakers, or on non-speech segments (very rare).

Voiceprint corruption causes rejection of the speaker's speech on the best case (marking his/her speech segments as anonymous or as unknown), and misidentification on the worst case (speaker confusion). This situation is unlikely to be resolved by itself, without manual intervention.

The user has two options to resolve this issue: voiceprint-reset, and voiceprint-rebuild. Note that the SR server does not provide a dedicated API for the latter action.

Voiceprint-reset means discarding all the information stored in the voiceprint, creating an empty voiceprint for the speaker. This allows the speaker's voiceprint to have "a fresh start".

Voiceprint-rebuild essentially refers to voiceprint-reset, followed by normal speaker enrollment.

In addition, in case of significant changes in the conditions (network, acoustic, etc.), it is also recommended to enroll from scratch (rebuild the voiceprint) in the presence of new conditions.

It is up to the application to decide who has the privileges to initiate such actions.

3 Enroll & Segment WebSocket API

3.1 Textual message payload to server

- Request to start enrollment and segmentation with parameters governing the process in JSON format.
- Request to stop speaker segmentation in JSON format.
- Base64 based audio streaming.

To start the enrollment and segmentation process, send action **start** along parameters related to the enrollment and with either of the following:

- Existing voiceprints (one or more) to use (to accumulate enrollment) of attendees and other invitees.
- Empty (one or more to create new ones from scratch) of attendees and other invitees.

The API supports a process where it includes multiple speaker enrollment with external speaker activity segments (e.g., from Microsoft Teams) that hints at the speaker identity of a segment. the output is accompanied with enhanced activity segments in addition to speaker segmentation.

To improve the system performance, the API denotes voiceprints that belong to ***attendees*** of the meeting (given under "voiceprints" array in the API, either previous or empty ones) and ***invitees*** (given under "othersvoiceprints" array in the API, either previous or empty ones).

Invitees include speakers who didn't attend the meeting according to the online meeting metadata (not by dominant speaker indication, because there maybe attendees who didn't speak at all according to the indication).

Optionally and in addition, an external speech activity segments indication (e.g., segmentation of words from speech-to-text process or other) could be sent in the API. The diarization algorithm utilizes them.

The speaker speech language is independent and must be set to "xx-yy" specified through the parameter "accept-language".

Enrollment and Segmentation use cases:

No.	Existing Voiceprints (attendees or invitees)	Speaker activity segments	Speaker ID per segments	External speech activity indication (e.g., from STT)	Notes
1	+	+	+	-/+	Enrollment is performed according to external speaker segments that belong to the speaker ID now being enrolled and existing voiceprints are enriched. (Teams use case). External speech activity indication is optional.
2	-	+	+	-/+	Like case No. 1, but with a new voiceprint being created to all speakers (one or more). External speech activity indication is optional
3	-\+	+	+	-/+	Like case No. 1, but some of the voiceprints are new voiceprints being created. External speech activity indication is optional.

Example:

This example starts a request with the following:

- Existing voiceprints (multiple for both attendees and invitees)
- Speaker activity segments
- Speaker id per segment
- Without external speech activity segments


```
{"action":"start","enrollment-control":1,"accept-language":"xx-
yy","content-type":"audio/l16;rate=16000","save-waveform":1,"save-
voiceprint":1,"cookie":"152617477","voiceprints":[{"id":"303F5D25-8F2E-
424D-990D-A0A0DB26E302","data":"
dHBsAGABAABTKGlpKUJCAAEAAAABAAAAgAAAAIU7CKxZmb/tDLaHnXgTIUesaULPOWFDbC3v4
edCCkCvAElU5P35a4BvRvT9RdQ7qiBQzAsUkODnuJI4hms4qTsQZMcCjMU8a5LBdpoEEUe6lM
04lcJR/hUaJ1hiTzgqn4tekX4atjqFzlzJx2dbsDpgjATIdUmxPQ4ww5B1Q5WbwAAAACg6106
t7OtKZtFQRwtl3WkLdVA5YmFXc+KLE7xs+M2dQ4F7mMknSvE9wsDF6h1t2l0sIFmyzT8RAJPo
/09qwf36cOrwm56yShvMrixbfUOBTY73o9ehzzCxS8BfoaEEA+oAIpPIiR3H0A68ajLC5dS0i
tGQCUBe2K9ZkfL6A+8ulj/7G6omexo0+jAdUSURhya5wVmyUCwr4QLhN29HjFVORhZItEpBqv
/g7Gu6A9/cio6W9yNXW8V3TFuMvteO6w=="},{"id":"303F5D25-8F2E-424D-990D-
A0A0DB26EAAA","data":"
dHBsAGABAABTKGlpKUJCAAEAAAABAAAAgAAAAIU7CKxZmb/tDLaHnXgTIUesaULPOWFDbC3v4
edCCkCvAElU5P35a4BvRvT9RdQ7qiBQzAsUkODnuJI4hms4qTsQZMcCjMU8a5LBdpoEEUe6lM
04lcJR/hUaJ1hiTzgqn4tekX4atjqFzlzJx2dbsDpgjATIdUmxPQ4ww5B1Q5WbwAAAACg6106
t7OtKZtFQRwtl3WkLdVA5YmFXc+KLE7xs+M2dQ4F7mMknSvE9wsDF6h1t2l0sIFmyzT8RAJPo
/09qwf36cOrwm56yShvMrixbfUOBTY73o9ehzzCxS8BfoaEEA+oAIpPIiR3H0A68ajLC5dS0i
tGQCUBe2K9ZkfL6A+8ulj/7G6omexo0+jAdUSURhya5wVmyUCwr4QLhN29HjFVORhZItEpBqv
/g7Gu6A9/cio6W9yNXW8V3TFuMvteO6w=="}],"othersvoiceprints":[{"id":"303F5D2
5-8F2E-424D-990D-A0A0DB26EEE","data":"
dHBsAGABAABTKGlpKUJCAAEAAAABAAAAgAAAAIU7CKxZmb/tDLaHnXgTIUesaULPOWFDbC3v4
edCCkCvAElU5P35a4BvRvT9RdQ7qiBQzAsUkODnuJI4hms4qTsQZMcCjMU8a5LBdpoEEUe6lM
04lcJR/hUaJ1hiTzgqn4tekX4atjqFzlzJx2dbsDpgjATIdUmxPQ4ww5B1Q5WbwAAAACg6106
t7OtKZtFQRwtl3WkLdVA5YmFXc+KLE7xs+M2dQ4F7mMknSvE9wsDF6h1t2l0sIFmyzT8RAJPo
/09qwf36cOrwm56yShvMrixbfUOBTY73o9ehzzCxS8BfoaEEA+oAIpPIiR3H0A68ajLC5dS0i
tGQCUBe2K9ZkfL6A+8ulj/7G6omexo0+jAdUSURhya5wVmyUCwr4QLhN29HjFVORhZItEpBqv
/g7Gu6A9/cio6W9yNXW8V3TFuMvteO6w=="},{"id":"303F5D25-8F2E-424D-990D-
A0A0DB26EFFF","data":"
dHBsAGABAABTKGlpKUJCAAEAAAABAAAAgAAAAIU7CKxZmb/tDLaHnXgTIUesaULPOWFDbC3v4
edCCkCvAElU5P35a4BvRvT9RdQ7qiBQzAsUkODnuJI4hms4qTsQZMcCjMU8a5LBdpoEEUe6lM
04lcJR/hUaJ1hiTzgqn4tekX4atjqFzlzJx2dbsDpgjATIdUmxPQ4ww5B1Q5WbwAAAACg6106
t7OtKZtFQRwtl3WkLdVA5YmFXc+KLE7xs+M2dQ4F7mMknSvE9wsDF6h1t2l0sIFmyzT8RAJPo
/09qwf36cOrwm56yShvMrixbfUOBTY73o9ehzzCxS8BfoaEEA+oAIpPIiR3H0A68ajLC5dS0i
tGQCUBe2K9ZkfL6A+8ulj/7G6omexo0+jAdUSURhya5wVmyUCwr4QLhN29HjFVORhZItEpBqv
/g7Gu6A9/cio6W9yNXW8V3TFuMvteO6w=="}],"speechsegments":[{"location":100,"
duration":50,"id":"303F5D25-8F2E-424D-990D-
A0A0DB26E302"},{"location":200,"duration":40,"id":"303F5D25-8F2E-424D-
990D-A0A0DB26E302"},{"location":300,"duration":30,"id":"303F5D25-8F2E-
424D-990D-A0A0DB26E302"},,{"location":350,"duration":30,"id":"303F5D25-
8F2E-424D-990D-A0A0DB26EAAA"}]}
```


> The example above meets the Microsoft Teams use case.

Example:

This example starts a request with the following:

- Empty and existing voiceprints (multiple)
- Speaker activity segments
- Speaker ID per segment
- Without external speech activity segments


```
{"action":"start","enrollment-control":1,"accept-language":"xx-
yy","content-type":"audio/l16;rate=16000","save-waveform":1,"save-
voiceprint":1,"cookie":"152617477","voiceprints":[{"id":"303F5D25-8F2E-
424D-990D-A0A0DB26E302","data":"
dHBsAGABAABTKGlpKUJCAAEAAAABAAAAgAAAAIU7CKxZmb/tDLaHnXgTIUesaULPOWFDbC3v4
edCCkCvAElU5P35a4BvRvT9RdQ7qiBQzAsUkODnuJI4hms4qTsQZMcCjMU8a5LBdpoEEUe6lM
04lcJR/hUaJ1hiTzgqn4tekX4atjqFzlzJx2dbsDpgjATIdUmxPQ4ww5B1Q5WbwAAAACg6106
t7OtKZtFQRwtl3WkLdVA5YmFXc+KLE7xs+M2dQ4F7mMknSvE9wsDF6h1t2l0sIFmyzT8RAJPo
/09qwf36cOrwm56yShvMrixbfUOBTY73o9ehzzCxS8BfoaEEA+oAIpPIiR3H0A68ajLC5dS0i
tGQCUBe2K9ZkfL6A+8ulj/7G6omexo0+jAdUSURhya5wVmyUCwr4QLhN29HjFVORhZItEpBqv
/g7Gu6A9/cio6W9yNXW8V3TFuMvteO6w=="},{"id":"303F5D25-8F2E-424D-990D-
A0A0DB26EAAA","data":""}],"othersvoiceprints":[{"id":"303F5D25-8F2E-424D-
990D-A0A0DB26E232","data":"
dHBsAGABAABTKGlpKUJCAAEAAAABAAAAgAAAAIU7CKxZmb/tDLaHnXgTIUesaULPOWFDbC3v4
edCCkCvAElU5P35a4BvRvT9RdQ7qiBQzAsUkODnuJI4hms4qTsQZMcCjMU8a5LBdpoEEUe6lM
04lcJR/hUaJ1hiTzgqn4tekX4atjqFzlzJx2dbsDpgjATIdUmxPQ4ww5B1Q5WbwAAAACg6106
t7OtKZtFQRwtl3WkLdVA5YmFXc+KLE7xs+M2dQ4F7mMknSvE9wsDF6h1t2l0sIFmyzT8RAJPo
/09qwf36cOrwm56yShvMrixbfUOBTY73o9ehzzCxS8BfoaEEA+oAIpPIiR3H0A68ajLC5dS0i
tGQCUBe2K9ZkfL6A+8ulj/7G6omexo0+jAdUSURhya5wVmyUCwr4QLhN29HjFVORhZItEpBqv
/g7Gu6A9/cio6W9yNXW8V3TFuMvteO6w=="},{"id":"303F5D25-8F2E-424D-990D-
A0A0DB26EDDD","data":""}],"speechsegments":[{"location":100,"duration":50
,"id":"303F5D25-8F2E-424D-990D-
A0A0DB26E302"},{"location":200,"duration":40,"id":"303F5D25-8F2E-424D-
990D-A0A0DB26E302"},{"location":300,"duration":30,"id":"303F5D25-8F2E-
424D-990D-A0A0DB26E302"},,{"location":350,"duration":30,"id":"303F5D25-
8F2E-424D-990D-A0A0DB26EAAA"}]}
```

Example:

This example starts a request with the following:

- Empty and existing voiceprints (multiple)
- Speaker activity segments
- Speaker ID per segment
- External speech activity segments


```
{"action":"start","enrollment-control":1,"accept-language":"xx-
yy","content-type":"audio/l16;rate=16000","save-waveform":1,"save-
voiceprint":1,"cookie":"152617477","voiceprints":[{"id":"303F5D25-8F2E-
424D-990D-A0A0DB26E302","data":"
dHBsAGABAABTKGlpKUJCAAEAAAABAAAAgAAAAIU7CKxZmb/tDLaHnXgTIUesaULPOWFDbC3v4
edCCkCvAElU5P35a4BvRvT9RdQ7qiBQzAsUkODnuJI4hms4qTsQZMcCjMU8a5LBdpoEEUe6lM
04lcJR/hUaJ1hiTzgqn4tekX4atjqFzlzJx2dbsDpgjATIdUmxPQ4ww5B1Q5WbwAAAACg6106
t7OtKZtFQRwtl3WkLdVA5YmFXc+KLE7xs+M2dQ4F7mMknSvE9wsDF6h1t2l0sIFmyzT8RAJPo
/09qwf36cOrwm56yShvMrixbfUOBTY73o9ehzzCxS8BfoaEEA+oAIpPIiR3H0A68ajLC5dS0i
tGQCUBe2K9ZkfL6A+8ulj/7G6omexo0+jAdUSURhya5wVmyUCwr4QLhN29HjFVORhZItEpBqv
/g7Gu6A9/cio6W9yNXW8V3TFuMvteO6w=="},{"id":"303F5D25-8F2E-424D-990D-
A0A0DB26EAAA","data":""}],"othersvoiceprints":[{"id":"303F5D25-8F2E-424D-
990D-A0A0DB26E232","data":"
dHBsAGABAABTKGlpKUJCAAEAAAABAAAAgAAAAIU7CKxZmb/tDLaHnXgTIUesaULPOWFDbC3v4
edCCkCvAElU5P35a4BvRvT9RdQ7qiBQzAsUkODnuJI4hms4qTsQZMcCjMU8a5LBdpoEEUe6lM
04lcJR/hUaJ1hiTzgqn4tekX4atjqFzlzJx2dbsDpgjATIdUmxPQ4ww5B1Q5WbwAAAACg6106
t7OtKZtFQRwtl3WkLdVA5YmFXc+KLE7xs+M2dQ4F7mMknSvE9wsDF6h1t2l0sIFmyzT8RAJPo
/09qwf36cOrwm56yShvMrixbfUOBTY73o9ehzzCxS8BfoaEEA+oAIpPIiR3H0A68ajLC5dS0i
tGQCUBe2K9ZkfL6A+8ulj/7G6omexo0+jAdUSURhya5wVmyUCwr4QLhN29HjFVORhZItEpBqv
/g7Gu6A9/cio6W9yNXW8V3TFuMvteO6w=="},{"id":"303F5D25-8F2E-424D-990D-
A0A0DB26EDDD","data":""}],"speechsegments":[{"location":100,"duration":50
,"id":"303F5D25-8F2E-424D-990D-
A0A0DB26E302"},{"location":200,"duration":40,"id":"303F5D25-8F2E-424D-
990D-A0A0DB26E302"},{"location":300,"duration":30,"id":"303F5D25-8F2E-
424D-990D-A0A0DB26E302"},{"location":350,"duration":30,"id":"303F5D25-
8F2E-424D-990D-
A0A0DB26EAAA"}],"activitysegments":[{"location":51,"duration":48,"label":
"אבל"},{"location":99,"duration":27,"label":"היה"}]}
```


Several media types are supported via "content-type":

- audio/l16;rate=8000
- **audio/l16;rate=16000 (recommended)**
- audio/pcma;rate=8000
- audio/pcmu;rate=8000
- audio/pcma;rate=16000
- audio/pcmu;rate=16000

To process stereo streams, the media type must indicate the number of channels, otherwise, it is considered as a mono stream.

e.g., audio/PCMU;rate=16000**;channels=2**

optionally media type may indicate explicitly that the stream is mono by denoting channels is one, e.g., audio/PCMU;rate=16000**;channels=1**

if stereo stream was indicated in API, a mix down stereo to mono operation is employed.

> segments (*speechsegments*, *speakersegments* and *activitysegments*) in API (command or message) are given in frames, where frame is 10 [msec].

3.2 Textual message payload from server

The server notifies of any status change, the final status is always **ABORTED**, this status indicates that the process has ended.

You must check the **error** token field **code**. Any value different than 0 indicates a problem in enrollment.

An error code 0 means no error and voiceprint is returned in the response. For logging purposes, the voiceprint id is given in the JSON field **name**.

The voiceprint cannot be used in the recognition process before the JSON field **state** is **ENROLLING** or **ENROLLED**. The **score JSON** field suggests the voiceprint matureness and if there is sufficient or insufficient speech for the speaker, to decide if it can be used in segmentation. See Table 1 in section 2.3 for **score** ranges ranking and recommendations.

Example: enrollment (status change along a real flow)


```
{"name":"1d856bd0-04b3-4dc8-bec3-
a85bcdad1d1d","type":"audiocodes.speech.SRESOperation","status":"READY","
cookie":"152617477"}
{"name":"1d856bd0-04b3-4dc8-bec3-
a85bcdad1d1d","type":"audiocodes.speech.SRESOperation","error":{},"status
":"ABORTED","cookie":"152617477","response":{"voiceprints":[{"id":"303F5D
25-8F2E-424D-990D-
A0A0DB26E302","data":"Wu55ekCBNCnLU5+FJD480hBAIKJoIgScyVeae4wnHVr5DSwut13
YlIcp/M0zZs0GwEux3ev6VSLUYvna9GQPilQ0BbIBA1+Xu9sIRe3K6tXiQrzsNcAbhIms3e97
gKIY1S2rDQS78LrNL2z3O7G8xtzu6NUtvIFLUvzswQrF/Y3jv8hHqsyzhibu0OsGZWphFxfeo
ro1rP6WHM5YinlJceWxAWnagNGCLuDbKDQsHS77hUZbxnXHJbzSLLzh5a1KRcwAeXwdcjdNzD
a2rk/9tvH/
==","state":"ENROLLING","score":49}],"speechsegments":[{"location":100,"d
uration":50,"id":"303F5D25-8F2E-424D-990D-
A0A0DB26E302"},{"location":200,"duration":40,"id":"303F5D25-8F2E-424D-
990D-A0A0DB26E302"},{"location":300,"duration":30,"id":"303F5D25-8F2E-
424D-990D-A0A0DB26E302"},{"location":350,"duration":30,"id":"303F5D25-
8F2E-424D-990D-A0A0DB26EAAA"}]
}}
```


3.3 Packing the voiceprint bytes as payload to server

The binary voiceprint is 64-bit encoded as a string and set to the **data** field under **voiceprints** array in the JSON text message payload. The **ID** is set by the client using arbitrary string to application specific requirements.

3.4 Enrollment parameters

Currently, there is a single parameter governing the enrollment process, namely ***enrollment-control***. The possible values and behavior are as follows:

- "enrollment-control": 0

  Non-automatic resolving of anonymous suspected speaker voiceprints, that must be resolved beyond the scope of the API.
- Otherwise (the default or given explicitly in API command "***enrollment-control***" : 1)

Automatic resolving of anonymous suspected speaker voiceprints.

Please refer to section 2.5.1 for more information.

The server allows a string value to be passed along with text messages (this is called ***cookie)***. The ***cookie*** is present in all subsequent server text messages resulting from the process **start** message. You can use it to identify the particular process within multiple asynchronous processes.

The server, for **debug purposes only** (due to sensitivity of the information), allows you to set:

save-waveform – save the session audio recording as received from the client.

save-voiceprint – save the session last voiceprint being enrolled.

3.5 Binary message payload to server

- Streaming audio in multiple chunks. Audio chunk size should equal to or be greater than 210 msec.
- When working from files, it is recommended to stream in larger chunks (e.g., 10 sec), to reduce the transfer time to the server.
- Zero bytes payload indicates to the server that the streaming ended (usually from EOF).

3.6 Text message for base64 based audio streaming to server

Streaming audio in multiple chunks utilizing base64 text audio streaming:

```
{"action":"stream",
"text":"kdXUUpEQkpmd3ENGEgZiBhZmE2MzQ1MjQyMXRycXdzZA==")
```

Zero bytes equivalent payload in base64 based audio streaming indicates to the server the streaming ended (usually from EOF indication)

```
{"action":"stream", "text":""})
```

3.7 Session Termination

Whenever the application wishes to stop the enrollment process, stop the process by either one of the methods below:

- Issue “zero” bytes message by either:
  - Issue zero bytes binary payload.
  - Issue empty text json “text” property in base64 text-based audio streaming text message.

  This results in processing all the samples, indicating end of file. Use it when processing from files.
- Send "stop" request - this results in processing the speech samples received already in the server (without further samples that may be still transmitted)
- Send “abort” request – this results in the server stopping immediately without processing the speech samples any further.

Example: request from client (stop request)

```
{"action":"stop"," cookie":"enrolling me"}
```

3.8 Control message payload to server

For long period operations without active transmissions such as enrollment or segmentation of large audio files being transmitted and waiting a long time for the server to answer (of textual and/or binary messages between client and server), it is recommended to ping periodically, every few seconds (see reference in unit test supplied). Ping message to the server, that is defined in the WebSocket protocol (https://tools.ietf.org/html/rfc6455#section-5.5 - see section 5.5.2 and 5.5.3). It has been observed in several cases that the WebSocket connection disconnects and is therefore unable to complete the operation.

4 Speaker Recognition Similarity Measure - WebSocket API

Speaker voiceprint similarity WebSocket endpoint

ws[s]://<server IP>:<port>/api/v1/speech: SRSimilarityMeasure

4.1 Understanding Session and Process

In this document, the word ***Session*** refers to a WebSocket session, the term ***Process*** refers to AC Speech server task taking place in an asynchronous way. A process is run employing a session as a means of communication and that session can, once the process is over, be used for executing another process in the same way. Whenever the term status is mentioned, it refers to process status.

4.2 Textual message payload to server

Request to start speaker voiceprint measurements with parameters governing the process in JSON format.

To request a speaker similarity measure task, send action **start** with voiceprints (left) array to be processed and measure each voiceprint similarity against another voiceprint array (right). the speech language is independent but is set through the parameter "accept-language" to constant value xx-yy (reserved for future). After processing as result an array per each voiceprint in left input array is provided with similarity array information against each of the (right) array input voiceprints with absolute score (0 to 100) and akin (boolean value), that can be used to connect voiceprints and segments (e.g., resolving anonymous indicated segments from enrollment and segmentation process).

The purpose of the "**strictness**" integer parameter (value varies between 0 to 100) is to ensure external speakers would not be falsely recognized as a company employee.

when it is known that no external speaker was present in the meeting, "low" value of strictness (less or equal 75) can be used, If the situation is unclear or an external speaker is present in the meeting, it is recommended to use "high" value of strictness (greater than 75).

The binary voiceprint is 64-bit encoded as a string and set to the **voiceprint** field in the JSON text message payload.

Example (start request)


```
{ "action": "start", "strictness" : 100, "accept-language": "xx-
yy", "cookie": "558615043", "lvoiceprints": [ { "id": "303F5D25-
8F2E-424D-990D-A0A0DB26E302", "data": "
AAARESIiMzNERFVVZmZ3d4iImZmQAAEREiIjMzRERVVWZmd3eIiJmZkAABERIiIzM0
REVVVmZnd3iIiZmZAAARESIiMzNERFVVZmZ3d4iImZmQAAEREiIjMzRERVVWZmd3eI
iJmZkA],", "rvoiceprints": [ { "id": "303F5D25-8F2E-424D-990D-
A0A0DB26E302", "data": "
AAARESIiMzNERFVVZmZ3d4iImZmQAAEREiIjMzRERVVWZmd3eIiJmZkAABERIiIzM0
REVVVmZnd3iIiZmZAAARESIiMzNERFVVZmZ3d4iImZmQAAEREiIjMzRERVVWZmd3eI
iJmZkA==" }, { "id": "780F91D6-4119-4C3F-9D24-ABD24BFD09E1",
"data": "
AAARESIiMzNERFVVZmZ3d4iImZmQAAEREiIjMzRERVVWZmd3eIiJmZkAABERIiIzM0
REVVVmZnd3iIiZmZAAARESIiMzNERFVVZmZ3d4iImZmQAAEREiIjMzRERVVWZmd3eI
iJmZkA==" }, { "id": "3E5421AC-48D4-419E-BC93-A677403F5813",
"data": "
ZwiXE4cxCDlwlmJ3FYlwKXdYNyEJVwgpeJcgaJJwiWcpAnq814OYk3YIOXY5djdjhj
l2OXZnhieGJ4lafhjklahjklhaklj25j4klkjaflljfahfja2hClnSJdolziXCXhpg
3iWOZc==" }, { "id": "FD8B36AC-9DFE-4C7F-910D-A1A179A373D4",
```

```
"data": "
ZwiXE4cxCDlwlmJ3FYlwKXdYNyEJVwgpeJcgaJJwiWcpAnq814OYk3YIOXY5djdjhj
l2OXZnhieGJ4lafhjklahjklhaklj25j4klkjaflljfahfja2hClnSJdolziXCXhpg
3iWOZc==" } ] } ]}
```


4.3 Textual message payload from server

From time to time the server notifies the status and results to the client in JSON format. The client should monitor the status and manage the process accordingly.

Example Speech Recognition task (status change along a real flow)


```
{"name":"7a43ed65-91a7-4d51-a0ee-
4bcf39954561","type":"audiocodes.speech.SRSMOperation","error":{},
"status":"STARTED","cookie":"558615043"}
{"name":"7a43ed65-91a7-4d51-a0ee-
4bcf39954561","type":"audiocodes.speech.SRSMOperation","status":"R
EADY","cookie":"558615043"}
{"name":"7a43ed65-91a7-4d51-a0ee-
4bcf39954561","type":"audiocodes.speech.SRSMOperation","error":{},
"status":"TERMINATED","cookie":"558615043"}
```


Example (with results)


```
{"type":"audiocodes.speech.SRSMOperation","name":"7a43ed65-91a7-
4d51-a0ee-
4bcf39954561","cookie":"558615043","status":"ABORTED","response":{
"similarity":[{"id":"303F5D25-8F2E-424D-990D-
A0A0DB26E302","scoring":[{"id":"303F5D25-8F2E-424D-990D-
A0A0DB26E302","score":100.0","akin":true},{"id":"780F91D6-4119-
4C3F-9D24-
ABD24BFD09E1","score":6.842","akin":false},{"id":"3E5421AC-48D4-
419E-BC93-A677403F5813","score":8.454","akin":false}]}]}}
```


4.4 Status Transitions

As can be observed from the examples above, the server reports process status within each text message. The status starts as **READY** right after the process is initialized, it continues with status **STARTED** until the process has ended with status **ABORTED**. In case of failure (some error occurred) the server changes the status to **FAILED**.

It is up to the client to stop the speaker similarity process; this is done as described above in Termination.

The server stops the process upon **stop** request, and this is done in stages.

Once all is processed, the process status changes to **TERMINATED** and when the process is halted completely the status becomes **ABORTED** provided with the similarity results.

4.5 Control message payload to server

In long period operations without active transmission (as could be the case in enrollment or segmentation of large audio files being transmitted and awaiting for long period the server to answer) of textual and\or binary messages in between client and server, it is recommended to send periodically (Every few seconds, see reference in unit test supplied) Ping message to the server, that is defined in the WebSocket protocol (https://tools.ietf.org/html/rfc6455#section-5.5 - see section 5.5.2 and 5.5.3). Otherwise, it has been observed in several cases the WebSocket connection is being disconnected and therefore unable to complete the operation.

5 Speaker Recognition Diarization - WebSocket API

Speaker recognition diarization WebSocket endpoint

ws[s]://<server IP>:<port>/api/v1/speech:SRDiarize

5.1 Understanding Session and Process

In this document, the word ***Session*** refers to a WebSocket session, the term ***Process*** refers to AC Speech server task taking place in an asynchronous way. A process is run employing a session as a means of communication and that session can, once the process is over, be used for executing another process in the same way. Whenever the term status is mentioned, it refers to process status.

5.2 Textual message payload to server

1. Request to start speaker diarization with parameters governing the process in JSON format.
2. Request to stop speaker diarization in JSON format.
3. Base64 based audio streaming.

To request a speaker diarization task, send action **start**, the speech language is independent but is set through the parameter "accept-language" to constant value xx-yy (reserved for future).

Optionally, an external speech activity segments indication (e.g. as result of speech-to-text process or other) could be sent in the API, so the diarization algorithm can utilize them.

Example (start request)

{"action":"**start**","accept-language":"xx-yy","content-type":"audio/l16;rate=16000","save-waveform":1,"cookie":"759550161"}

Example (start request with external activity indication)

```
{"action":"start","accept-language":"xx-yy","content-type":"audio/L16;rate=16000;channels=1","save-waveform":0,"cookie":"2110989790","activitysegments":[{"location":51,"duration":48,"label":"אבל"},{"location":99,"duration":27,"label":"היה"}]}
```

Several media types are supported via "content-type":

- audio/l16;rate=8000
- **audio/l16;rate=16000 (recommended)**
- audio/pcma;rate=8000
- audio/pcmu;rate=8000
- audio/pcma;rate=16000
- audio/pcmu;rate=16000

To process stereo streams, the media type must indicate the number of channels, otherwise, it is considered as a mono stream.

e.g., audio/PCMU;rate=16000**;channels=2**

optionally media type may indicate explicitly that the stream is mono by denoting channels is one, e.g., audio/PCMU;rate=16000**;channels=1**

if stereo stream was indicated in API, a mix down stereo to mono operation is employed.

Textual message payload from server

From time to time the server notifies the status and results to the client in JSON format. The client should monitor the status and manage the process accordingly.

Example Speech Recognition diarization task (status change along a real flow)


```
{"cookie":"759550161","name":"6c774a9f-c394-4137-8783-da181e8f7e7f","status":"READY","type":"audiocodes.speech.SRDOperation","waveform-tag":""}
{"cookie":"759550161","error":{"code":0},"name":"6c774a9f-c394-4137-8783-da181e8f7e7f","status":"STARTED","type":"audiocodes.speech.SRDOperation","waveform-tag":""}
{"cookie": "759550161","error": {"code": 0},"name": "6c774a9f-c394-4137-8783-da181e8f7e7f","status": "TERMINATED","type":"audiocodes.speech.SRDOperation","waveformTag": "languages/xx-yy/contexts/srd/2022-08-28/07/2022-08-28.07-18-38.386-b65d1822.wav"}
```


Example (with results)


```
{"cookie": "759550161","error": {"code": 0},"name": "6c774a9f-c394-4137-8783-da181e8f7e7f","response": {"speakerSegments": [{"confidence": 0.99,"duration": 200,"id": "Anonymous-Speaker-2","location": 44},{"confidence": 0.99,"duration": 100,"id": "Anonymous-Speaker-3","location": 244},{"confidence": 0.99,"duration": 62,"id": "Anonymous-Speaker-1","location": 29546}]},"status": "ABORTED","type": "audiocodes.speech.SRDOperation","waveform-tag": "languages/xx-yy/contexts/srd/2022-08-28/07/2022-08-28.07-18-38.386-b65d1822.wav"}
```


Binary message payload to server

- Streaming audio in multiple chunks. Audio chunks should be in size equal are greater than 210msec.
- When working from files it is recommended to stream in larger chunks (e.g. 10sec) to reduce the transfer time to the server.
- Zero bytes payload indicates to the server the streaming ended (usually from EOF indication).

Text message for base64 based audio streaming to server

```
Streaming audio in multiple chunks utilizing base64 text audio streaming:
 {"action":"stream", "text":"kdXUUpEQkpmd3ENGEgZiBhZmE2MzQ1MjQyMXRycXdzZA==")
 Zero bytes equivalent payload in base64 based audio streaming indicates to the server the streaming ended (usually from EOF indication)
 {"action":"stream", "text":""})
```

5.3 Termination

Whenever the application wishes to stop the enrollment process, stop the process by either method:

- Issue “zero” bytes message by either:
  - Issue zero bytes binary payload.
  - Issue empty text json “text” property in base64 text-based audio streaming text message. This results in processing all the samples, indicating end of file. Use it when processing from files (this is the common use in recording offline diarization).
- Send "stop" request - this results in processing the speech samples received already in the server (without further samples that may be still transmitted)
- Send “abort” request – this results in the server stopping immediately without processing the speech samples any further.

Example request from client (stop request)

```
{"action":"stop"," cookie":"diarize me"}
```

5.4 Status Transitions

As can be observed from the examples above, the server reports process status within each text message. The status starts as **READY** right after the process is initialized, it continues with status **STARTED** until the process has ended with status **ABORTED**. In case of failure (some error occurred) the server changes the status to **FAILED**.

It is up to the client to stop the speaker diarization process, this is done as described above in Termination.

The server stops the process upon **stop** request, and this is done in stages. A brief description of these stages:

1. Server stops reading samples from binary messages.
2. Server processes all samples that have been already received and accumulated, at the same time any events are issued as during real-time.
3. Once all samples are processed, the process status changes to **TERMINATED** and when the process is halted completely the status becomes **ABORTED** provided with the diarization results.

5.5 Control message payload to server

In long period operations without active transmission (as could be the case in diarization of large audio files being transmitted and awaiting for long period the server to answer) of textual and\or binary messages in between client and server, it is recommended to send periodically (Every few seconds, see reference in unit test supplied) Ping message to the server, that is defined in the WebSocket protocol (https://tools.ietf.org/html/rfc6455#section-5.5 - see section 5.5.2 and 5.5.3). Otherwise, it has been observed in several cases the WebSocket connection is being disconnected and therefore unable to complete the operation.

6 Speaker Recognition Enrollment WebSocket API

Speaker segmentation enrollment WebSocket endpoint

ws[s]://<server IP>:<port>/api/v1/speech:SREnroll

6.1 Textual message payload to server

- Request to start enrollment with parameters governing the process in JSON format.
- Request to stop speaker segmentation in JSON format.
- Base64 based audio streaming.

To start the enrollment process, send action **start** along parameters related to the enrollment and with either of the following:

- Existing voiceprints (one or more) to use (to accumulate enrollment)
- Empty (one or more to create new ones from scratch).

Multiple speaker enrollment is supported only with external speech activity segments to hint the speaker identity of a segment.

Additionally, external speaker activity segments can be optionally set (e.g., from Microsoft Teams). Otherwise, an internal algorithm is being used to find speech activity. in addition, if provided as input, the enrollment output is accompanied with enhanced activity segments as well.

Optionally and in addition, an additional external speech activity segments indication (e.g., as result of speech to text process or other) could be sent in the API, so enrollment algorithm could utilize them.

The speaker speech language is independent and must be set to fixed to "xx-yy" specified through the parameter "accept-language".

Enrollment use cases:

No.	Existing Voiceprints	Speech activity segments	Speaker id per segment	Notes
1	+	+	+	Enrollment is performed according to external segments that belongs to the speaker id now being enrolled and existing voiceprints are enriched. (Teams use case)
2	+	+	-	Like use case 1, but the external segments are implicitly assigned to the speaker id now being enrolled and existing voiceprint is enriched. Applicable only for the use case of single speaker that resides in the audio and being enrolled.
3	+	-	-	Enrollment is performed on all utterance and existing voiceprint is enriched. Applicable only for the use case of single speaker that resides in the audio and being enrolled.
4	-	+	+	Like case No. 1, but with a new voiceprint being created to all speakers (one or more).
5	-\+	+	+	Like case No. 1, but some of the voiceprints are new voiceprints being created.
6	-	+	-	Like case No. 2, but with a new voiceprint being created. Applicable only for the use case of single speaker that resides in the audio and being enrolled.
7	-	-	-	Like case No. 3, but with a new voiceprint being created. Applicable only for the use case of single speaker that resides in the audio and being enrolled.

Example (start request with existing voiceprints (multiple) and with speech activity segments and with speaker id per segment)


```
{"action":"start","accept-language":"xx-yy","content-
type":"audio/l16;rate=16000","save-waveform":1,"save-
voiceprint":1,"cookie":"152617477","voiceprints":[{"id":"303F5D25-
8F2E-424D-990D-A0A0DB26E302","data":"
dHBsAGABAABTKGlpKUJCAAEAAAABAAAAgAAAAIU7CKxZmb/tDLaHnXgTIUesaULPOW
FDbC3v4edCCkCvAElU5P35a4BvRvT9RdQ7qiBQzAsUkODnuJI4hms4qTsQZMcCjMU8
a5LBdpoEEUe6lM04lcJR/hUaJ1hiTzgqn4tekX4atjqFzlzJx2dbsDpgjATIdUmxPQ
4ww5B1Q5WbwAAAACg6106t7OtKZtFQRwtl3WkLdVA5YmFXc+KLE7xs+M2dQ4F7mMkn
SvE9wsDF6h1t2l0sIFmyzT8RAJPo/09qwf36cOrwm56yShvMrixbfUOBTY73o9ehzz
CxS8BfoaEEA+oAIpPIiR3H0A68ajLC5dS0itGQCUBe2K9ZkfL6A+8ulj/7G6omexo0
+jAdUSURhya5wVmyUCwr4QLhN29HjFVORhZItEpBqv/g7Gu6A9/cio6W9yNXW8V3TF
uMvteO6w=="},{"id":"303F5D25-8F2E-424D-990D-A0A0DB26EAAA","data":"
dHBsAGABAABTKGlpKUJCAAEAAAABAAAAgAAAAIU7CKxZmb/tDLaHnXgTIUesaULPOW
FDbC3v4edCCkCvAElU5P35a4BvRvT9RdQ7qiBQzAsUkODnuJI4hms4qTsQZMcCjMU8
a5LBdpoEEUe6lM04lcJR/hUaJ1hiTzgqn4tekX4atjqFzlzJx2dbsDpgjATIdUmxPQ
4ww5B1Q5WbwAAAACg6106t7OtKZtFQRwtl3WkLdVA5YmFXc+KLE7xs+M2dQ4F7mMkn
SvE9wsDF6h1t2l0sIFmyzT8RAJPo/09qwf36cOrwm56yShvMrixbfUOBTY73o9ehzz
CxS8BfoaEEA+oAIpPIiR3H0A68ajLC5dS0itGQCUBe2K9ZkfL6A+8ulj/7G6omexo0
+jAdUSURhya5wVmyUCwr4QLhN29HjFVORhZItEpBqv/g7Gu6A9/cio6W9yNXW8V3TF
uMvteO6w=="}],"speechsegments":[{"location":100,"duration":50,"id"
:"303F5D25-8F2E-424D-990D-
A0A0DB26E302"},{"location":200,"duration":40,"id":"303F5D25-8F2E-
424D-990D-
A0A0DB26E302"},{"location":300,"duration":30,"id":"303F5D25-8F2E-
424D-990D-
A0A0DB26E302"},{"location":350,"duration":30,"id":"303F5D25-8F2E-
424D-990D-A0A0DB26EAAA"}]}
```


The example above meets the Microsoft Teams use case.

Example (start request with existing voiceprints (multiple) and with speech activity segments and with speaker id per segment and with external activity indication)


```
{"action":"start","accept-language":"xx-yy","content-
type":"audio/l16;rate=16000","save-waveform":1,"save-
voiceprint":1,"cookie":"152617477","voiceprints":[{"id":"303F5D25-
8F2E-424D-990D-A0A0DB26E302","data":"
dHBsAGABAABTKGlpKUJCAAEAAAABAAAAgAAAAIU7CKxZmb/tDLaHnXgTIUesaULPOW
FDbC3v4edCCkCvAElU5P35a4BvRvT9RdQ7qiBQzAsUkODnuJI4hms4qTsQZMcCjMU8
a5LBdpoEEUe6lM04lcJR/hUaJ1hiTzgqn4tekX4atjqFzlzJx2dbsDpgjATIdUmxPQ
4ww5B1Q5WbwAAAACg6106t7OtKZtFQRwtl3WkLdVA5YmFXc+KLE7xs+M2dQ4F7mMkn
SvE9wsDF6h1t2l0sIFmyzT8RAJPo/09qwf36cOrwm56yShvMrixbfUOBTY73o9ehzz
CxS8BfoaEEA+oAIpPIiR3H0A68ajLC5dS0itGQCUBe2K9ZkfL6A+8ulj/7G6omexo0
+jAdUSURhya5wVmyUCwr4QLhN29HjFVORhZItEpBqv/g7Gu6A9/cio6W9yNXW8V3TF
uMvteO6w=="},{"id":"303F5D25-8F2E-424D-990D-A0A0DB26EAAA","data":"
dHBsAGABAABTKGlpKUJCAAEAAAABAAAAgAAAAIU7CKxZmb/tDLaHnXgTIUesaULPOW
FDbC3v4edCCkCvAElU5P35a4BvRvT9RdQ7qiBQzAsUkODnuJI4hms4qTsQZMcCjMU8
a5LBdpoEEUe6lM04lcJR/hUaJ1hiTzgqn4tekX4atjqFzlzJx2dbsDpgjATIdUmxPQ
4ww5B1Q5WbwAAAACg6106t7OtKZtFQRwtl3WkLdVA5YmFXc+KLE7xs+M2dQ4F7mMkn
SvE9wsDF6h1t2l0sIFmyzT8RAJPo/09qwf36cOrwm56yShvMrixbfUOBTY73o9ehzz
CxS8BfoaEEA+oAIpPIiR3H0A68ajLC5dS0itGQCUBe2K9ZkfL6A+8ulj/7G6omexo0
+jAdUSURhya5wVmyUCwr4QLhN29HjFVORhZItEpBqv/g7Gu6A9/cio6W9yNXW8V3TF
uMvteO6w=="}],"speechsegments":[{"location":100,"duration":50,"id"
:"303F5D25-8F2E-424D-990D-
A0A0DB26E302"},{"location":200,"duration":40,"id":"303F5D25-8F2E-
424D-990D-
A0A0DB26E302"},{"location":300,"duration":30,"id":"303F5D25-8F2E-
424D-990D-
A0A0DB26E302"},{"location":350,"duration":30,"id":"303F5D25-8F2E-
424D-990D-A0A0DB26EAAA"}]}
```


Example (start request with empty and existing voiceprints (multiple) and with speech activity segments and with speaker id per segment)


```
{"action":"start","accept-language":"xx-yy","content-
type":"audio/l16;rate=16000","save-waveform":1,"save-
voiceprint":1,"cookie":"152617477","voiceprints":[{"id":"303F5D25-
8F2E-424D-990D-A0A0DB26E302","data":"
dHBsAGABAABTKGlpKUJCAAEAAAABAAAAgAAAAIU7CKxZmb/tDLaHnXgTIUesaULPOW
FDbC3v4edCCkCvAElU5P35a4BvRvT9RdQ7qiBQzAsUkODnuJI4hms4qTsQZMcCjMU8
a5LBdpoEEUe6lM04lcJR/hUaJ1hiTzgqn4tekX4atjqFzlzJx2dbsDpgjATIdUmxPQ
4ww5B1Q5WbwAAAACg6106t7OtKZtFQRwtl3WkLdVA5YmFXc+KLE7xs+M2dQ4F7mMkn
SvE9wsDF6h1t2l0sIFmyzT8RAJPo/09qwf36cOrwm56yShvMrixbfUOBTY73o9ehzz
CxS8BfoaEEA+oAIpPIiR3H0A68ajLC5dS0itGQCUBe2K9ZkfL6A+8ulj/7G6omexo0
+jAdUSURhya5wVmyUCwr4QLhN29HjFVORhZItEpBqv/g7Gu6A9/cio6W9yNXW8V3TF
uMvteO6w=="},{"id":"303F5D25-8F2E-424D-990D-
A0A0DB26EAAA","data":""}],"speechsegments":[{"location":100,"durat
ion":50,"id":"303F5D25-8F2E-424D-990D-
A0A0DB26E302"},{"location":200,"duration":40,"id":"303F5D25-8F2E-
424D-990D-
A0A0DB26E302"},{"location":300,"duration":30,"id":"303F5D25-8F2E-
424D-990D-
A0A0DB26E302"},{"location":350,"duration":30,"id":"303F5D25-8F2E-
424D-990D-
A0A0DB26EAAA"}],"activitysegments":[{"location":51,"duration":48,"
label":"
אבל "},{"location":99,"duration":27,"label":"היה "}] }
```


The example above meets the Microsoft Teams use case.

Example (start request with existing voiceprint and with speech activity segments and without speaker id per segment)


```
{"action":"start","accept-language":"xx-yy","content-
type":"audio/l16;rate=16000","save-waveform":1,"save-
voiceprint":1,"cookie":"152617477","voiceprints":[{"id":"303F5D25-
8F2E-424D-990D-A0A0DB26E302","data":"
dHBsAGABAABTKGlpKUJCAAEAAAABAAAAgAAAAIU7CKxZmb/tDLaHnXgTIUesaULPOW
FDbC3v4edCCkCvAElU5P35a4BvRvT9RdQ7qiBQzAsUkODnuJI4hms4qTsQZMcCjMU8
a5LBdpoEEUe6lM04lcJR/hUaJ1hiTzgqn4tekX4atjqFzlzJx2dbsDpgjATIdUmxPQ
4ww5B1Q5WbwAAAACg6106t7OtKZtFQRwtl3WkLdVA5YmFXc+KLE7xs+M2dQ4F7mMkn
SvE9wsDF6h1t2l0sIFmyzT8RAJPo/09qwf36cOrwm56yShvMrixbfUOBTY73o9ehzz
CxS8BfoaEEA+oAIpPIiR3H0A68ajLC5dS0itGQCUBe2K9ZkfL6A+8ulj/7G6omexo0
+jAdUSURhya5wVmyUCwr4QLhN29HjFVORhZItEpBqv/g7Gu6A9/cio6W9yNXW8V3TF
uMvteO6w=="}],"speechsegments":[{"location":100,"duration":50},{"l
ocation":200,"duration":40},{"location":300,"duration":30}]}
```


Example (start request with existing voiceprint and without speech activity segments)


```
{"action":"start","accept-language":"xx-yy","content-
type":"audio/l16;rate=16000","save-waveform":1,"save-
voiceprint":1,"cookie":"200087927","voiceprints":[{"id":"303F5D25-
8F2E-424D-990D-A0A0DB26E302","data":"
dHBsAGABAABTKGlpKUJCAAEAAAABAAAAgAAAAIU7CKxZmb/tDLaHnXgTIUesaULPOW
FDbC3v4edCCkCvAElU5P35a4BvRvT9RdQ7qiBQzAsUkODnuJI4hms4qTsQZMcCjMU8
a5LBdpoEEUe6lM04lcJR/hUaJ1hiTzgqn4tekX4atjqFzlzJx2dbsDpgjATIdUmxPQ
4ww5B1Q5WbwAAAACg6106t7OtKZtFQRwtl3WkLdVA5YmFXc+KLE7xs+M2dQ4F7mMkn
SvE9wsDF6h1t2l0sIFmyzT8RAJPo/09qwf36cOrwm56yShvMrixbfUOBTY73o9ehzz
CxS8BfoaEEA+oAIpPIiR3H0A68ajLC5dS0itGQCUBe2K9ZkfL6A+8ulj/7G6omexo0
+jAdUSURhya5wVmyUCwr4QLhN29HjFVORhZItEpBqv/g7Gu6A9/cio6W9yNXW8V3TF
uMvteO6w=="}]}
```


Example (start request with empty one and without speech activity segments)


```
{"action":"start","accept-language":"xx-yy","content-
type":"audio/l16;rate=16000","save-waveform":1,"save-
voiceprint":1,"cookie":"200087927","voiceprints":[{"id":"303F5D25-
8F2E-424D-990D-A0A0DB26E302","data":""}]}
```


Several media types are supported via "content-type":

- audio/l16;rate=8000
- **audio/l16;rate=16000 (recommended)**
- audio/pcma;rate=8000
- audio/pcmu;rate=8000
- audio/pcma;rate=16000
- audio/pcmu;rate=16000

To process stereo streams, the media type must indicate the number of channels, otherwise, it is considered as a mono stream.

e.g., audio/PCMU;rate=16000**;channels=2**

optionally media type may indicate explicitly that the stream is mono by denoting channels is one, e.g., audio/PCMU;rate=16000**;channels=1**

if stereo stream was indicated in API, a mix down stereo to mono operation is employed.

6.2 Packing the voiceprint bytes as payload to server

The binary voiceprint is 64-bit encoded as a string and set to the **data** field under **voiceprints** array in the JSON text message payload. The **id** is set by the client using arbitrary string to application specific requirements.

6.3 Enrollment parameters

There are currently no parameters governing the enrollment process.

The server allows a string value to be passed along with text messages, this is called ***cookie***. The ***cookie*** is present in all subsequent server text messages resulting from the process **start** message. You can use it to identify the particular process within multiple asynchronous processes.

The server, for **debug purposes only** (due to sensitivity of the information), allows setting:

save-waveform – save the session audio recording as received from the client.

save-voiceprint – save the session last voiceprint being enrolled.

6.4 Binary message payload to server

- Streaming audio in multiple chunks. Audio chunks should be in size equal are greater than 210msec.
- When working from files it is recommended to stream in larger chunks (e.g. 10sec) to reduce the transfer time to the server.
- Zero bytes payload indicates to the server the streaming ended (usually from EOF indication).

6.5 Text message for base64 based audio streaming to server

Streaming audio in multiple chunks utilizing base64 text audio streaming:

```
{"action":"stream",
"text":"kdXUUpEQkpmd3ENGEgZiBhZmE2MzQ1MjQyMXRycXdzZA==")
```

Zero bytes equivalent payload in base64 based audio streaming indicates to the server the streaming ended (usually from EOF indication)

```
{"action":"stream", "text":""})
```

6.6 Termination

Whenever the application wishes to stop the enrollment process, stop the process by either one of the methods below:

- Issue “zero” bytes message by either:
  - Issue zero bytes binary payload.
  - Issue empty text json “text” property in base64 text-based audio streaming text message. this results in processing all the samples, indicating end of file. Use it when processing from files.
- Send "stop" request - this results in processing the speech samples received already in the server (without further samples that may be still transmitted)
- Send “abort” request – this results in the server stopping immediately without processing the speech samples any further.

Example request from client (stop request)

```
{"action":"stop"," cookie":"enrolling me"}
```

6.7 Textual message payload from server

The server notifies of any status change, the final status is always **ABORTED**, this status indicates that the process has ended.

You must check the **error** token field **code**. Any value different than 0 indicates a problem in enrollment.

An error code 0 means no error and voiceprint is returned in the response. For logging purposes, the voiceprint id is given in the JSON field **name**.

The voiceprint cannot be used in recognition process before the JSON field **state** is **ENROLLING** or **ENROLLED**. The **score JSON** field suggests about the voiceprint matureness and if there is enough or insufficient speech for the speaker, to decide if it can be used in segmentation. See application guidelines documentation for **score** ranges ranking and recommendation.

Example enrollment (status change along a real flow)


```
{"name":"1d856bd0-04b3-4dc8-bec3-a85bcdad1d1d","type":"audiocodes.speech.SREOperation","status":"READY","cookie":"152617477"}
{"name":"1d856bd0-04b3-4dc8-bec3-a85bcdad1d1d","type":"audiocodes.speech.SREOperation","error":{},"status":"ABORTED","cookie":"152617477","response":{"voiceprints":[{"id":"303F5D25-8F2E-424D-990D-A0A0DB26E302","data":"Wu55ekCBNCnLU5+FJD480hBAIKJoIgScyVeae4wnHVr5DSwut13YlIcp/M0zZs0GwEux3ev6VSLUYvna9GQPilQ0BbIBA1+Xu9sIRe3K6tXiQrzsNcAbhIms3e97gKIY1S2rDQS78LrNL2z3O7G8xtzu6NUtvIFLUvzswQrF/Y3jv8hHqsyzhibu0OsGZWphFxfeoro1rP6WHM5YinlJceWxAWnagNGCLuDbKDQsHS77hUZbxnXHJbzSLLzh5a1KRcwAeXwdcjdNzDa2rk/9tvH/==","state":"ENROLLING","score":49}],"speechsegments":[{"location":100,"duration":50,"id":"303F5D25-8F2E-424D-990D-A0A0DB26E302"},{"location":200,"duration":40,"id":"303F5D25-8F2E-424D-990D-A0A0DB26E302"},{"location":300,"duration":30,"id":"303F5D25-8F2E-424D-990D-A0A0DB26E302"},,{"location":350,"duration":30,"id":"303F5D25-8F2E-424D-990D-A0A0DB26EAAA"}]
}}
```


6.8 Control message payload to server

In long period operations without active transmission (as could be the case in enrollment or segmentation of large audio files being transmitted and awaiting for long period the server to answer) of textual and\or binary messages in between client and server, it is recommended to send periodically (Every few seconds, see reference in unit test supplied) Ping message to the server, that is defined in the WebSocket protocol (https://tools.ietf.org/html/rfc6455#section-5.5 - see section 5.5.2 and 5.5.3). Otherwise, it has been observed in several cases the WebSocket connection is being disconnected and therefore unable to complete the operation.

7 Speaker Recognition Segmentation - WebSocket API

Speaker segmentation recognition WebSocket endpoint

ws[s]://<server IP>:<port>/api/v1/speech: SRSegment

7.1 Understanding Session and Process

In this document, the word ***Session*** refers to a WebSocket session, the term ***Process*** refers to AC Speech server task taking place in an asynchronous way. A process is run employing a session as a means of communication and that session can, once the process is over, be used for executing another process in the same way. Whenever the term status is mentioned, it refers to process status.

7.2 Textual message payload to server

- Request to start speaker segmentation with parameters governing the process in JSON format.
- Request to stop speaker segmentation in JSON format.
- Base64 based audio streaming.

To request a speaker segmentation task, send action **start** with voiceprints array specifying the speakers that may be found in audio (please follow application guidelines documentation for voiceprints matureness, the application is responsible to manage the use of voiceprints according to their score and ranking to ensure best performance), the speech language is independent but is set through the parameter "accept-language" to constant value xx-yy (reserved for future).

Additionally, external speech activity segments can be optionally set (e.g. from speech-to-text word segmentation or by other means). Otherwise, an internal algorithm is being used to detect speech activity.

The binary voiceprint is 64-bit encoded as a string and set to the **voiceprint** field in the JSON text message payload.

Example (start request and with speech activity segments and with speaker segmentation information)


```
{"action":"start","accept-language":"xx-yy","content-
type":"audio/l16;rate=16000","save-waveform":1,"save-
voiceprint":1,"cookie":"558615043","voiceprints":[{"id":"303F5D25-
8F2E-424D-990D-A0A0DB26E302","data":"
AAARESIiMzNERFVVZmZ3d4iImZmQAAEREiIjMzRERVVWZmd3eIiJmZkAABERIiIzM0
REVVVmZnd3iIiZmZAAARESIiMzNERFVVZmZ3d4iImZmQAAEREiIjMzRERVVWZmd3eI
iJmZkA=="},{"id":"780F91D6-4119-4C3F-9D24-ABD24BFD09E1","data":"
AAARESIiMzNERFVVZmZ3d4iImZmQAAEREiIjMzRERVVWZmd3eIiJmZkAABERIiIzM0
REVVVmZnd3iIiZmZAAARESIiMzNERFVVZmZ3d4iImZmQAAEREiIjMzRERVVWZmd3eI
iJmZkA=="},{"id":"3E5421AC-48D4-419E-BC93-A677403F5813","data":"
ZwiXE4cxCDlwlmJ3FYlwKXdYNyEJVwgpeJcgaJJwiWcpAnq814OYk3YIOXY5djdjhj
l2OXZnhieGJ4lafhjklahjklhaklj25j4klkjaflljfahfja2hClnSJdolziXCXhpg
3iWOZc=="},{"id":"FD8B36AC-9DFE-4C7F-910D-A1A179A373D4","data":"
ZwiXE4cxCDlwlmJ3FYlwKXdYNyEJVwgpeJcgaJJwiWcpAnq814OYk3YIOXY5djdjhj
l2OXZnhieGJ4lafhjklahjklhaklj25j4klkjaflljfahfja2hClnSJdolziXCXhpg
3iWOZc=="}],"speechsegments":[{"location":100,"duration":50,"id":"
780F91D6-4119-4C3F-9D24-
ABD24BFD09E1"},{"location":200,"duration":40,"id":"FD8B36AC-9DFE-
4C7F-910D-
A1A179A373D4"},{"location":300,"duration":30,"id":"FD8B36AC-9DFE-
4C7F-910D-A1A179A373D4"}]}
```


The example above meets the Microsoft Teams use case.

Example (start request and with speech activity segments and without speaker segmentation information)


```
{"action":"start","accept-language":"xx-yy","content-
type":"audio/l16;rate=16000","save-waveform":1,"save-
voiceprint":1,"cookie":"558615043","voiceprints":[{"id":"303F5D25-
8F2E-424D-990D-A0A0DB26E302","data":"
AAARESIiMzNERFVVZmZ3d4iImZmQAAEREiIjMzRERVVWZmd3eIiJmZkAABERIiIzM0
REVVVmZnd3iIiZmZAAARESIiMzNERFVVZmZ3d4iImZmQAAEREiIjMzRERVVWZmd3eI
iJmZkA=="},{"id":"780F91D6-4119-4C3F-9D24-ABD24BFD09E1","data":"
AAARESIiMzNERFVVZmZ3d4iImZmQAAEREiIjMzRERVVWZmd3eIiJmZkAABERIiIzM0
REVVVmZnd3iIiZmZAAARESIiMzNERFVVZmZ3d4iImZmQAAEREiIjMzRERVVWZmd3eI
iJmZkA=="},{"id":"3E5421AC-48D4-419E-BC93-A677403F5813","data":"
ZwiXE4cxCDlwlmJ3FYlwKXdYNyEJVwgpeJcgaJJwiWcpAnq814OYk3YIOXY5djdjhj
l2OXZnhieGJ4lafhjklahjklhaklj25j4klkjaflljfahfja2hClnSJdolziXCXhpg
3iWOZc=="},{"id":"FD8B36AC-9DFE-4C7F-910D-A1A179A373D4","data":"
ZwiXE4cxCDlwlmJ3FYlwKXdYNyEJVwgpeJcgaJJwiWcpAnq814OYk3YIOXY5djdjhj
l2OXZnhieGJ4lafhjklahjklhaklj25j4klkjaflljfahfja2hClnSJdolziXCXhpg
3iWOZc=="}],"speechsegments":[{"location":100,"duration":50},{"loc
ation":200,"duration":40},{"location":300,"duration":30}]}
```

Example (start request and without speech activity segments)


```
{"action":"start","accept-language":"xx-yy","content-
type":"audio/l16;rate=16000","save-waveform":1,"save-
voiceprint":1,"cookie":"558615043","voiceprints":[{"id":"303F5D25-
8F2E-424D-990D-A0A0DB26E302","data":"
AAARESIiMzNERFVVZmZ3d4iImZmQAAEREiIjMzRERVVWZmd3eIiJmZkAABERIiIzM0
REVVVmZnd3iIiZmZAAARESIiMzNERFVVZmZ3d4iImZmQAAEREiIjMzRERVVWZmd3eI
iJmZkA=="},{"id":"780F91D6-4119-4C3F-9D24-ABD24BFD09E1","data":"
AAARESIiMzNERFVVZmZ3d4iImZmQAAEREiIjMzRERVVWZmd3eIiJmZkAABERIiIzM0
REVVVmZnd3iIiZmZAAARESIiMzNERFVVZmZ3d4iImZmQAAEREiIjMzRERVVWZmd3eI
iJmZkA=="},{"id":"3E5421AC-48D4-419E-BC93-A677403F5813","data":"
ZwiXE4cxCDlwlmJ3FYlwKXdYNyEJVwgpeJcgaJJwiWcpAnq814OYk3YIOXY5djdjhj
l2OXZnhieGJ4lafhjklahjklhaklj25j4klkjaflljfahfja2hClnSJdolziXCXhpg
3iWOZc=="},{"id":"FD8B36AC-9DFE-4C7F-910D-A1A179A373D4","data":"
ZwiXE4cxCDlwlmJ3FYlwKXdYNyEJVwgpeJcgaJJwiWcpAnq814OYk3YIOXY5djdjhj
l2OXZnhieGJ4lafhjklahjklhaklj25j4klkjaflljfahfja2hClnSJdolziXCXhpg
3iWOZc=="}]}
```


Several media types are supported via "content-type":

- audio/l16;rate=8000
- **audio/l16;rate=16000 (recommended)**
- audio/pcma;rate=8000
- audio/pcmu;rate=8000
- audio/pcma;rate=16000
- audio/pcmu;rate=16000

To process stereo streams, the media type must indicate the number of channels, otherwise, it is considered as a mono stream.

e.g., audio/PCMU;rate=16000**;channels=2**

optionally media type may indicate explicitly that the stream is mono by denoting channels is one, e.g., audio/PCMU;rate=16000**;channels=1**

if stereo stream was indicated in API, a mix down stereo to mono operation is employed.

7.3 Textual message payload from server

From time to time the server notifies the status and results to the client in JSON format. The client should monitor the status and manage the process accordingly.

Example Speech Recognition task (status change along a real flow)


```
{"name":"7a43ed65-91a7-4d51-a0ee-4bcf39954561","type":"audiocodes.speech.SRSOperation","error":{},"status":"STARTED","cookie":"558615043"}
{"name":"7a43ed65-91a7-4d51-a0ee-4bcf39954561","type":"audiocodes.speech.SRSOperation","status":"READY","cookie":"558615043"}
{"name":"7a43ed65-91a7-4d51-a0ee-4bcf39954561","type":"audiocodes.speech.SRSOperation","error":{},"status":"TERMINATED","cookie":"558615043"}
```


Example (with results)


```
{"name":"a9c9ef34-7737-44fd-98e8-ba4a36856537","type":"audiocodes.speech.SRSOperation","error":{},"status":"ABORTED","cookie":"430088109","response":{"speakersegments":[{"location":42,"duration":24,"id":"303F5D25-8F2E-424D-990D-A0A0DB26E302","confidence":0.8123},{"location":78,"duration":32,"id":"780F91D6-4119-4C3F-9D24-ABD24BFD09E1","confidence":0.5124}]}}
```


7.4 Binary message payload to server

- Streaming audio in multiple chunks. Audio chunks should be in size equal are greater than 210msec.
- When working from files it is recommended to stream in larger chunks (e.g. 10sec) to reduce the transfer time to the server.
- Zero bytes payload indicates to the server the streaming ended (usually from EOF indication).

7.5 Text message for base64 based audio streaming to server

```
Streaming audio in multiple chunks utilizing base64 text audio streaming:
 {"action":"stream",
"text":"kdXUUpEQkpmd3ENGEgZiBhZmE2MzQ1MjQyMXRycXdzZA==")
 Zero bytes equivalent payload in base64 based audio streaming indicates to the server the streaming ended (usually from EOF indication)
 {"action":"stream", "text":""})
```

7.6 Termination

Whenever the application wishes to stop the enrollment process, stop the process by either method:

- Issue “zero” bytes message by either:
  - Issue zero bytes binary payload.
  - Issue empty text json “text” property in base64 text-based audio streaming text message.
- this results in processing all the samples, indicating end of file. Use it when processing from files (this is the common use in meetings offline segmentation).
- Send "stop" request - this results in processing the speech samples received already in the server (without further samples that may be still transmitted)
- Send “abort” request – this results in the server stopping immediately without processing the speech samples any further.

Example request from client (stop request)

{"action":"stop"," cookie":"segment me"}

7.7 Status Transitions

As can be observed from the examples above, the server reports process status within each text message. The status starts as **READY** right after the process is initialized, it continues with status **STARTED** until the process has ended with status **ABORTED**. In case of failure (some error occurred) the server changes the status to **FAILED**.

It is up to the client to stop the speaker segmentation process; this is done as described above in Termination.

The server stops the process upon **stop** request, and this is done in stages. A brief description of these stages:

1. Server stops reading samples from binary messages.
2. Server processes all samples that have been already received and accumulated, at the same time any events are issued as during real-time.
3. Once all samples are processed, the process status changes to **TERMINATED** and when the process is halted completely the status becomes **ABORTED** provided with the segmentation results.

7.8 Control message payload to server

In long period operations without active transmission (as could be the case in enrollment or segmentation of large audio files being transmitted and awaiting for long period the server to answer) of textual and\or binary messages in between client and server, it is recommended to send periodically (Every few seconds, see reference in unit test supplied) Ping message to the server, that is defined in the WebSocket protocol (https://tools.ietf.org/html/rfc6455#section-5.5 - see section 5.5.2 and 5.5.3). Otherwise, it has been observed in several cases the WebSocket connection is being disconnected and therefore unable to complete the operation.

8 AudioCodes Speech REST API

8.1 Offline Diarization API

The `<Speech_Server_IP>/v1/speech:diarize` URL when used with the POST method, provides the ability for the transcription client to send a request to the server to transcribe and diarize an audio file.

REST Resource

`<Speech_Server_IP>/v1/speech:diarize`

HTTP Method

`POST`

Content-Type

`application/json`

Path Variables

Attribute	Type	Description

Request Message Body

Fields	Description
String : audio-file String: content-type String: cookie Integer: diarization-gap Integer: save-waveform Array: word-segments (see below - optional)	**audio-file** base64 audio file **content-type** mime type of the **audio-file** one of: ■ audio/l16;rate=8000;channels=1 ■ audio/l16;rate=16000;channels=1 ■ audio/PCMA;rate=8000;channels=1 ■ audio/PCMA;rate=16000;channels=1 ■ audio/PCMU;rate=8000;channels=1 ■ audio/PCMU;rate=16000;channels=1 ■ audio/l16;rate=8000;channels=2 ■ audio/l16;rate=16000;channels=2 ■ audio/PCMA;rate=8000;channels=2 ■ audio/PCMA;rate=16000;channels=2 ■ audio/PCMU;rate=8000;channels=2 ■ audio/PCMU;rate=16000;channels=2 **cookie** optional application session labeling **diarization-gap** an optional parameter that governs merge segments of same speaker, where gap is less than the parameter given in [msec]. the default value is 20000 [msec]. **save-waveform** an optional parameter that enables (value 1) or disable (value 0) waveform save at server side (usually used for debugging purposes) **word-segments** an optional array of objects with word segmentation in sequence (e.g. output of speech to text process) introduced as input to the diarization algorithm
word-segments	Array: *Object*
Object	**word** text of the word **location** word begins in frames (frame equals 10msec) **duration** word duration in frames (frame equals 10msec) **confidence** word confidence between 0 to 1

Reply Content-Type

`application/json; charset=utf-8`

Reply Message Body

Entity	Fields	Description
transcription	Array : *objectarray1 (see below)* String: cookie String: waveform-tag	**cookie** application session labeling **waveform-tag** uri path to server waveform recording. the entity appears only if save-waveform was enabled.
objectarray1	String: id Int: location Int: duration String: text Array: words *(see below)*	**id** diarization speaker label **location** frame index where diarized speaker segment starts in audio frames (x10msec) **duration** period of diarized speaker lasts in audio frames (x10msec) **text** speech to text words sequence recognized under the diarized segment **words** speech to text words detailed information including location, duration, confidence, and text
Words	Array : *objectarray2 (see below)*	
objectarray2	String: word Int: location Int: duration Float: confidence	**word** the text representing the word under the segment **location** frame index where word starts in audio frames (x10msec) **duration** period of word lasts in audio frames (x10msec) **confidence** word level confidence score in the range 0.0 to 1.0

HTTP Response

- `200 OK`

8.2 Offline Diarize API response example (with word-segments as input)


```
{
    "transcription": [
            {
                    "id": "Anonymous-Speaker-1",
                    "location": 250,
                    "duration": 170,
                    "text": "word1 word2 word3",
                    "words": [
                            {
                                    "word": "word1",
                                    "location": 250,
                                    "duration": 40,
                                    "confidence": 0.5854
                            },
                            {
                                    "word": "word2",
                                    "location": 320,
                                    "duration": 50,
                                    "confidence": 0.6854
                            },
                            {
                                    "word": "word3",
                                    "location": 370,
                                    "duration": 30,
                                    "confidence": 0.7854
                            }
                    ]
            },
            {
                    "id": "Anonymous-Speaker-2",
                    "location": 800,
                    "duration": 250,
                    "text": "word3 word4 word5",
                    "words": [
                            {
                                    "word": "word3",
                                    "location": 800,
                                    "duration": 80,
                                    "confidence": 0.2854
                            },
                            {
                                    "word": "word4",
                                    "location": 880,
                                    "duration": 60,
                                    "confidence": 0.4454
                            },
                            {
                                    "word": "word5",
                                    "location": 1000,
                                    "duration": 30,
                                    "confidence": 0.9854
                            }
                    ]
            }
    ]
"cookie" : "application defined cookie"
}
```

International Headquarters
1 Hayarden Street,
Airport City
Lod 7019900, Israel
Tel: +972-3-976-4000
Fax: +972-3-976-4040

AudioCodes Inc.
80 Kingsbridge Rd
Piscataway, NJ 08854, USA
Tel: +1-732-469-0880
Fax: +1-732-469-2298

Contact us: https://www.audiocodes.com/corporate/offices-worldwide
Website: https://www.audiocodes.com

Document #: LTRT-26013