API Reference Guide

AudioCodes Secured AI Meeting Management for Highly Regulated Industries

Meeting Insights On-Prem Integration Guide

SW Version: 2.4.3

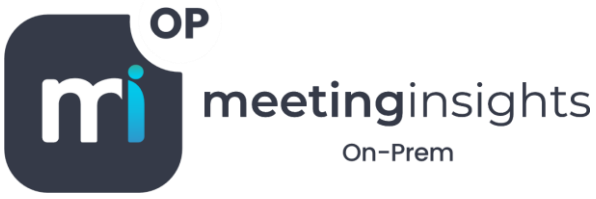


Table of Contents

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-08-2026

Security Vulnerabilities

All security vulnerabilities should be reported to vulnerability@audiocodes.com.

WEEE EU Directive

Pursuant to the WEEE EU Directive, electronic and electrical waste must not be disposed of with unsorted waste. Please contact your local recycling authority for disposal of this product.

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
Meeting Insights On-Prem Installation Manual
Meeting Insights On-Prem Release Notes
Meeting Insights On-Prem Solution Description and Use Cases

Document Revision Record

LTRT	Description
26020	Initial document release (Ver. 1.0).
26021	Updated with: ■ Task Submission API: • "taskId" is optional and can be received in response body to /hrec request • Polling done on task metadata • sttRemainingTime value for optimize polling ■ Workflow overview ■ Code examples ■ New section 'Get Task Metadata' ■ New section 'Get Tasks List'
26022	■ Added **genericAdditionalInfo** to task metadata. ■ Updated multi-language support: use **baseLanguage** instead of **sttLanguage** properties. ■ Updated script examples.

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 the end-to-end process for creating **offline transcription tasks** using AudioCodes **Meeting Insights On-Prem** backend services. The workflow involves authenticating via Keycloak, submitting audio files as part of a transcription task, and retrieving both metadata and transcription results once processing is complete.

The process is parameterized through a combination of static (environment-level) and dynamic (task-level) inputs, and is designed to support both automated and manual execution.

1.1 Workflow Parameters

- **Static (ideally derived from configuration):**

Key	Example value
KEYCLOAK_BASE_URL	https://example.audiocodes.com:8443
API_BASE_URL	https://example.audiocodes.com

- **Dynamic (ideally set on runtime per task):**

Key	Example value
TENANT	demo
USERNAME	user
PASSWORD	Psw12345!
SUPERVISOR_EMAIL	user@example.com
FILES_DIR	./files
BASE_LANGUAGE	"Hebrew"
GENERIC_ADDITIONAL_INFO	'{"generic_key":" generic_value"}'

1.2 Workflow Overview

1. **Pre-requisites:**

   The MIAOP system must be installed, with the appropriate tenant configured. The tenant must include a valid (existing) user account for authentication when performing REST API operations.

2. **Parameter definition:**

   Establish the static and/or dynamic workflow parameter values (either from a configuration or at runtime).

3. **Authentication:**

   Perform authentication against the Keycloak identity provider. A valid access token is acquired for subsequent API calls.

> It's recommended to either obtain a new access token before every request that requires an Authorization header, or automatically refresh the access token whenever it's about to expire (see Section 'Authentication – Get Access Token').

4. **Task submission:**

   The audio files are submitted to the Meeting Insights On-Prem backend as part of a new offline transcription task.

   The task is assigned to a task supervisor, who is set to be the task owner.

   Each file is registered as a separate **"meeting"** within the task context.

   The submission is performed via a single API request using the previously acquired access token.

   The response to the submission requests contains the unique identifier of the task (taskId).

> The content of AI summaries is determined by the default template defined for the selected tenant or by the system default configuration.

5. **Polling for task status until completion:**

   See Section 'Polling Task Status'.

6. **Output export:**

   Once the task is completed, the API can be used to retrieve meeting metadata and transcription results for all meetings included in the task, which are listed in the task metadata under "meetings".

   See sections 'Get Task Metadata' and 'Get Meeting Transcription'.

   The server returns the meeting metadata and transcription results in structured JSON format. These results can be consumed programmatically or saved locally as files for further processing.

7. **Listing existing tasks:**

   At any given time, the API can be used to list the tasks in the system for a given tenant, task supervisor, and start / end dates. See section 2.4 Get Tasks List.

1.2.1 Polling Task Status

Once the task is submitted and a taskId is received, the system begins processing each audio file asynchronously.

The application is expected to start polling the task metadata after a few seconds to be aware of the task status and completion. See Section ‘Get Task Metadata’.

Polling should continue at a steady interval until the task status is either "error" or "sttDone".

Polling can be optimized, as described in Section ‘Optimize Polling’.

1.2.1.1 Optimize Polling

For each polling request, if the response metadata "taskStatus" == "sttProcessing" and the previous response is "taskStatus" != "sttProcessing" (i.e. the task has just entered the processing state), the next polling request should be performed after "sttRemainingTime" seconds.

The sttRemainingTime property is a rough estimation which may be inaccurate. It assists with leveraging the polling frequency; however, the amount of files and system load can increase the margin of error for this value..

2 API Reference

2.1 Authentication – Get Access Token

Method:

```
POST
```

URL:

```
{KEYCLOAK_BASE_URL}/realms/{TENANT}/protocol/openid-connect/token
```

Purpose:

Obtain an access token from Keycloak.

Headers:

```
Content-Type: application/x-www-form-urlencoded
```

Body Parameters:

Parameter	Type	Value Source	Description
`client_id`	String	Browser_client	Keycloak client ID.
`grant_type`	String	"password"	Must be a password.
`username`	String	USERNAME	End-user login.
`password`	String	PASSWORD	End-user password.

2.1.1 Response JSON Body

Property	Data Type	Notes
`access_token`	String	Bearer access token for authentication.
`expires_in`	Integer	Total lifetime of the access token from the moment it was granted. This can be used to calculate when the token should be retrieved again.

2.2 Task Submission – Upload Offline Task

Method:

```
POST
```

URL:

```
{API_BASE_URL}/hrec
```

Purpose:

Submits a new offline transcription task, including one or more input files and task metadata.

Headers:

```
Authorization: Bearer <access_token>
Content-Type: multipart/form-data
```

Form Fields:

Field	Type	Required	Description
`task`	**TaskMetadata** JSON string (see request payload below)	Yes	Serialized task metadata.
`soundBlob`	file	Yes	One or more audio files to transcribe.

2.2.1 Request Payload

2.2.1.1 TaskMetadata Object Structure

Property	Data Type	Notes
`taskId`	String	(Optional) Determined by the application or auto-generated if not provided. The taskId is returned in the response body.
`tenant`	String	Equivalent to the *TENANT* setting.
`taskName`	String	Equivalent to the *TASK_NAME* setting.
`taskType`	String	Must be set to "offline".
`taskCreationTime`	String	Set to the current date and time.
`taskStatus`	String	Must be set to "sttPending".
`baseLanguages`	Array of String.	Specifies the task's meeting baseLanguage values (currently singular). Valid values are the supported system languages, listed in the MIAOP version release notes. For example: ["Hebrew"] or ["English"]. The value is case-sensitive and must match one of the supported system languages.
`genericAdditionalInfo`	String	A generic JSON object (maximum size: 10 KB) used for adding custom attributes as needed. Must be valid JSON. Example: {"generic_prop1":"generic_val1"}
`meetings`	Array of MeetingDesc objects	
`supervisor`	SupervisorInfo object: {name: string, email: string}	Equivalent to the *SUPERVISOR_EMAIL* parameter. Example: {name: "", email: "user@ac.com"}

2.2.1.2 MeetingDesc Object Structure

Property	Data Type	Notes
`files`	Array of a **single item** of type: {file: string, baseLanguage: string}	**file**: File name and extension **baseLanguage**: Preferred transcription language. The value **must** exist in the task baseLanguages array. Example: [{file: recording-1.mp3, baseLanguage: "Hebrew"},]

2.2.2 Response Payload JSON

Property	Data Type	Notes
`taskId`	String	The taskId of the submitted task.

2.3 Get Task Metadata

Method:

```
GET
```

URL:

```
{API_BASE_URL}/api/tasks/{taskId}
```

Purpose:

Retrieves metadata related to a specific task.

Headers:

```
Authorization: Bearer <access_token>
```

Path Parameters:

Parameter	Type	Required	Description
`taskId`	string	Yes	The taskId value which was defined or generated when uploading the task.

2.3.1 Response Payload JSON: TaskMetadataResponse Object Structure

2.3.1.1 TaskMetadataResponse Object Structure extends TaskMetadata

Property	Data Type	Notes
`meetings`	Array of JSON objects: {meetingId: string}[]	Each item corresponds to the item in the "files" array in the task submission request, and defines its meetingId. These meetingId values can be used to obtain metadata and transcription for individual meetings.
`taskStatus`	String	Can be one of the following: ■ "sttPending" ■ "sttProcessing" ■ "sttDone" ■ "error"
`sttRemainingTime`	Number (Integer)	Estimated total time (in seconds) for the task to be in status "sttProcessing".

2.4 Get Tasks List

Method:

```
GET
```

URL:

```
{API_BASE_URL}/api/tasks?filter={filterSting}
```

Purpose:

Retrieves the current tasks in the system, filtered by a filter string

Headers:

```
Authorization: Bearer <access_token>
```

Query Parameters:

Parameter	Type	Required	Description
`filterString`	URL-encoded JSON-string	Yes	URL-encoded JSON string which can define filtering of the tasks list.

2.4.1 filterString Syntax

The filterString parameter must be a URL-encoding of a JSON string, created from a JSON object below.

All fields are optional filters.

```
{  "$and": [
      { "extraData.taskUpdateTimeNumeric": { "$gte": 1754006400, "$lt": 1754438400 } },
      { "extraData.archive": { "$ne": true } },
      { "extraData.tenant": "test" },
      { "extraData.supervisor.email": "test@test.co.il" }
      { "extraData.supervisor.name": "test supervisor" }
] }
```

Description:

- **extraData.taskUpdateTimeNumeric**

  Filters tasks by update timestamp range, given as numeric GMT epoch-time values.
  - **$gte:** Greater than or equal to start timestamp
  - **$lt:** Less than end timestamp
- **extraData.archive**

  Ensures only non-archived tasks are returned.

  This attribute is recommended to be always included.
  - **$ne:** True means "not archived"

  Excludes archived tasks from the result.
- **extraData.tenant**

  Filters tasks by tenant name. In this example, it's "test".

  This is useful when the authenticated user is on the system level (user tenant is "master")
- **extraData.supervisor.email / name**

  Filters tasks by the supervisor's email address or name.

  It returns only tasks assigned to the supervisor's email or name.

2.4.2 Response Payload JSON

Array of TaskMetadataResponse objects.

2.5 Get Meeting Metadata

Method:

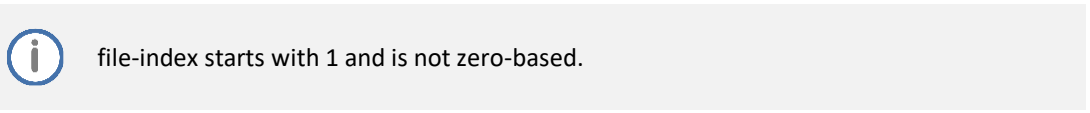
```
GET
```

URL:

```
{API_BASE_URL}/api/meetings/{meetingId}
```

Purpose:

Retrieves metadata related to a specific meeting created as part of the task. The meeting is referred to by its `meetingId` property, whose value is <file-index>.<taskId>.



file-index starts with 1 and is not zero-based.

For example: For a task with taskId "abcdef", and uploading files [file1.mp3, file2.mp3…], you can get the meeting metadata of file1 by using `meetingId = 1.abcdef`.

Headers:

```
Authorization: Bearer <access_token>
```

Path Parameters:

Parameter	Type	Required	Description
`meetingId`	string	Yes	Meeting identifier (<index>.<taskId>).

2.5.1 Response Payload JSON

Property	Data Type	Notes
`meetingId`	String	Given as <index>.<taskId>.
`metadata`	JSON object	
`metadata.status`	String	Can be one of the following: ■ "sttPending" ■ "sttProcessing" ■ "sttDone" ■ "error"
`metadata.errorInfo`	JSON object: {name: string, message: string, errorLevel: "warning" or "critical"}	

2.6 Get Meeting Transcription

This should be performed once the status received from the previous request is "sttDone".

Method:

```
GET
```

URL:

```
{API_BASE_URL}/api/stt/{meetingId}/0
```

Purpose:

Retrieves the transcription (speech-to-text) results for a specific meeting.

Headers:

```
Authorization: Bearer <access_token>
```

Path Parameters:

Parameter	Type	Required	Description
`meetingId`	String	Yes	Same format as above.

2.6.1 Response Payload JSON

Property	Data Type	Notes
`transcription`	Array of ParagraphDesc Objects	

2.6.1.1 ParagraphDesc Object Structure

Property	Data Type	Notes
`duration`	Number (integer)	Denotes the paragraph duration.
`Id`	String	Denotes speaker id.
`location`	Number (integer)	Denotes the audio location.
`text`	String	Total paragraph text.
`words`	Array of Word items	Paragraph words data.

3 Examples and Reference Implementation

This section provides practical examples that demonstrate how to interact with the Meeting Insights On-Prem backend using the described API workflow, which includes the following:

- Bash-based execution instructions.
- Configuration structure (via config.env).
- JavaScript code samples for each API step.

A complete reference script (offline-transcription.sh) is available to automate the full workflow.

3.1 Execution Instructions

This script is executed via a Bash shell and supports both predefined configuration and runtime arguments.

By default, all values are loaded from a config.env file located in the same directory as the script. However, users may override any of the following parameters using command-line arguments:

Flag	Description	Required	Example
`-u`	Username	No	-u user
`-p`	Password	No	-p mypass123
`-t`	Tenant name	No	-t myTenant
`--tn`	Task name	No	--tn "My Offline Task"
`-d`	Input folder path containing WAV files	No	-d ./samples
`-s`	Supervisor email	No	-s example@demo.co.il
`-l`	Base language	No	-l Hebrew
`--gai`	Generic additional information (JSON object)	No	--gai '{"generic_prop1":"generic_val1"}'

3.2 Execution Example on a Linux Machine

1. Connect to the target machine:

   Open PuTTY (or your preferred terminal client) and connect to the Linux host.
2. Navigate to the script directory:

   cd /home/myproject/createOfflineTranscription
3. Make the script executable:

   chmod +x offline-transcription.sh
4. Run the script:
   - *** Option 1:** Use the default configuration from `config.env`:

     ./offline-transcription.sh
   - *** Option 2:** Override specific values at runtime**:

     ./offline-transcription.sh --tn "New Offline Task" -s "test@audiocodes.com"

All command-line arguments are optional and will override only the corresponding values from `config.env`.

3.3 Configuration File (config.env)

The following environment variables must be defined in a file called config.env.

These values are loaded by the script before execution:

- KEYCLOAK_BASE_URL=https://example.com:8443
- API_BASE_URL=https://example.com
- TENANT=my-tenant
- SUPERVISOR_EMAIL=admin@example.com
- TASK_NAME=Offline Upload Task
- FILES_DIR=./files
- USERNAME=user
- PASSWORD=123
- BASE_LANGUAGE=Hebrew
- GENERIC_ADDITIONAL_INFO='{"generic_prop1":"generic_val1"}'

3.4 Get Access Token

```
const params = new URLSearchParams({
  client_id: "browser_client",
  grant_type: "password",
  username: "user@example.com",
  password: "your-password"
});
const response = await fetch("https://example.com:8443/realms/my-
tenant/protocol/openid-connect/token", {
  method: "POST",
  headers: { "Content-Type": "application/x-www-form-urlencoded"
},
  body: params
});
const data = await response.json();
const accessToken = data.access_token;
```

Response Example:

```
{
  "access_token": "eyJhbGciOi...",
  "expires_in": 300,
  "refresh_expires_in": 1800,
  ...
}
```

3.4.1 Upload Task (with Audio File)

```
const formData = new FormData();

const task = {
  tenant: "my-tenant",
  taskName: "Offline Upload Task",
  taskType: "offline",
  taskCreationTime: "15.07.2025 10:30",
  taskUpdateTime: "15.07.2025 10:30",
  taskStatus: "sttPending",
  baseLanguages: ["Hebrew"],
  meetings: [
    {
      files: [{ file: "example_input_file",
baseLanguage
: "Hebrew" }],
      aiGenerationMode: "auto"
    }
  ],
  docSections: [],
  supervisor: { name: "", email: "admin@example.com" }
};

formData.append("task", JSON.stringify(task));
formData.append("soundBlob", yourAudioBlob, "example_input_file");

const uploadResponse = await fetch("https://example.com/hrec", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${accessToken}`
  },
  body: formData
});
const uploadResponseJson = await uploadResponse.json();
const taskId = uploadResponseJson.taskId;
```

Response Example:

```
{ "taskId": "7uq0wa6b0bdugcjy" }
```

3.5 Get Task Metadata

```
const metadataResponse = await
fetch(`https://example.com/api/task/${taskId}`, {
  headers: { Authorization: `Bearer ${accessToken}` }
});
const taskJson = await metadataResponse.json();
const taskStatus = taskJson.taskStatus;
const sttRemainingTime = taskJson.sttRemainingTime;
```

Response Example:

```
{
  "taskId": "123e4567-e89b-12d3-a456-426614174000",
  "taskStatus": "sttDone",
  "sttRemainingTime": 1000,
  ...

}
```

3.6 Get Tasks List

```
const startDate = 1754006400000
const endDate = 1754438400000
const tenant = "testTenant"
const supervisorEmail = "test@test.co.il"
const filter = {
  "$and": [
    { "extraData.taskUpdateTimeNumeric": { "$gte": startDate,
"$lt": endDate } },
    { "extraData.archive": { "$ne": true } },
    { "extraData.tenant": tenant },
    { "extraData.supervisor.email": supervisorEmail }
  ]
}
const filterString =
`?filter=${encodeURIComponent(JSON.stringify(filter))}`

const response = await
fetch(`https://example.com/api/tasks${filterString}`, {
  headers: { Authorization: `Bearer ${accessToken}` }
});
const tasksList= await response.json();
```

Response Example:

```
[
{
  "taskId": "123e4567-e89b-12d3-a456-426614174000",
  "taskStatus": "sttDone",
  "sttRemainingTime": 1000,
  ...
},
{
  "taskId": "123e4567-e89b-12d3-a456-426614174000",
  "taskStatus": "sttDone",
  "sttRemainingTime": 1000,
  ...
},
...
]
```

3.7 Get Meeting Metadata

```
const meetingId = "1.123e4567-e89b-12d3-a456-426614174000";
const metadataResponse = await
fetch(`https://example.com/api/meetings/${meetingId}`, {
  headers: { Authorization: `Bearer ${accessToken}` }
});
const meetingJson = await metadataResponse.json();
```

Response Example:

```
{
  "meetingId": "1.123e4567-e89b-12d3-a456-426614174000",
  "metadata":
{
  "status": "sttDone",
  "files": [...],
  ...
}
}
```

3.8 Get Meeting Transcription Results

```
const sttResponse = await
fetch(`https://example.com/api/stt/${meetingId}/0`, {
  headers: { Authorization: `Bearer ${accessToken}` }
});
const sttJson = await sttResponse.json();
```

Response Example:

```
{
  "transcription": [
    {
      "id": "דובר-1",
      "location": 0,
      "duration": 2200,
      "text": "היי שלום לכולם",
      "words": [
        { "word": "היי", "location": 0, "duration": 700,
"confidence": 0.975 },
        { "word": "שלום", "location": 700, "duration": 200,
"confidence": 0.940 },
        { "word": "לכולם", "location": 900, "duration": 1300,
"confidence": 0.958 }
      ]
    }
  ]
}
```

International Headquarters
Naimi Park
6 Ofra Haza Street
Or Yehuda, 6032303, 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-26022