API Reference Guide

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

AudioCodes Automatic Speech Recognition - WebSocket API

Version 0.49

audiocodes

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: June-20-2023

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-26008 AudioCodes Speech – LVCSR WebSocket API (v0.59)
LTRT-26009 AudioCodes Speech REST API (v0.5)
LTRT-26004 AudioCodes Speech –Speaker Recognition Enrollment and Segment WebSocket API (v0.11)

Document Revision Record

LTRT	Description
26001	Initial document release for Version 0.49.
26007	Added two API parameters.

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 utilize AudioCodes' Automatic Speech Recognition (ASR) technology, specifically designed for Voice BOTSs with limited input duration. The technology operates via WebSocket-based protocol API, which is explained below. The document provides details of ASR session interaction including the parameters that govern the recognition session.

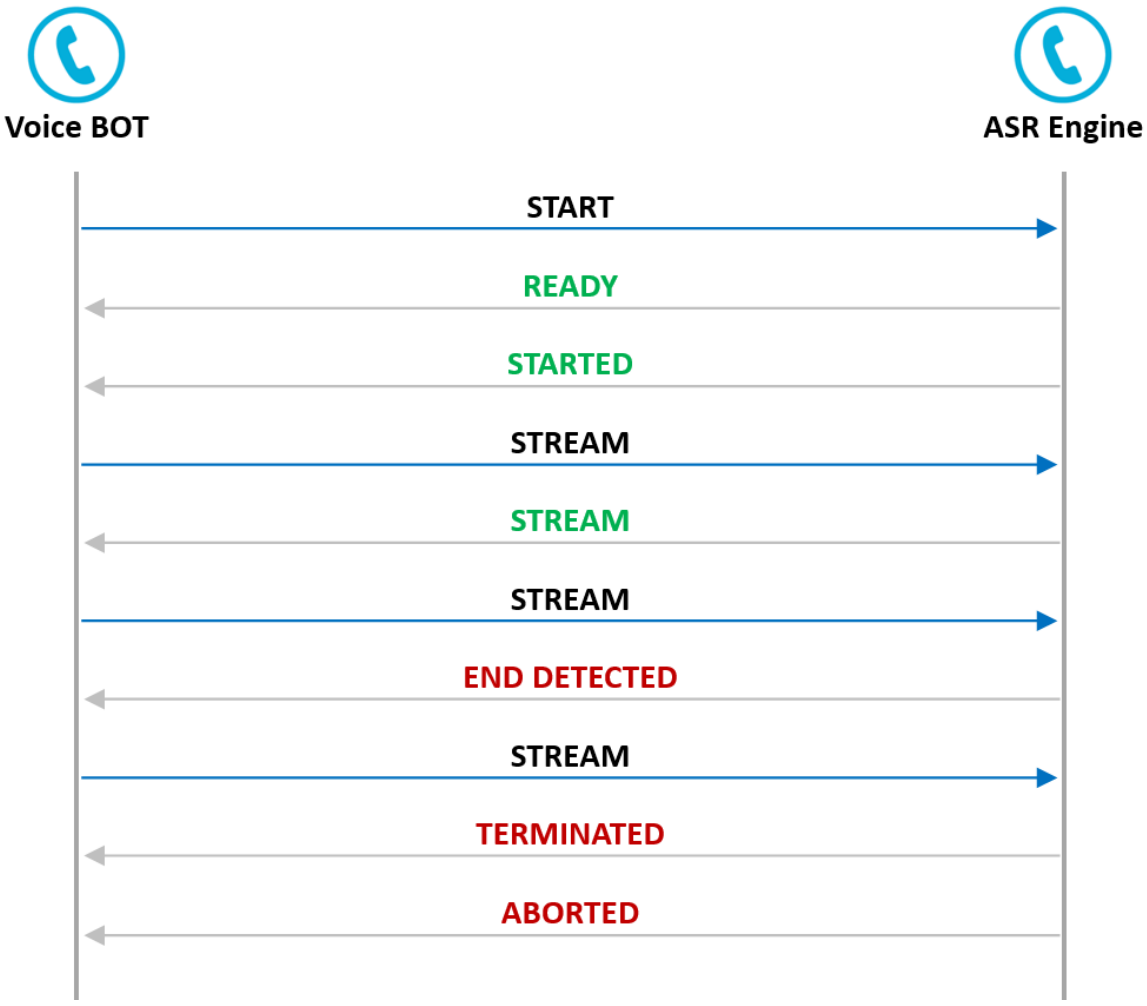
Example: Speech recognition web socket endpoint

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

Understanding Session and Process

For the purpose of this document, the word *"Session"* refers to a web-socket session, while the term *"Process"* refers to an AudioCodes Speech server task executed asynchronous. A process utilizes a session for communications, and once the process is over, the same session can execute another process in exactly the same way. Whenever the term *"status"* is mentioned, it refers to process status.

Figure 1: Example of messages flow between VOICE BOT and ASR Engine



2 API Reference

2.1 VOICE BOT -> ASR Engine START textual message

1. Request to start ASR with parameters governing the process in JSON format.
2. Request to stop ASR in JSON format.
3. Request ASR to start input timers in JSON format.
4. Base64 based audio streaming.

To request an LVCSR process, send action **start** (or use **start-mc** for stereo processing over default mix down stereo to mono in the case of stereo stream) with ***context*** specifying the context to use (context needs to be present on the server and must be coordinated to the specified domain). The speech language is specified through the ***accept-language*** parameter. The type of waveform samples supplied to the server must be provided in the parameter ***content-type***.

Adhoc glossary is an optional feature of ASR/LVCSR recognition customization. it allows you to add preset words and/or phrases that should be recognized in your audio data. The adhoc glossary is passed in the session ASR /LVCSR API together with phonetic lexicon (optional), adhoc-glossary-strictness, and adhoc-glossary-sensitivity parameters.

adhoc-glossary-strictness an optional parameter that determines how flexible the glossary could be spoken (e.g., saying "Open Word" instead of "Open Microsoft Word for Windows" and prefix & suffix flexibility). Value range: 0 to 100. Where 100 restricts recognition to the exact full phrases and exact prefix or suffix used). The value selection policy is highly dependent on the ability to characterize the recognizer actual inputs in the service. Changing the parameter value employs glossary compilation.

adhoc-glossary-sensitivity an optional parameter that determines the sensitivity in holding to the glossary defined phrases/words rather than generic speech. Value range: 0 to 100. Where 100 sensitivity is more attuned to the glossaries phrases (low rate of detection but higher accuracy), and 0 leads to less recognition to glossaries phrases and allows for more generic speech (high rate of detection but lower accuracy). Changing the parameter value does not employ glossary compilation.

In addition, a precompiled glossary functionality can be deployed using adhoc-glossary-id achieved by other sessions. The adhoc-glossary-id is bound and validated against a specific language and context. In case where adhoc-glossary and/or adhoc-glossary-lexicon are provided in API, they are favored over the adhoc-glossary-id.

To control tradeoff of detection rate vs accuracy, adhoc-glossary-sensitivity (value 0 to 100) is provided to determine the recognition. Changing the parameter value does not employ glossary compilation.

Example: start request

```
{"action":"start","context":"city_names","accept-language":"he-
il","content-type":"audio/l16;rate=8000","confidence-threshold":0,"n-
best-list-length":1,"recognition-timeout":5000,"speech-complete-
timeout":500,"speech-incomplete-timeout":1500, "no-input-timeout" : 5000,
"speed-vs-accuracy":50, "save-waveform":1, "start-input-timers":1,
"sensitivity-level": 90, "cookie":"say city name"}
```

Example: start request with adhoc glossary

```
{"action":"start","context":"city_names","accept-language":"he-
il","content-type":"audio/l16;rate=8000","confidence-threshold":0,"n-
best-list-length":1,"recognition-timeout":5000,"speech-complete-
timeout":500,"speech-incomplete-timeout":1500, "no-input-timeout" : 5000,
"speed-vs-accuracy":50, "save-waveform":1, "start-input-timers":1,
"sensitivity-level": 90, "cookie":"say city name"","adhoc-glossary":
["voip sbc", "meeting insights", "vocanom"]}
```

Example: start request with adhoc glossary and strictness

```
{"action":"start","context":"city_names","accept-language":"he-
il","content-type":"audio/l16;rate=8000","confidence-threshold":0,"n-
best-list-length":1,"recognition-timeout":5000,"speech-complete-
timeout":500,"speech-incomplete-timeout":1500, "no-input-timeout" : 5000,
"speed-vs-accuracy":50, "save-waveform":1, "start-input-timers":1,
"sensitivity-level": 90, "cookie":"say city name"","adhoc-glossary":
["voip sbc", "meeting insights", "vocanom"],"adhoc-glossary-strictness":
100}
```

Example: start request with adhoc glossary and adhoc glossary lexicon

```
{"action":"start","context":"city_names","accept-language":"he-
il","content-type":"audio/l16;rate=8000","confidence-threshold":0,"n-
best-list-length":1,"recognition-timeout":5000,"speech-complete-
timeout":500,"speech-incomplete-timeout":1500, "no-input-timeout" : 5000,
"speed-vs-accuracy":50, "save-waveform":1, "start-input-timers":1,
"sensitivity-level": 90, "cookie":"say city name"","adhoc-glossary":
["voip sbc", "meeting insights", "vocanom"],"adhoc-glossary-lexicon" :
{"voip": [" v o j s o v e R a j p i","v o j p"],"sbc" : ["e s b i s
i"],"meeting" : ["m i t i n g"],"insights" : ["i n s a j ts"],"vocanom" :
["v o k a n o m"]}}
```

Example: start request with adhoc glossary id (pre-compiled glossary usage)

```
{"action":"start","context":"city_names","accept-language":"he-
il","content-type":"audio/l16;rate=8000","confidence-threshold":0,"n-
best-list-length":1,"recognition-timeout":5000,"speech-complete-
timeout":500,"speech-incomplete-timeout":1500, "no-input-timeout" : 5000,
"speed-vs-accuracy":50, "save-waveform":1, "start-input-timers":1,
"sensitivity-level": 90, "cookie":"say city name","adhoc-glossary-id": "
sttg-387c2e04d5cf4741bc6bdd7bba30aea3-
748d0485a1cc01e8d756129c6300d186147678064db798aa2a8ec0275d8abcec"}
```

Example: start request with adhoc glossary id (pre-compiled glossary usage) and sensitivity

```
{"action":"start","context":"city_names","accept-language":"he-
il","content-type":"audio/l16;rate=8000","confidence-threshold":0,"n-
best-list-length":1,"recognition-timeout":5000,"speech-complete-
timeout":500,"speech-incomplete-timeout":1500, "no-input-timeout" : 5000,
"speed-vs-accuracy":50, "save-waveform":1, "start-input-timers":1,
"sensitivity-level": 90, "cookie":"say city name","adhoc-glossary-id": "
sttg-387c2e04d5cf4741bc6bdd7bba30aea3-
748d0485a1cc01e8d756129c6300d186147678064db798aa2a8ec0275d8abcec","adhoc-
glossary-sensitivity": 50}
```

Example: start input timers request

This request is effective only if preceding action **start** request was sent with "start-input-timers":0:

```
{"action" : "start-input-timers"}
```

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

- audio/l16;rate=8000
- audio/l16;rate=16000
- audio/PCMA;rate=8000
- audio/PCMA;rate=16000
- audio/PCMU;rate=8000
- audio/PCMU;rate=16000

To process stereo streams, the media type must indicate the number of channels, otherwise it will be 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=1, e.g., audio/PCMU;rate=16000**;channels=1**

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

2.2 ASR Engine -> VOICE BOT Textual messages payload

From time to time the server notifies the status and ASR 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)

```
{"cookie":"say city name”,"name":"03fa6589-3475-4bd2-b73b-
acee0494bd84","type":"audiocodes.speech.ASROperation","status":"READY"}
{"cookie":"say city name","name":"03fa6589-3475-4bd2-b73b-
acee0494bd84","type":"audiocodes.speech.ASROperation","status":"STARTED"}
{"cookie":" say what you want ","name":"03fa6589-3475-4bd2-b73b-
acee0494bd84","type":"audiocodes.speech.ASROperation","status":"END_DETEC
TED"}
{"cookie":" say what you want ","name":"03fa6589-3475-4bd2-b73b-
acee0494bd84","type":"audiocodes.speech.ASROperation","status":"TERMINATE
D"}
{"cookie":" say what you want ","name":"03fa6589-3475-4bd2-b73b-
acee0494bd84","type":"audiocodes.speech.ASROperation","status":"ABORTED"}
```

Example: start of input event

```
{"name":"03fa6589-3475-4bd2-b73b-
acee0494bd84","type":"audiocodes.speech.ASROperation","error":{},"status"
:"STARTED","cookie":" say city name","event":"start-of-input"})
```

This event can be used to stop a prompt.

2.2.1 END_DETECTED status

Upon receiving the END_DETECTED status, regardless of the completion cause (i.e., success, no-match, no-input-timeout, recognizer-error, speech-too-early, success-maxtime, partial-match, partial-match-maxtime, or no-match-maxtime), the application must issue a "stop" command.

The END_DETECTED arises from:

- Timer expiry (e.g., incomplete-timeout, no-input, etc.).
- Licensing policy is not met, and the recognizer issues an event with status “END_DETECTED” and cause set to “recognizer-error”

In certain debug or offline scenarios where only files are being streamed, and the END_DETECTED status is not generated (as expiration conditions have not occurred), to successfully complete the session, the application must send a “stop” command. Otherwise, the engine will be waiting for more audio forever.

Sending “stop” will cause the process to end, and message with “ABORTED” state with results.

Example: results from a real flow


```
{"cause":"success","cookie":"say city name","name":"03fa6589-3475-4bd2-
b73b-
acee0494bd84","type":"audiocodes.speech.ASROperation","response":{"altern
atives":[{"transcript":"כן","interpretation":"כן","confidence":0.884182,"w
ords":[{"word":"כן","location":42,"duration":24,"confidence":0.884182}],"t
okens":[{"slot":"","value":"כן"}]}]},"status":"ABORTED"}
```


Example: results from a real flow with adhoc glossary id tag in results


```
{"cause":"success","cookie":"say city name","name":"03fa6589-3475-4bd2-
b73b-
acee0494bd84","type":"audiocodes.speech.ASROperation","response":{"altern
atives":[{"transcript":"כן","interpretation":"כן","confidence":0.884182,"w
ords":[{"word":"כן","location":42,"duration":24,"confidence":0.884182}],"t
okens":[{"slot":"","value":"כן"}]}]},"status":"ABORTED","adhoc-glossary-
id-tag": "languages/he-il/glossaries/sttg-
387c2e04d5cf4741bc6bdd7bba30aea3-
748d0485a1cc01e8d756129c6300d186147678064db798aa2a8ec0275d8abcec"}
```


Along the response with recognition results, the JSON results includes completion cause (only applicable to recognition process) according to MRCP v2 specifications (https://tools.ietf.org/html/rfc6787):

- success
- no-match
- no-input-timeout
- recognizer-error
- speech-too-early
- success-maxtime
- partial-match
- partial-match-maxtime
- no-match-maxtime

2.3 VOICE BOT -> ASR Engine Binary message

Streaming audio in multiple chunks. Zero bytes payload indicates to the server to stop the ASR process.

2.4 VOICE BOT -> ASR Engine STREAM textual message

Example: Streaming audio in multiple chunks utilizing base64 text audio streaming

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

Example: Zero bytes equivalent payload in base64 based audio streaming indicates to the server to stop the recognition process

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

2.5 Usage of adhoc glossary id

The JSON string entity specifies the glossary ID to be used in the session. The adhoc glossary ID, obtained from the response as 'adhoc-glossary-id-tag', is utilized when an adhoc glossary is employed, with or without an adhoc glossary lexicon. The purpose of this approach is to enable the use of pre-compiled glossary functionalities without explicitly providing the adhoc-glossary and lexicon, if they are provided.

Format of adhoc glossary

A JSON entity providing a string array of where its elements designate the word or phrases of the glossaries.

Example:

```
"adhoc-glossary": ["voip sbc", "meeting insights", "vocanom"]
```

2.6 Format of lexicon for providing transcriptions to adhoc glossary

The JSON entity contains a mapping of "attribute" to "value" pairs. The "**attribute**" represents the word's spelling, while the "**value**" is an array consisting of one or more transcriptions for that word. Linguistic support is necessary for word transcriptions. Although the adhoc glossary lexicon is not obligatory, it is recommended as it enhances accuracy.

Example:

```
"adhoc-glossary-lexicon":{"מיה":["m i a","m i j a"],"תודה":["t o d a"]}
```

2.7 Session Termination

Once status of the operation becomes ABORTED or FAILED stop the task by either method:

- Issue zero bytes binary payload (see sample code).
- Issue empty text JSON “text” property in base64 text-based audio streaming text message
- Send "stop" request (see sample code).

Afterword’s the WebSocket should be closed.

2.8 ASR Engine parameters

Our ASR parameters follow the MRCP standard, see https://tools.ietf.org/html/rfc6787 paragraph 9.4

Listed below are a few prominent recognition parameters:

confidence-threshold

This parameter threshold is employed in deciding which results to transfer to the client, a sentence with confidence value lower than confidence-threshold is removed from the results. The range of this parameter is 0-1

n-best-list-length

This parameter indicates the number of results to return.

no-input-timeout

This parameter indicates the number of milliseconds of no speech to elapse before the engine returns with no results.

recognition-timeout

This parameter indicates the number of milliseconds of speech to elapse before the engine returns with 'speech too long' results.

speech-complete-timeout

This parameter indicates the number of milliseconds of silence after end of speech so the engine returns with results.

In free speech grammar, *speech-complete-timeout* as complete timeout is irrelevant. The value doesn't affect the recognizer functionality.

speech-incomplete-timeout

This parameter indicates the number of milliseconds of silence after end of speech so the engine returns without results.

This refers to situations where the speech segment does not contain a complete and valid phrase in closed grammars.

save-waveform

The parameter flag is used to indicate whether the received waveform should be saved in its original form as provided by the server. The server includes a waveform-tag in the reported messages indicating the recording location.

start-input-timers

The parameter flag indicating to start no-input-timer from the begin of an audio stream. If not set, the application may use "start-input-timers" message to start (i.e., for barge-in use case after prompt end either finished or stopped as a result of start-of-input event).

sensitivity-level

This parameter controls the detection level of speech start.

Speed-vs-accuracy

This parameter controls the accuracy level of the recognizer in expense of delay (as it introduces higher CPU consumption). the range is 0 to 100 (50 as default).

2.9 Session 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 **TERMINATED** (or **ABORTED**). In case of failure (some error occurred) the server changes the status to **FAILED**.

It is the responsibility of the client to stop the LVCSR process, this is described above in Session Termination.

The server stops the process upon receiving a **stop** request and is completed in the following 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 it is during real-time.
3. Once all samples are processed, the process status changes to **TERMINATED** and the server emits the final events (if there are any). This is why after the status is reported to be **TERMINATED** you can still see events generated; these events will report words that have the **final** flag 1.
4. When the process is halted completely the status becomes **ABORTED**

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-26007