



OMNI INTELLIGENCE

OMNI Voice Biometrics

Confidentiality and non-disclosure

This document contains information, which is proprietary and confidential to Omni Intelligence Pty. Ltd. Except as expressly and directly required by the intended recipient for evaluation of the content of this proposal, this document and material information contained herein may not be copied, distributed, or circulated in any form without explicit prior written consent from Omni Intelligence Pty. Ltd.

Contents

OMNI Voice CLI	4
Prerequisites	4
Running the CLI	4
CLI Examples	4
Enrolment	5
Voice Verification	5
Voice Log-in	5
Voice Log-in Strong	6
Voice Sample Guidelines	7
Abstract	7
Key-Points Summary	7
Recommendations Summary	8
Enrolment vs Authentication	9
Net Speech vs Audio Duration	9
Performance Across Multiple Channels	10
Same-Phrase Strategy	11
Audio Quality and Environmental Factors	11
Automatic Speech Recognition (ASR)	12
Voice Recording Instructions to Users	12
Good Authentication Prompt Examples	13
Good Enrolment Prompt Examples	13
Non-Recommended Prompt Examples for <i>Voice Log-in</i> and <i>Voice Log-in Strong</i> scenarios	13
Authentication Flows	14
Abstract	14
OMNI Voice Flows	14
Enrolment	14
Voice Verification	15
Voice Log-in Basic	16
Voice Log-in Strong	17
Biometrics API Reference	18
Abstract	18
Header-Based API Parameters	18

Audio Upload	19
Authorization Header Scheme.....	20
Error Handling	20
Data Structures	21
Workflow Types	22
Workflow States	22
API Reference.....	22
POST enrollVoiceSample.....	23
GET startVoiceVerification.....	23
POST voiceLogin.....	24
POST voiceLogin_Strong	25
POST authenticate	26
GET authenticateWithCode	26
Security Model Reference	28
Abstract.....	28
Accessing OMNI Voice API	28
Data & API Classification	29
Table 1: Data Classes across all OMNI Voice apps and the platform:	30
Table 2: API classes (<i>surfaces</i>) across all OMNI Voice apps and the platform:	30
Security Architecture	31
Platform	31
Biometrics	32
Data In-Transit Policy	33
Genesys Integration	34
Abstract.....	34
OMNI Voice Media Server	34
AudioHook Integration	34
OMNI Voice Data Actions:	36
Telephony Appliance (Cloud).....	37
Telephony Appliance (OnPrem+BYOCC).....	38
Genesys Cloud Installation	39
Prerequisites	39
Phone Trunk Configuration.....	39
AudioHook Integration	40
Transcribe Feature	43

Enable Transcribe/AudioHook on desired queues	44
Permission Prerequisites	44
Configuration steps:.....	44
Import OMNI Voice Data Actions and Architect Flow modules	45
Create or modify IVR's	47
Screen Popup Configuration	49

OMNI Voice CLI

The easiest way to get started with OMNI Voice is to use the Command Line Interface (CLI) that can be run on Windows, Linux and macOS.

The CLI is built with .net 6 so you'll need to install the runtime (<https://dotnet.microsoft.com/en-us/download/dotnet/6.0>) to be able to run it.

A complete API reference is available [here](#). See also [Best Practices](#) that explain the speech sample requirements in detail.

Prerequisites

Be sure to familiarize yourself with [OMNI Voice Flows](#) and [Security Model](#) (at least the "Accessing OMNI Voice API" part) first.

To run the API, you will need to:

1. Obtain API Keys (Publishable and Private)
2. Find out your public IP Address and add it to the ACL <https://omnivoice.tech/biometrics/manage-API-keys>

Running the CLI

The CLI accepts command-line parameters the *usual* way and outputs the returned results in JSON format. A complete list of parameters is as follows:

```
> biometrics-api-cli.exe

OMNI Voice.tech by Omni Intelligence; https://omnivoice.tech/
biometrics-api-cli.exe (Windows) or api-client-cli.dll (Non-Windows)
*****
Usage: dotnet biometrics-api-cli.dll [--verbose (optional)] [...switches]
Switches:
  -u, --api-url          URL of the OMNI Voice biometrics backend; defaults to https://omnivoice.tech/biometrics/api
  -k, --api-key          A required publishable or private API key; see https://omnivoice.tech/biometrics/docs/api-keys for details
  -h, --channel          Channel of the operation; defaults to 'web'
  -o, --api-operation    A required API operation to be executed; possible values are: 'voice_verification', 'voice_login', 'voice_login_strong', 'authenticate_wav',
'authenticate_code' and 'Enrol'
  -n, --id-name          The name of speaker identifier (either 'account' or 'phone'); defaults to 'phone', if not provided; required for 'voice_verification', 'voice_login',
'voice_login_strong' and 'Enrol' operations
  -v, --id-value         The value of speaker identifier; required for 'voice_verification' and 'Enrol' operations
  -f, --wav-filename     The name of the wav file; required for 'voice_login', 'voice_login_strong', 'authenticate_wav' and 'Enrol' operations
  -m, --metadata         A string up to 1024 bytes long; may be optionally provided for any of the API operations
  -w, --workflow-id      A workflow uuid identifier that is required for 'authenticate_wav' and 'authenticate_code'
  -c, --verification-code The verification code is required only for 'authenticate_code'
  -l, --language-code    A bcp47 language code required for 'voice_login', 'voice_login_strong' and 'authenticate_wav'; see https://omnivoice.tech/biometrics/docs/supported-
languages for all supported languages
The output of the command is a JSON structure described in detail here: https://omnivoice.tech/biometrics/docs/biometrics-api
```

CLI Examples

Enrolment

You will need to record a wav file containing your voice (see also [Best Practices](#) that explain the speech sample requirements in detail). The wav file format is documented [here](#).

Note: The mobile phone number must be provided in E.164 format (e.g. +1234567890)

Note: Enrolment requires a valid phone number; it will not complete until the enrolled speaker clicks on the link sent to their mobile phone

```
> biometrics-api-cli.exe -k [PASTE PRIVATE API KEY HERE] -v [YOUR MOBILE PHONE NUMBER] -o Enrol -f [PATH TO WAV AUDIO]
```

Voice Verification

You will need to record a wav file containing your voice (see also [Best Practices](#) that explain the speech sample requirements in detail). The wav file format is documented [here](#).

Note workflow ID - it is returned by the first command and must be passed to the second command with the -w key

```
> biometrics-api-cli.exe -k [PASTE PUB API KEY HERE] -v +1234567890 -o voice_verification
{
  "id": "853121cd-63f8-4809-9463-190a70ad5e7f",
  "WorkflowType": "authentication",
  "state": "identified",
  "metadata": "",
  "expired": false,
  "identifier": "\u002B1234567890",
  "authentications": [],
  "success": true,
  "error": null
}
> biometrics-api-cli.exe -k [PASTE PUB API KEY HERE] -v +1234567890 -o authenticate_wav -f [PATH TO WAV AUDIO] -w 853121cd-63f8-4809-9463-190a70ad5e7f
{
  "id": "853121cd-63f8-4809-9463-190a70ad5e7f",
  "WorkflowType": "authentication",
  "state": "authenticated",
  "metadata": "",
  "expired": false,
  "identifier": "\u002B1234567890",
  "authentications": [
    {
      "timestamp": "2022-09-15T03:13:21.4680457+00:00",
      "sampleLength_sec": 6.69,
      "sampleQuality": 1,
      "score": 0.792033600131906
    }
  ],
  "success": true,
  "error": null
}
```

Voice Log-in

You will need to record a wav file containing your voice (see also [Best Practices](#) that explain the speech sample requirements in detail). The wav file format is documented [here](#).

```
> biometrics-api-cli.exe -k [PASTE PUB API KEY HERE] -v +1234567890 -o voice_login -f [PATH TO WAV AUDIO]
{
  "id": "49500a56-f2d5-4abd-bc61-fafe6b6bb8d5",
  "WorkflowType": "authentication",
  "state": "authenticated",
  "metadata": "",
  "expired": false,
  "identifier": "\u002B1234567890",
  "authentications": [
    {
      "timestamp": "2022-09-15T03:28:20.2895252+00:00",
      "sampleLength_sec": 6.72,
      "sampleQuality": 1,

```

```
"score": 0.804761457555627
},
"success": true,
"error": null
}
```

Voice Log-in Strong

You will need to record a wav file containing your voice (see also [Best Practices](#) that explain the speech sample requirements in detail). The wav file format is documented [here](#).

Note workflow ID - it is returned by the first command and must be passed to the second command with the -w key

Note: see state of the workflow returned by the first command - it is "identified" and not "authenticated" meaning that the workflow expects a verification code is expected

The second command passes authentication code sent to the mobile phone.

```
> biometrics-api-cli.exe -k [PASTE PUB API KEY HERE] -v +1234567890 -o voice_login_strong -f [PATH TO WAV AUDIO]
{
  "id": "1824c51f-0eb7-4ba3-8eea-7de9f0712f3e",
  "WorkflowType": "authentication",
  "state": "identified",
  "metadata": "",
  "expired": false,
  "identifier": "\u002B1234567890",
  "authentications": [
    {
      "timestamp": "2022-09-15T03:29:31.0029064+00:00",
      "sampleLength_sec": 6.72,
      "sampleQuality": 1,
      "score": 0.804761457555627
    }
  ],
  "success": true,
  "error": null
}

> biometrics-api-cli.exe -k [PASTE PUB API KEY HERE] -v +1234567890 -o authenticate_code -c [VERIF CODE] -w 1824c51f-0eb7-4ba3-8eea-7de9f0712f3e
```

Voice Sample Guidelines

Abstract

Intuitively, it is clear how a Voice Biometrics technology works: it compares voice samples and provides comparison results; if the results show a good match (usually, expressed as a numeric score), the voice samples are considered to be coming from the same person.

In reality, there are various factors that may impact the performance of the technology. Some of them include:

- Background noise
- Presence of multiple speakers during voice sample collection
- Channel (e.g. phone line vs laptop mic)
- Mic gain distortions

In addition, human voice is a very versatile *device* that arguably exhibits numerous properties depending on a vocal task being performed (e.g. singing vs speaking). OMNI Voice is a practical technology and it's specifically trained on human speech. It is hard to define specific properties that can be captured here. Indeed, depending on the context and our emotional state, even the same phrase may have different emphases that communicate complex semantics.

OMNI Voice Biometrics engine is an artificial neural network designed to capture a combination of properties of individual voice as well as the way the individual speaks.

Even though OMNI Voice Biometrics engine is language / phrase agnostic, its performance will vary depending on the contents and duration of speech samples. The general rule is that the longer a speech sample is, the more voice properties it will expose and, subsequently, the better its performance gets.

This data sheet provides recommendations around the collection of speech samples for various scenarios to maximize voice biometrics performance and customer satisfaction.

Key-Points Summary

- **Enrolment** audio should be longer than **Authentication** audio
- **Net Speech** duration is less than audio file duration in most cases
- Cross-channel authentications will show lower authentication scores; increase speech sample length or use same-phrase strategy to mitigate
- Encourage users to make multiple Enrolments for each of the channels they normally use

- Min. **Net Speech** requirement can be lowered when similar phrases are used for both, **Enrolment** and **Authentication**
- Voice samples containing multiple speakers will affect system performance
- Avoid Enrolling multiple speakers against the same account / phone number
- Speech Recognition is not used for Voice Biometrics; it is used to improve *Customer Experience*
- Provide clear instructions to users on what they should say during the speech sample collection process
- Customer identifiers should be numeric; speech samples containing spoken identifiers must not have any other numbers in them
- Numeric identifiers work best with Automatic Speech Recognition and yield reliable performance; do not use non-numeric speaker identifiers

Recommendations Summary

Parameter	Recommended Value	Notes
Min Enrolment Audio (Random)	30 sec	Net speech requirement for Enrolment that is agnostic to language / speech content
Min Enrolment Audio (Phrase-dependant)	10 sec	Net speech requirement for Enrolment - specific phrase
Min Auth. Audio (Random)	10 sec	Net speech requirement for authentication - content agnostic
Min Auth. Audio (Phrase-dependant)	6 sec	Net speech requirement for authentication - specific phrase
Score Threshold	0.7	Maybe lowered for cross-channel scenarios

Parameter	Recommended Value	Notes
Verification Code Complexity	6	Number of digits in the verification code
Audio / Net Speech Ratio	1.4:1	Typical ratio between an audio file and net speech extracted from it. E.g. a 14 sec audio file will typically contain 10 sec of net speech

Enrolment vs Authentication

Key-Point: Enrolment Audio should be longer than Authentication Audio

There are two main scenarios involved in OMNI Voice:

- Enrolment - the initial (or reference) voiceprint registration
- Authentication - subsequent verification of the person's voice (e.g. when a previously registered person wishes to log-in using their voice)

The main difference between these two scenarios is that **Enrolment** is done quite rarely (maybe once or twice) but it requires that the person being Enrolled is *authentic*. We also try to capture as much of the person's voice as practically possible to improve the quality of their **Enrolment** voiceprint.

Contrary to **Enrolment**, **Authentication** is done each time a previously Enrolled person wishes to gain access to a resource (e.g. their website account). During **Authentication**, a fresh voice sample is collected and compared against the **Enrolment** voiceprint. If the **Authentication** fails, another attempt can be re-tried provided that voice sample collection is quick and easy (i.e. short audio samples).

Net Speech vs Audio Duration

Key-Point: Net Speech duration is less than audio file duration

While audio files may capture a wide variety of environmental sounds (including silence), OMNI Voice requires just human speech as its input. In particular, OMNI Voice performance is rated based on the amount of speech only, which we refer to as **Net Speech**.

OMNI Voice uses a number of techniques to filter non-speech intervals from the input audio files. This means that typically, the amount of net speech extracted from a recording will be less than the recording duration itself.

For short phrases (range 10-20 seconds), the average ratio between audio duration and **Net Speech** is 1.4:1. This, however, will depend on whether a person reads some unfamiliar text or says something that they are *comfortable* with. This ratio may have to be adjusted depending on the phrase strategy used.

Performance Across Multiple Channels

Key-Point: Cross-channel authentications will show lower authentication scores; increase speech sample length or use same-phrase strategy to mitigate

Key-Point: Encourage users to make multiple Enrolments for each of the channels they normally use

Cross-channel authentication happens when **Enrolments** and **Authentications** arrive on different channels. For example, a person could be registered (Enrolled) on a telephone call; after that they could try to authenticate themselves by voice using a web browser (e.g. logging-in to a website).

In this case, the voice sample audio properties submitted over the phone will be different to those of authentication; this will usually lower the comparison score between the cross-channel audio samples.

There are many reasons for it, including:

- Different microphone acoustics
- Different sampling rates which affect audio quality (e.g. telephony audio runs at 8,000 samples per second while browsers can record at 16,000 and above)

There are several strategies to mitigate a natural performance degradation for cross-channel authentications:

1. Increase minimal speech sample duration for authentications
 - The longer the speech sample, the more speech / voice properties can be captured which improves the comparison scores even in cross-channel situations
2. Use *Same-Phrase* strategy
 - The system performs better when comparing similar speech contents (i.e. Enrolment and authentication both contain the same spoken phrase)
3. OMNI Voice can have multiple **Enrolments** per each person
 - It is recommended to Enrol more than 1 voice sample for each channel they might come in through
4. Require that your users submit a verification code using audio (**Voice Login Strong** scenarios)
 - A verification code sent for 2nd Factor Authentication can be recorded as well and sent to OMNI Voice as an audio recording; in this case, the recording

will be appended to the initial speech sample thus increasing the amount of **Net Speech** provided

5. Use a combination of the above

Example:

- It is relatively easy to get a user to record a little more of their voice when they're trying to log-in. For example, you can instruct them to say their full name, town and suburb followed by their phone number; the phone number part will then be used by OMNI Voice speech recognition to identify the user, followed by the voiceprint comparison
- During Enrolment, ask the user to also say their name, address, town and suburb followed by counting from 0 to 9; this will provide a good amount of speech while keeping Enrolment and authentication phrases fairly similar

Same-Phrase Strategy

Key-Point: Min. **Net Speech** requirement can be lowered when similar phrases are used for both, **Enrolment** and **Authentication**

While OMNI Voice Biometrics engine has been trained to recognise people without any dependency on what they actually say, it poses certain requirements on the amount of speech the engine has at its disposal. For arbitrary speech samples it has been estimated that for the system to perform well, the minimum amount of **Net Speech** for **Enrolment** and **Authentication** should be 30 and 10 seconds respectively.

While it is a remarkable result, collecting 10 seconds of **Net Speech** each time might prove to be difficult in some scenarios. It has also been shown that the **Min. Net Speech** requirement can be lowered when *same-phrase* approach is used. In this approach, both **Enrolment** and **Authentication** would be done with voice samples capturing the same phrase; in this case, the Voice Biometrics engine *faces* an easier task because it compares *apples* with *apples* and so its performance is better which is reflected in the comparison score.

The phrases used for **Enrolment** and subsequent **Authentications** do not have to match exactly. For this to work, it is sufficient if the phrases are just similar. One of the examples could be the address: two different addresses uttered by the same person will likely produce a high comparison score allowing us to conclude a positive authentication result.

Audio Quality and Environmental Factors

Key-Point: Voice samples containing multiple speakers will affect system performance

Key-Point: Avoid Enrolling multiple speakers against the same account / phone number

While OMNI Voice has been designed to be resilient to things like noise, large signal-to-noise ratios (SNR) will degrade system performance significantly. The audio quality rating returned by the system reflects automatic audio quality assessment. There are factors, however, that cannot be assessed by simply studying SNR or other *objective* audio properties.

It has been shown that when audio samples sent for voice matching contain multiple speakers talking, system performance will degrade. For this reason, if multiple people need to be Enrolled against the same account, it is best to Enrol them to OMNI Voice individually. A recommended approach in this situation is to provide a unique identifier (e.g. phone number) for each person being Enrolled in the system and implement *authorized person* policy in the internal system of the organization.

Automatic Speech Recognition (ASR)

Key-Point: Speech Recognition is not used for Voice Biometrics; it is used to improve *Customer Experience*

Several [Authentication Flows](#) implemented in the system rely on ASR for two reasons:

- Identifier extraction (e.g. phone number)
- Verification code extraction

The main reason to use ASR is to reduce the number of steps required to complete a voice-enabled log-in process. Indeed, when a customer is asked to say their identifier as well as verification code, with just two steps we can:

1. Use this recording for voice authentication
2. Find out who the customer are (their account / phone number)
3. Complete 2-Factor-Authentication for enhanced security

This is all done mostly hands-free and achieves a great *Customer Experience* (CX). This approach, however, introduces language dependency because ASR needs to know which language it is dealing with. The language detection is done routinely by the majority of OS Browsers and will not require any manual selection unless the default browser language is less preferable. This choice could be deferred to the customer.

Voice Recording Instructions to Users

Key-Point: Provide clear instructions to users on what they should say during speech sample collection process

Key-Point: Customer identifiers should be numeric; speech samples containing spoken identifiers must not have any other numbers in them

OMNI Voice relies on audio files containing human speech at its core and the system performance will greatly depend on the quality of the collected speech samples. The key to collect good quality speech samples is to provide simple-to-follow, clear, and unambiguous instructions to the user at collection time.

One of the important aspects is that if *Voice Login* or *Voice Login Strong* is used (see [Authentication Flows](#)), the speech sample should contain only a single numeric identifier (e.g. phone number) and no other numbers. This is required only for **Authentication** speech samples but not for **Enrolment**.

Note: Numeric identifiers work best with Automatic Speech Recognition and yield reliable performance

Good Authentication Prompt Examples

- Please state your full name (including any middle names) and your mobile phone number
- Please state your name, town and the ACME account number
- Please state your suburb, town and mobile phone number

Good Enrolment Prompt Examples

- Please state your full name (including middle names) and your mobile phone number
- Please state your office address followed by your mobile phone number
- Please state your full name and count 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Non-Recommended Prompt Examples for *Voice Log-in* and *Voice Log-in Strong* scenarios

- Please say something for 10 seconds... (too vague, may not have any identifiers)
- Please state your address and mobile phone number (address may contain numbers)
- Please state your name, postcode and mobile phone number (postcode is usually numeric)
- Please state your account number followed by your phone number (two identifiers will not work)

Authentication Flows

Abstract

This data sheet provides a detailed description of the flows to enable voice authentication in a third-party system (e.g. a website or telephony). The flowcharts given in this paper are designed around the assumption that there is some sort of User Interface to present information (e.g. error messages). However, the same basic flows apply to devices that have no user interface (e.g. telephony) where information can be conveyed using voice prompts.

There are two fundamentally different scenarios that are involved in the process of identification / authentication by voice:

- **Enrolment,**
- **Login / Authentication**

The way we do the **Login / Authentication** is we generate a *voiceprint* based on the given audio that contains a speech sample. The *voiceprint* is then compared against each of the reference *voiceprints* stored in the org's database. The best match that passes a *confidence threshold* is then returned. Among other parameters documented below, this match contains:

- Comparison Score (a value between [0..1])
- Speaker's Identifier (a phone number) that corresponds to the best match

The **Enrolment** scenario is required to populate the reference database in the first place. This is done by calling the **Enrolment API**.

OMNI Voice Flows

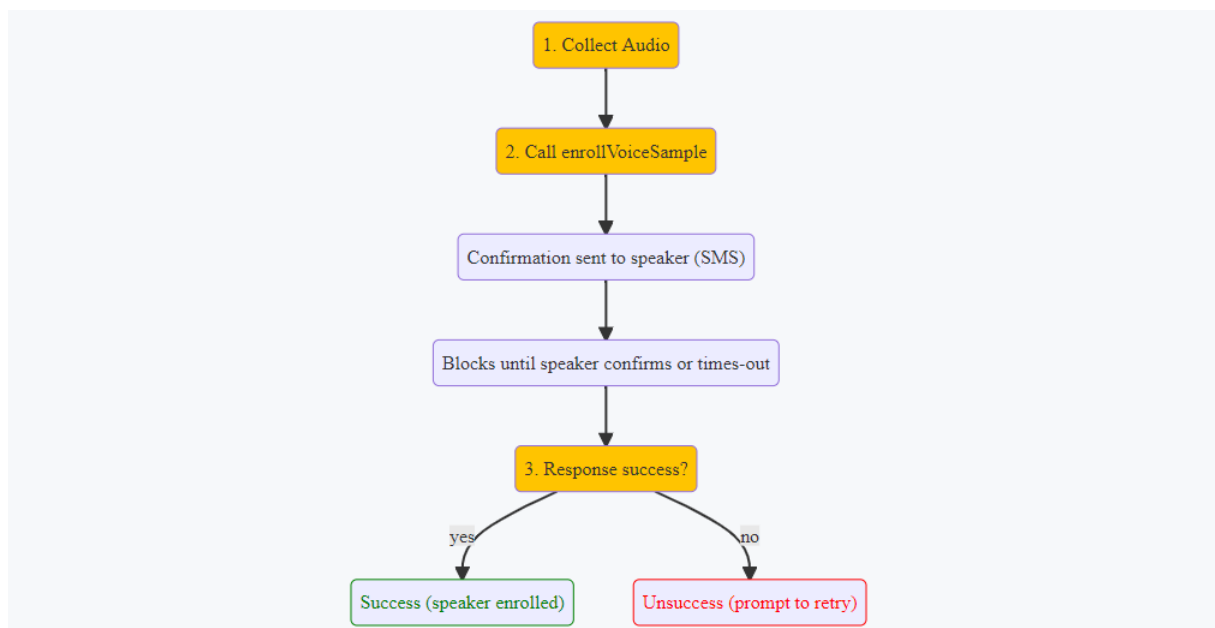
Enrolment

The flow sends an audio file containing speaker's voice recording to create a voiceprint and associate it with the speaker's identifier. OMNI Voice requires that the speaker whose voice is being enrolled into the system manually confirms the Enrolment. The confirmation is usually done with their mobile phone.

Use this Flow When

- Registering a new speaker for voice authentication

The flow starts with a call to POST `EnrollVoiceSample` (see also [Biometrics API Reference](#)).



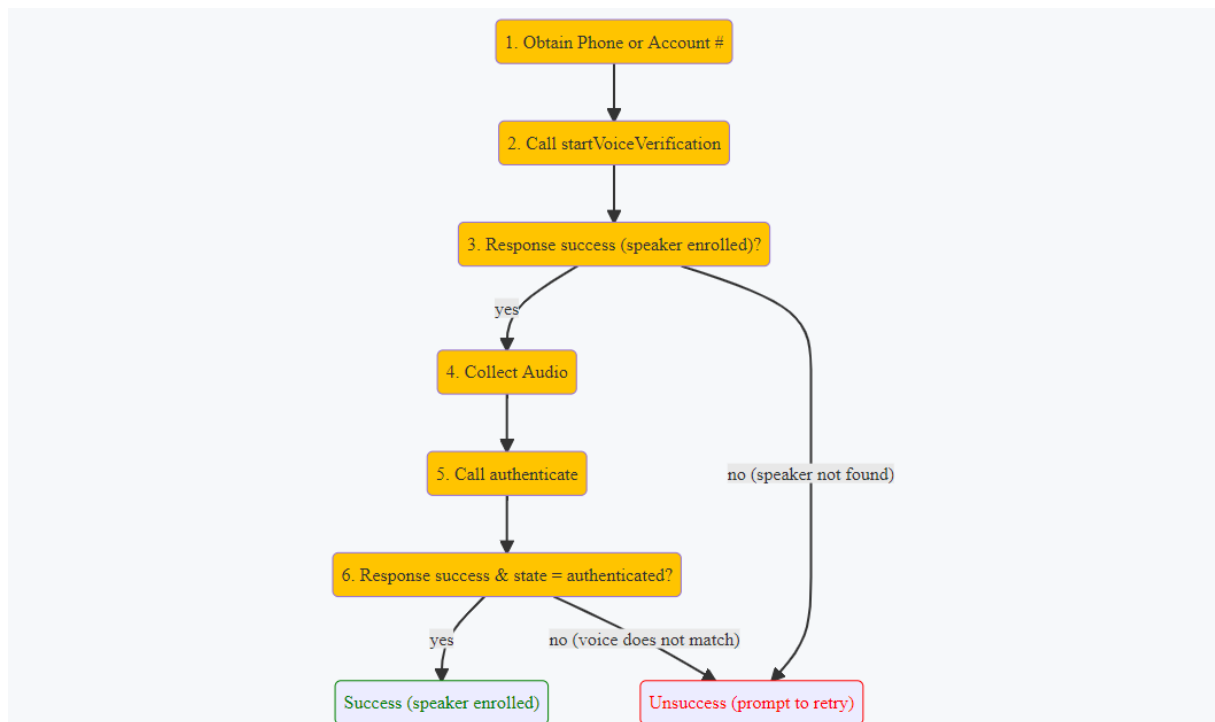
Voice Verification

The flow checks if the system has an enrolled voiceprint associated with a speaker identifier (e.g. account number) and, if so, sends an audio file containing speaker's voice and compares it with the voiceprint stored in the system.

Use this Flow When

- You need to check if a speaker is registered for voice authentication
- Only voice verification is needed
- You need an alternative modality for 2-nd factor authentication

The flow starts with a call to GET startVoiceVerification and ends with a call to POST authenticate (see also [Biometrics API Reference](#)).



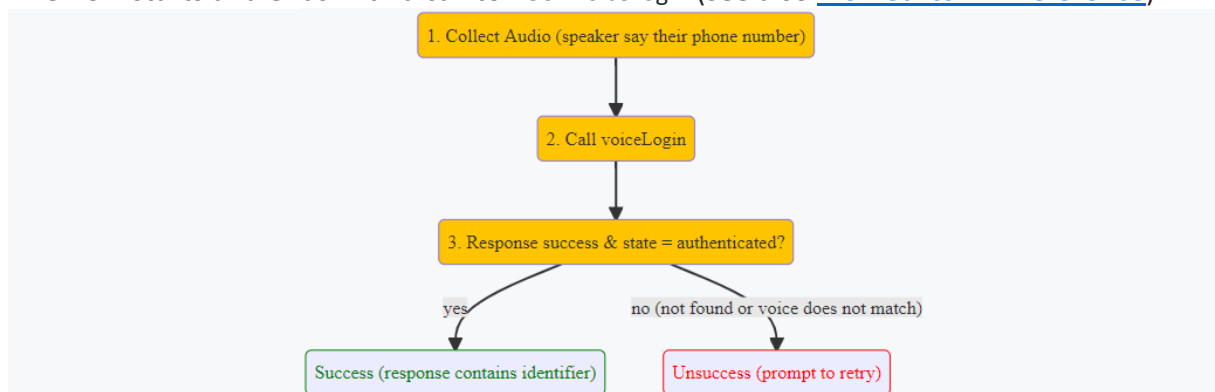
Voice Log-in Basic

The flow checks if the system has an enrolled voiceprint associated with a speaker identifier (identifier is obtained using speech recognition) and, if so, compares speaker's voice with the voiceprint stored in the system.

Use this Flow When

- You need to identify and authenticate a person with just 1 audio file (the audio must contain a spoken mobile phone number) (see also [Best Practices](#))
- You need an alternative modality for 2-nd factor authentication
- Suitable for efficient voice-based internal authentication scenarios (e.g. employee authentication)

The flow starts and ends with a call to POST voiceLogin (see also [Biometrics API Reference](#)).



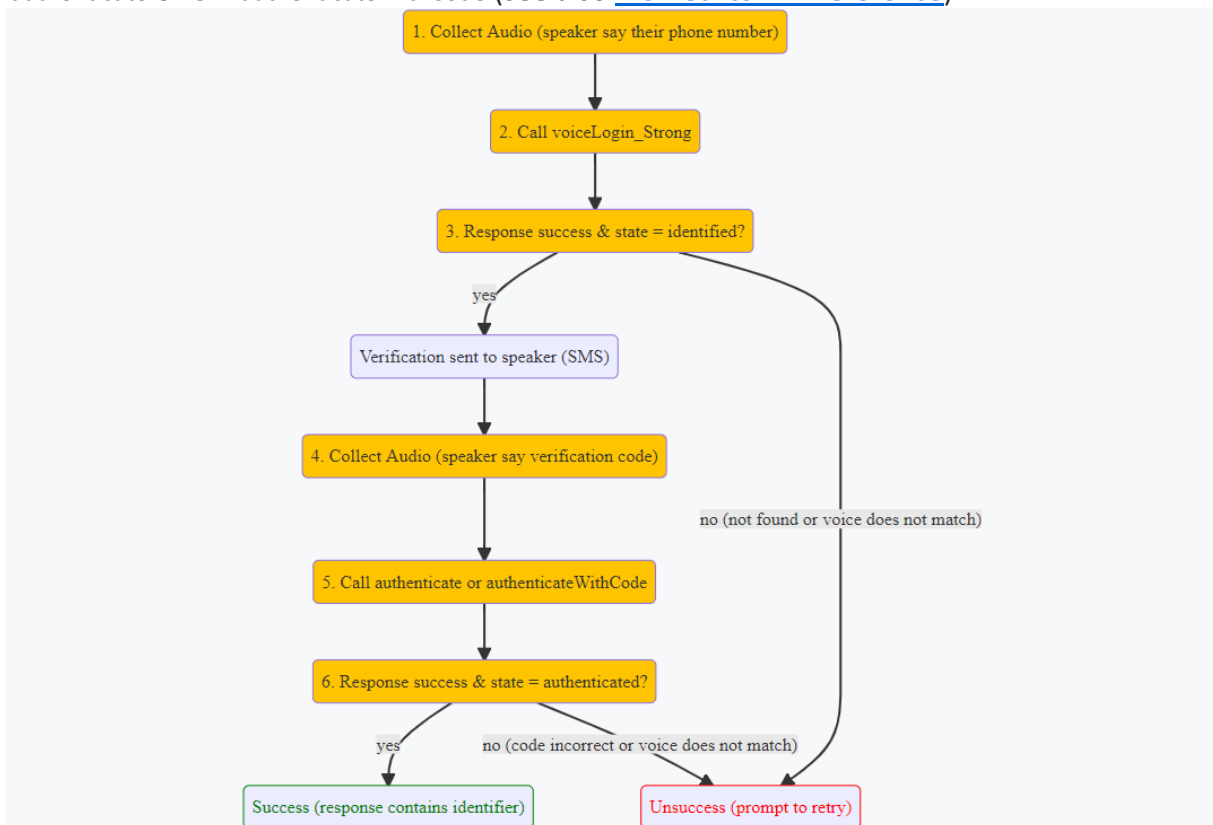
Voice Log-in Strong

This flow is similar to **Voice Log-in Basic** (where the speaker records themselves saying their phone number) but it also requires that the speaker submits a verification code sent via a cellular network. The verification code can be submitted as a voice recording in which case speech recognition **and** voice verification is performed on both, initial and verification audio submission.

Use this Flow When

- You would like to implement a voice-enabled log-in
- You require a secure identification and authentication mechanism
- Suitable for efficient voice-based external authentication scenarios (e.g. password resets)

The flow starts with a call to POST `voiceLogin_Strong` and ends with a call to either POST `authenticate` OR GET `authenticateWithCode` (see also [Biometrics API Reference](#)).



Biometrics API Reference

Base URL: <https://omnivoice.tech/biometrics/api>

Abstract

This document provides a detailed reference guide for the **OMNI Voice API**, including security, calling conventions, data structures, and the methods themselves.

There are two fundamentally different scenarios that are involved in the process of identification / authentication by voice:

- **Enrolment**,
- **Login / Authentication** ([click here](#) for flow diagrams)

The way we do the **Login / Authentication** is we generate a *voiceprint* based on the given audio that contains a speech sample. The *voiceprint* is then compared against each of the reference *voiceprints* stored in the org's database. The best match that passes a *confidence threshold* is then returned. Among other parameters documented below, this match contains:

- Comparison Score (a value between [0..1])
- Speaker's Identifier (a phone number) that corresponds to the best match

The **Enrolment** scenario is required to populate the reference database in the first place. This is done by calling the **Enrolment API**.

Note: ACL must be configured before accessing the API. Refer to [Security Model Reference](#) for more details

Header-Based API Parameters

Conventionally, most of **OMNI Voice Biometrics API** parameters are passed as **X-headers**. The table below lists all **X-header parameters** used in the API:

Parameter Name	Header Name	Description
spk_id_type	X-OMNI VOICE-	Contains only type portion of speaker ID; only account and phone types are supported

Parameter Name	Header Name	Description
	SPEAKER-ID-TYPE-ONLY	
spk_id	X-OMNI VOICE-SPEAKER-ID	Contains the full speaker ID which is pipe-separated; the type must be either account or phone (example: 'phone +1234567890')
meta	X-OMNI VOICE-META	An arbitrary metadata to be stored with the Enrolment; must be up to 1024 chars
channel	X-OMNI VOICE-CHANNEL	Up to 10 chars - provides a name of the channel on which the Enrolment is delivered. E.g. 'telephony' or 'web'
workflow_id	X-OMNI VOICE-WORKFLOW-ID	A GUID ID of the authentication workflow (used by multi-step scenarios, e.g. 2-Factor authentication)
lang	X-OMNI VOICE-LANG	A language-code used for scenarios involving speech recognition (full list)

Note: Language code convention follows <https://datatracker.ietf.org/doc/html/rfc2616#section-3.10>

Note: All phone number identifiers must follow the E.164 format (<https://www.itu.int/rec/T-REC-E.164/>) for Enrolment (e.g. '+1-xxx-yyy-yyy'). This is validated with an SMS message. Phone numbers that are not validated will not be available for voice authentication.

Audio Upload

Depending on which API method is called, an audio file may need to be passed along with the request. In this case, the file should be passed in the **FormData** under the key of **File**.

The hard limit on the file size is 2MB.

OMNI Voice expects that the passed in audio is packed into a PCM WAV file with the following format:

Spec Name	Value
wav format	PCM
sample type	integer
sample size	16 bits
sampling rate	16 KHz
channels	1 (mono)

Note: ! mp3 or any other compression algorithms are not currently supported !

Note: Other sampling rates will usually be automatically resampled. However, it is not officially supported and may be dropped in the future without notice. If your audio has a sampling rate that is not 16 KHz, it is advisable to resample to 16 KHz before attaching it to an API request.

Authorization Header Scheme

OMNI Voice API uses a custom authorization scheme for HTTP requests in which the **Authorization** header contains a single value prefixed with `apikey_`.

Example: `apikey_6Txgr9CvRPzwrcusJ+SczhqyU22KjDbsEGxYUflwYLk=`

Error Handling

OMNI Voice API distinguishes the following classes of errors:

Error Class	HTTP Response	Description
Application Layer Errors	200 OK	Errors that result from normal authentication flows. For example, an unsuccessful voice authentication result; such errors are passed along in a JSON content of a 200 OK response. These errors are conveyed in English only and can be displayed on the UI for the appropriate audiences "AS IS". See Data Structures below for information on how to handle these kinds of errors

Error Class	HTTP Response	Description
Argument Validation Errors	400 Bad Request	Errors that occur if any of the required X-headers are missing or contain invalid values
Throttling Errors	418 Teapot	Errors that occur if the request rates are too high. The allowed request rates are specified for each API individually - see API reference below
Authorization Errors	401 Unauthorized	Errors that result from invalid API keys provided in the Authorization header. <i>Note: API keys can be revoked / renewed using the OMNI Voice WEB interface</i>
Application Layer Error	500 Server Error	Any internally unhandled errors that might occur at OMNI Voice backend. <i>Note: hopefully, you won't have to deal with any of those :-)</i> not too frequently :-)

Data Structures

We suggest that any **4xx** and **5xx** errors are considered as *exceptions* and should be handled as such.

All **200 OK** responses comply with a schema that can be described with the following typescript hierarchy:

```
1interface IOMNI VoiceApiResponse {
2  success: boolean; // if false, the API call is considered unsuccessful
3  error: string; // conveys the error message if the value of success is false
4};

1interface IEnrollVoiceSampleResult extends IOMNI VoiceApiResponse {
2  metadata?: string; // contains the metadata passed along with the request
3  netSpeechDuration_sec?: number; // the duration of net speech (in seconds) after audio pre-processing at the backend
4  identifier?: string; // the Enrolled speaker's identifier (e.g. phone number); phone number identifiers are stored in **E.164** format
5};
```

Note: When Enrolling, OMNI Voice API will automatically issue a verification SMS to new phone numbers containing a validation link

```
1interface IOMNI VoiceWorkflow extends IOMNI VoiceApiResponse {
2  id?: string; // the internal identifier of the authentication workflow
3  timestamp?: string; // the timestamp of when the workflow was created (in ISO 8601 format)
4  workflowType?: string; // see table below
5  state?: string; // see table below
6  metadata?: string; // contains the metadata passed along with the request
7  expired?: boolean; // if present and false, indicates that the workflow is expired (no 2-factor verification was provided)
8  identifier?: string; // if present and set, corresponds to the identifier of the found speaker (e.g. phone number)
9};
```

Note: The identifier field of the IOMNI VoiceWorkflow structure, when present, will always contain phone numbers in E.164 format (e.g. "+1-123-456-789").

Workflow Types

Type	Description
authentication	Indicates that the workflow represents a voice verification scenario (see flows)
login-weak	Indicates that the workflow represents a basic voice login scenario (see flows)
login-strong	Indicates that the workflow represents a 2-Factor voice login scenario (see flows)

Workflow States

State	Description
authenticated	The speaker has been successfully identified and authenticated
identified	The speaker has been identified but not authenticated
unenrolled	The speaker is unenrolled
unauthenticated	The speaker has been identified but failed the authentication

API Reference

Language code convention follows <https://datatracker.ietf.org/doc/html/rfc2616#section-3.10>

All phone number identifiers must follow the E.164 format (<https://www.itu.int/rec/T-REC-E.164/>) for Enrolment (e.g. '+1-xxx-yyy-yyy'). This is validated with an SMS message. Phone numbers that are not validated will not be available for voice authentication.

Audio files passed along with a request should be passed in the **FormData** under the key of **File**. Max. file size is 2MB.

POST enrollVoiceSample

Summary:

Adds a reference voiceprint to the database and associates it with a speaker identified by `spk_id` (which is a pair: {identifier, identifier type}). If the identifier type is **phone**, **OMNI Voice** will attempt to send a text message to the phone number with a verification link. There may be multiple identifiers associated with the same person. The default limit is 5 voiceprints per speaker. If an attempt to Enrol more than 5 voiceprints is made, the most similar voiceprints are replaced to keep the total number of voiceprints at 5.

Result:

The result, if successful, means that a confirmation link was sent to the mobile phone of the speaker. A person who receives this link will need to click on it within 2 minutes after which the identifier will be considered validated. Failing to confirm the link within the expiration window will result in the deletion of the expired Enrolment.

NOTE: Voice verification test requires a length of speech. The longer the provided speech sample is, the higher the chance to pass the voice verification test; please review our [voice sample guidelines](#) to ensure high success rates of your voice-authentication integration

Spec Name	Value
API URI	EnrolVoiceSample
Method	POST
Parameters	spk_id, channel, meta, lang, audio
Result	IEnrolVoiceSampleResult

GET startVoiceVerification

Summary:

Initiates a 2-step process called **Voice Verification** which involves:

1. Speaker identification
2. Authentication (done by calling `authenticate` - see below)

`startVoiceVerification` requires that `spk_id` (which is a pair: {identifier, identifier type}) be provided. The speaker search for identification will be done using the exact match on the passed in `spk_id`.

Voice verification is useful in situations where speaker's identity is a known reliable quality.

Result:

A successful result will contain a workflow record (see IOMNI VoiceWorkflow) with state set to identified. Any other result is considered an error.

Spec Name	Value
API URI	startVoiceVerification
Method	GET
Parameters	spk_id, channel, meta
Result	IOMNI VoiceWorkflow

POST voiceLogin

Summary:

Invokes the basic voice-based login flow which entails:

1. Transcribing a recording of a person saying their phone number or account number (see list of [supported language](#))
2. Generating a voiceprint from the same audio
3. Invoking a look-up query against the database of the Enrolled references to find the speaker's identity

This flow is the most user-friendly one. Indeed, all the user would need to do is record themselves saying their identifier. This option is a good choice for the less secure scenarios that do not require multi-factor authentication.

Result:

A successful result will contain a workflow record (see IOMNI VoiceWorkflow) with state set to authenticated. Any other result is considered an error. That being said, if the returned workflow's state is identified, it means that while the speaker's identity was successfully established (from the speech-recognised identifier), the voice verification test has failed.

NOTE: Voice verification test requires a length of speech. The longer the provided speech sample is, the higher the chance to pass the voice verification test; please review our [voice sample guidelines](#) to ensure high success rates of your voice-authentication integration

Spec Name	Value
API URI	voiceLogin
Method	POST

Spec Name	Value
Parameters	spk_id_type, channel, meta, lang, audio
Result	IOMNI VoiceWorkflow

POST voiceLogin_Strong

Summary:

NOTE: voiceLogin_Strong will not work for users who do not have at least one phone number associated with them.

Initiates the most robust voice-based login flow which entails:

1. Transcribing a recording of a person saying their phone number or account number (see list of [supported language](#))
2. Generating a voiceprint from the same audio
3. Invoking a look-up query against the database of the Enroled references to find the speaker's identity
4. Sending a verification code to the found speaker's phone number
5. Validating the verification code with either audio or direct submission

When a call to voiceLogin_Strong is dispatched, the **OMNI Voice** backend will run steps 1 to 4. Step 5 can then be submitted with another call to either authenticate (passing another audio recording of the user speaking the verification code they receive) or authenticateWithCode.

Result:

A successful result will contain a workflow record (see IOMNI VoiceWorkflow) with state set to identified. Any other result is considered an error.

NOTE: Voice verification test requires a length of speech. The longer the provided speech sample is, the higher the chance of passing the voice verification test; please review our [voice sample guidelines](#) to ensure high success rates of your voice-authentication integration

Spec Name	Value
API URI	voiceLogin_Strong
Method	POST
Parameters	spk_id_type, channel, meta, lang, audio
Result	IOMNI VoiceWorkflow

POST authenticate

Summary:

There are 2 authentication scenarios that this method concludes; depending on the workflow type & state:

1. workflowType: **authentication**, state: **identified**:
 - the wav must contain a speech sample with any content
 - just voiceprint matching is done
2. workflowType: **login-strong**, state: **identified**:
 - the wav must be a speaker's recording of the verification code
 - speech recognition, SMS code comparison and voiceprint matching are done

Result:

A successful result will contain a workflow record (see IOMNI VoiceWorkflow) with state set to authenticated. Any other result is considered an error.

NOTE: Voice verification test requires a length of speech. The longer the provided speech sample is, the higher the chance of passing the voice verification test; please review our [voice sample guidelines](#) to ensure high success rates of your voice-authentication integration

Spec Name	Value
API URI	authenticate
Method	POST
Parameters	workflow_id, lang, audio
Result	IOMNI VoiceWorkflow

GET authenticateWithCode

Summary:

Concludes an authentication workflow with type **login-strong** and state **identified**.

Note: it is recommended to prefer POST authenticate OVER GET authenticateWithCode since the former will append the audio containing a verification code to the audio obtained at POST voiceLogin_Strong for voiceprinting thus improving success rates of the voiceprint test.

Result:

A successful result will contain a workflow record (see IOMNI VoiceWorkflow) with state set to authenticated. Any other result is considered an error.

Spec Name	Value
API URI	authenticateWithCode/{verification_code}
Method	GET
Parameters	Route-based parameter verification_code
Result	IOMNI VoiceWorkflow

Security Model Reference

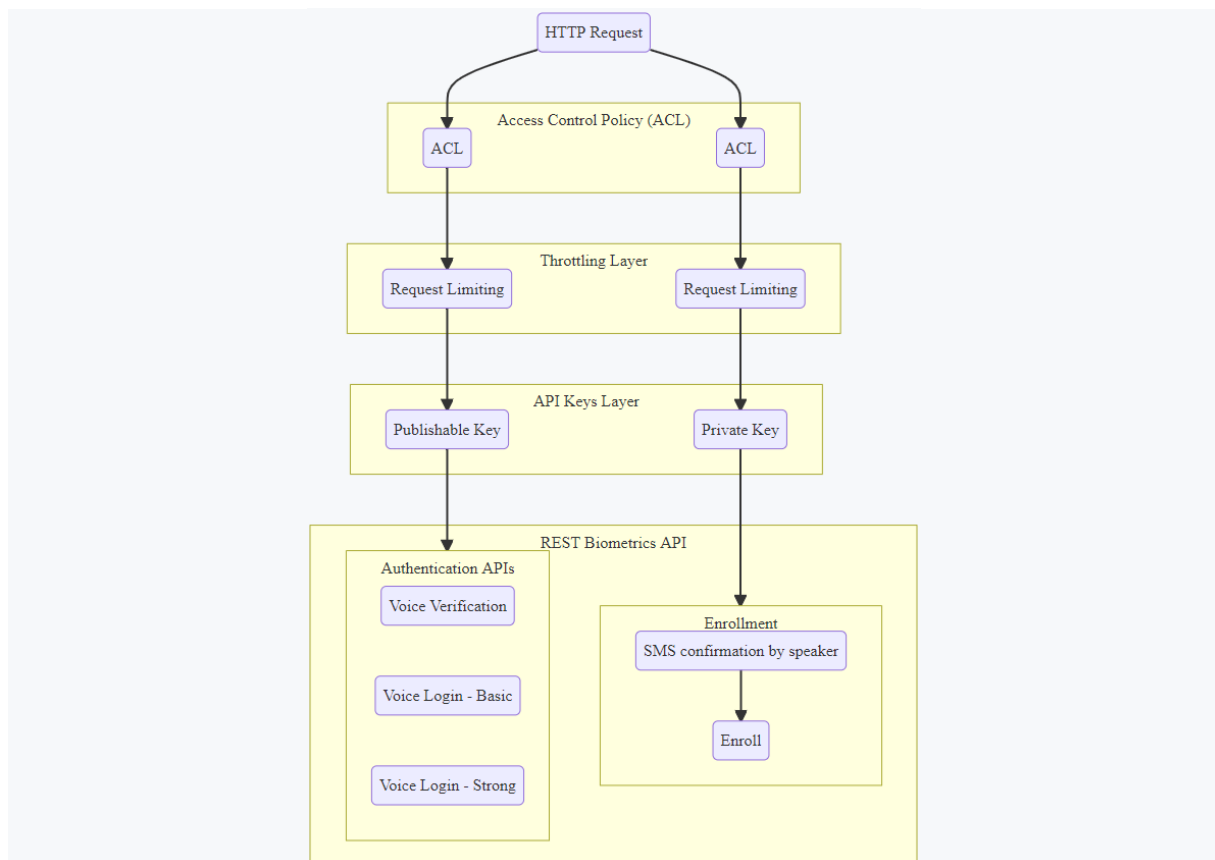
Abstract

This data sheet summarizes most frequently asked questions around OMNI Voice security model. The data sheet covers the following topics:

1. Quick security reference to get started with the API. The rest of the document can be skipped if this is all that's being sought
2. General overview of security policies implemented in OMNI Voice

Accessing OMNI Voice API

OMNI Voice exposes [REST Biometrics API](#) that is used to enable Voice Biometrics on any cloud-enabled application. Measures taken to protect biometrics data managed by OMNI Voice can be viewed as stacked layers where each layer solves a specific security issue.



The above diagram shows the stages that each API request has to go through before it actually reaches the API Layer. There are 2 types of API keys that are available to access the API:

- Publishable API Key - enables access to any **authentication** scenarios
- Private API Key - enables access to just **Enrolment**

See also: [OMNI Voice Flows](#)

ACL and **Throttling** are configured for each API key. The default Throttling values are limited to 10/sec which can be modified on a case-by-case basis by the OMNI Voice support team. The default ACL's are empty which means that OMNI Voice will respond with a **403 Forbidden** to any requests unless the **IP Address** of the endpoint making a request is added to the ACL of the corresponding API Key.

API Keys configuration can be accessed via <https://omnivoice.tech/biometrics/manage-API-keys> (you have to be logged in as an Org Administrator and have a mobile phone number associated with your account to update the ACL's).

Data & API Classification

OMNI Voice renders its services on the data managed by the system on behalf of its customers. The data is carefully segregated based on its nature and the API context it is used in.

OMNI Voice is comprised of a number of *vertical* services (apps) that are completely decoupled from each other. The decoupling is done on all levels, including:

- Backend (APIs) and frontend functionalities
- Databases and data models
- Authentication
- Configuration

Note: the paper focuses on the Biometrics app only; it will cover the security aspects of other OMNI Voice components only to a degree that is required to understand the Biometrics Security Model

Apps are managed and accessed from a central *management* point that is referred to as the *OMNI Voice Platform*. Very little data is actually shared between the apps.

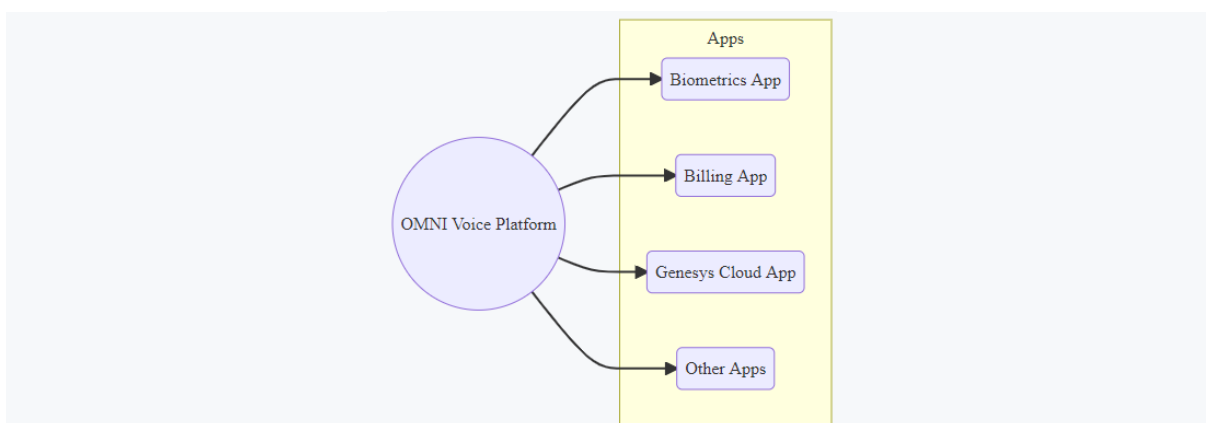


Table 1: Data Classes across all OMNI Voice apps and the platform:

Data Class Name	Description
Customer Data	Voiceprint metadata, Authentication records, external system references, speakers (including identifiers), authentication workflow records, audio files
Sensitive Customer Data	Mainly Voiceprints (Enrolments) and their linkage with identifiers
Sensitive System Data	Platform login credentials, API Keys, JWT and OAuth Tokens, ACLs
System Data	Any data that is not Customer, Customer Sensitive, or System Sensitive Data

Table 2: API classes (*surfaces*) across all OMNI Voice apps and the platform:

API Class Name	Description
Anonymous APIs	APIs that can be executed by anyone from any location (only protected with request limiting policies)
Public APIs	Backend APIs whose policy allows their execution with API keys
Frontend APIs	Backend APIs whose policy allows their execution on OAuth or JWT token regardless of the API key policy
System APIs	APIs for communication between the backends

API Class Name	Description
Sensitive APIs	APIs which either deal with Customer / System Sensitive Data or allow Create / Update access to Customer Data, excl. System APIs
Management APIs	APIs which deal with System Data

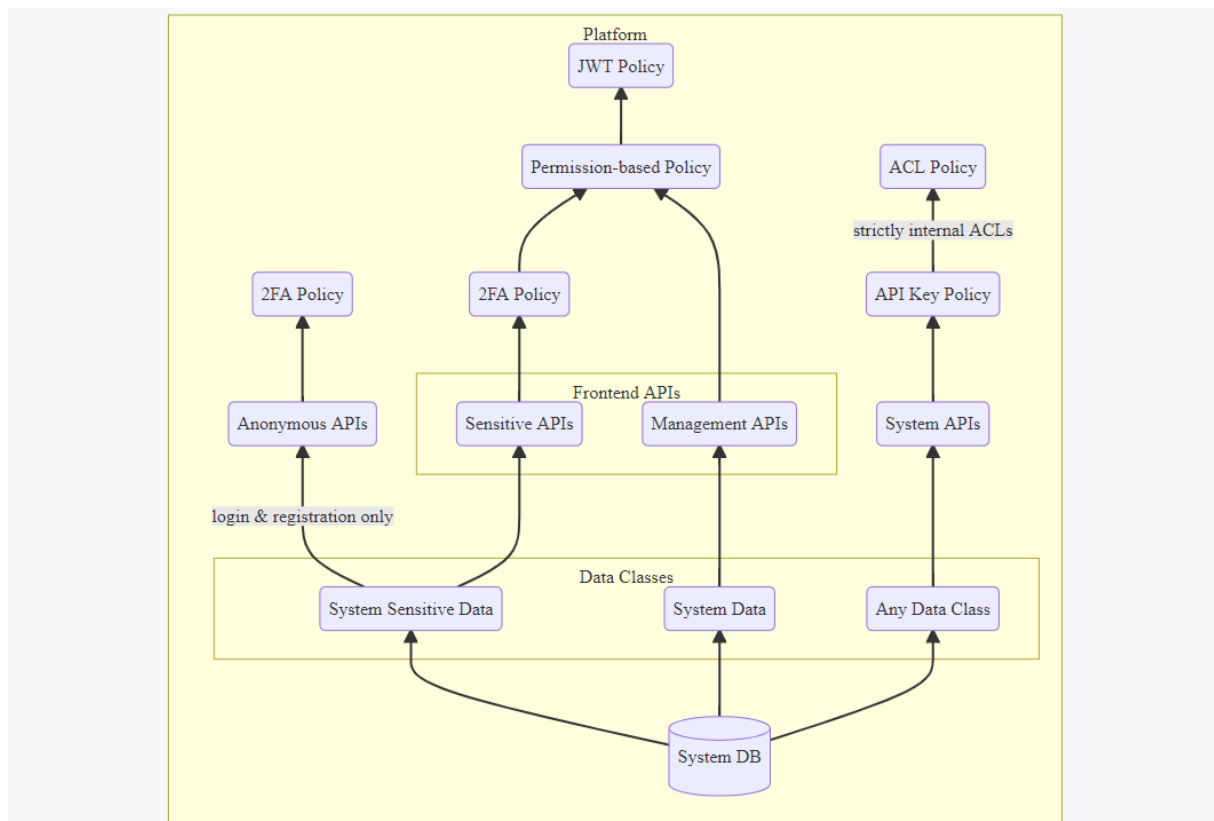
API surface closely follows data classification and is segregated based on the data class it handles. In practical terms, API segregation means that each of the API surfaces given in **Table 2** are protected with either **ACL Policy**, **2-Factor Authentication**, or both, regardless of the authorization scheme.

Security Architecture

Platform

The platform does not expose any *Public APIs*. It implements a permission-based security model and issues expirable JSON WEB Tokens (JWT) to logged-in users that are handled by the Front-End run in browsers. The *Frontend APIs* that deal with *Sensitive Data* are additionally protected using a traditional 2-Factor-Authentication (2FA) scheme. Each API has an *orthogonal* permission associated with it and its access is fully determined whether the currently logged in user has the respective permission in its role.

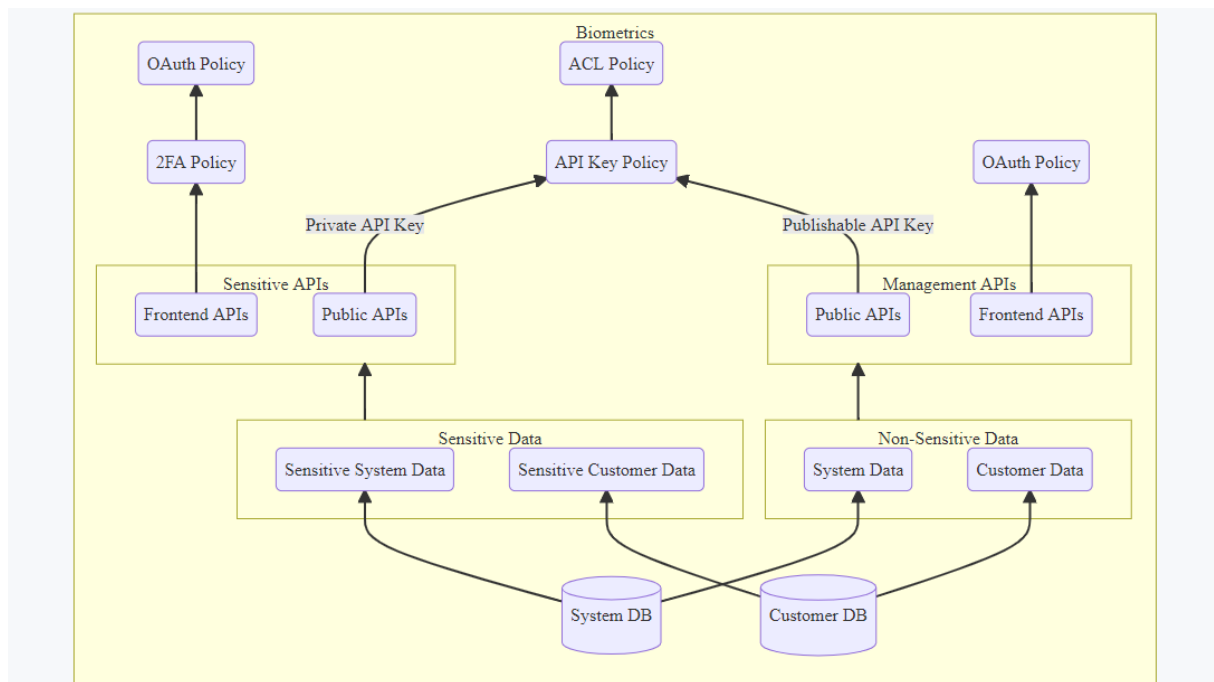
The platform uses a dedicated data store that is physically and logically isolated from the publicly available network.



Biometrics

Unlike the Platform, access to any Biometrics API is determined by the type of API Key. This is true for both, Frontend and Backend APIs. Additionally, Frontend APIs require an OAuth token issued by the *Biometrics OAuth Service* when a redirect from the Platform is made. The OAuth tokens are always bound to an API Key, thus, when a *Business User* makes a Frontend API call, the permission is determined by the type of API Key the OAuth token is bound to.

The data stores utilized by Biometrics are physically and logically isolated from the publicly available network. Furthermore, *Customer Data* is implemented as a separate data store and can be hosted by a customer (enterprise customers only).



Data In-Transit Policy

In summary, the policy ensures that data in-transit is always encrypted (the current standard of TLS) and never contains *Customer Sensitive Data*.

High sensitivity of assets handled by OMNI Voice warrants an explicit policy on which data is actually returned by its APIs. The policy dictates that no APIs (public or not) handle Customer Sensitive Data explicitly. In other words, *Voiceprints* are never submitted, nor retrieved as a direct API payload.

Furthermore, *Voiceprints* emitted as a result of Enrolment are retained by OMNI Voice only upon a confirmation from the speaker that they belong to. The modality on which the confirmations are delivered is usually a cellular network (i.e. SMS confirmations).

Genesys Integration

Abstract

This section provides an overview of the data communication channels and flows between the OMNI Voice Platform and Genesys Cloud.

Two distinct approaches can be utilized to connect your call center to the OMNI Voice Platform:

1. OMNI Voice Media Server (Cloud)
2. Telephony Appliance (Cloud or OnPrem)

Detailed architecture and technical diagrams illustrating the data flows used to establish and manage communication channels are outlined below.

OMNI Voice Media Server

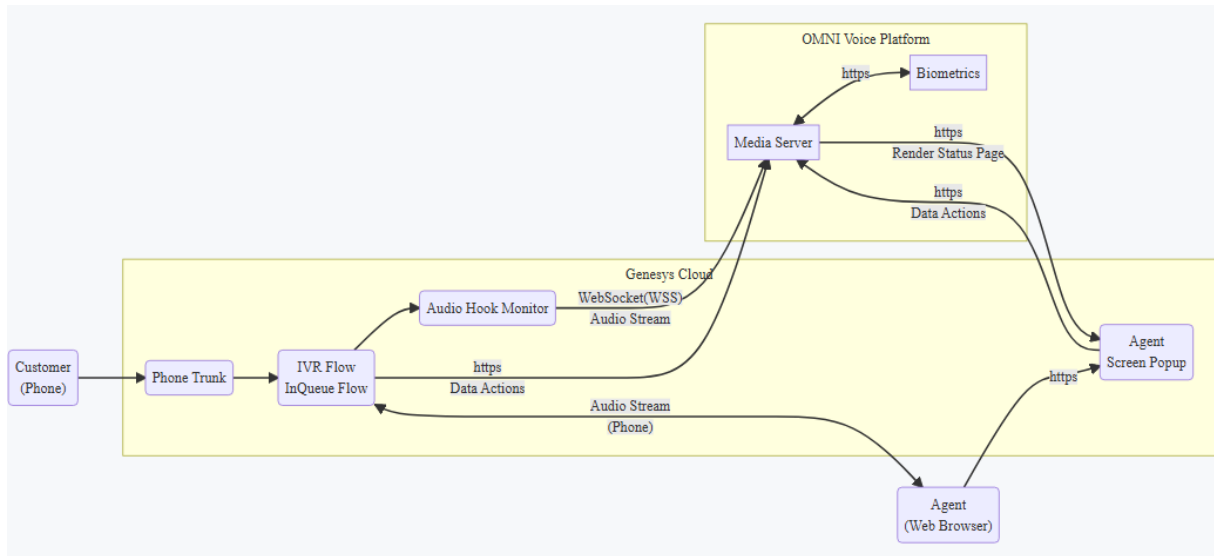
OMNI Voice Media Server has been introduced to support passive authentication scenarios and fully cloud voice authentication deployments.

AudioHook Integration

The Genesys AudioHook Monitor Integration facilitates the streaming of real-time conversational audio and metadata to the OMNI Voice platform. This integration empowers partners and customers to enhance the Genesys Cloud platform with features such as Biometrics, Voice Identification, and Authentication, delivered as a cloud service.

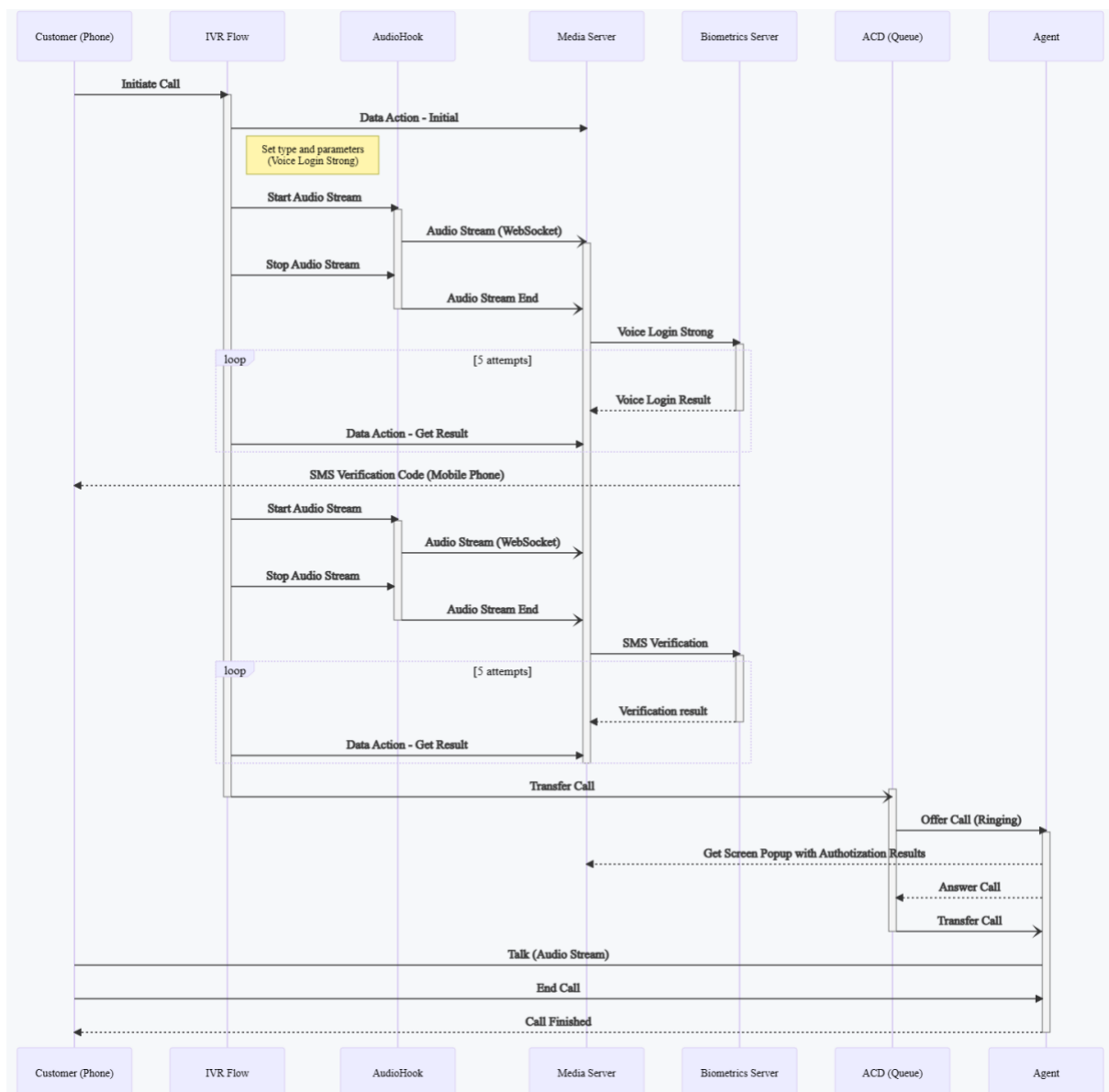
OMNI Voice is fully integrated with Genesys Cloud through the use of Data Actions that can be placed in Architect Flows as required.

Data Actions designed by OMNI Voice are imported during the automated installation procedure (see **Genesys Cloud Installation** section for details). When invoked by a Genesys Flow, they call the appropriate functions of the public REST API.



All audio is data transferred using secure Web Socket Protocol (WSS). The AudioHook Monitoring stream is automatically stopped as soon as the Media Server obtains enough audio to finish the requested action.

The following diagram illustrates all events involved in a 2FA authentication scenario (**Voice Log-in Strong**). Other supported flows would follow a similar or simpler signalling and so are omitted for brevity.



OMNI Voice Data Actions:

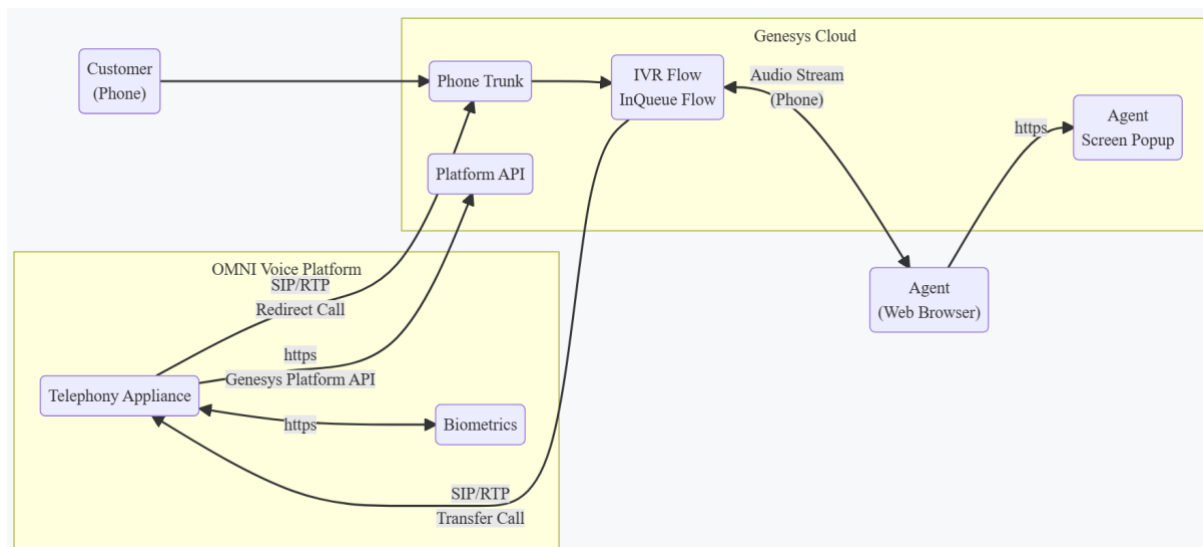
1. **OMNI Voice Initial:** initializes a desired stream processing mode and configures parameters.
2. **OMNI Voice Verify:** Initiates the Biometrics Voice verification scenario
3. **OMNI Voice StartAudio:** Starts a paused Genesys AudioHook stream.
4. **OMNI Voice GetResult:** Obtains the data processing results

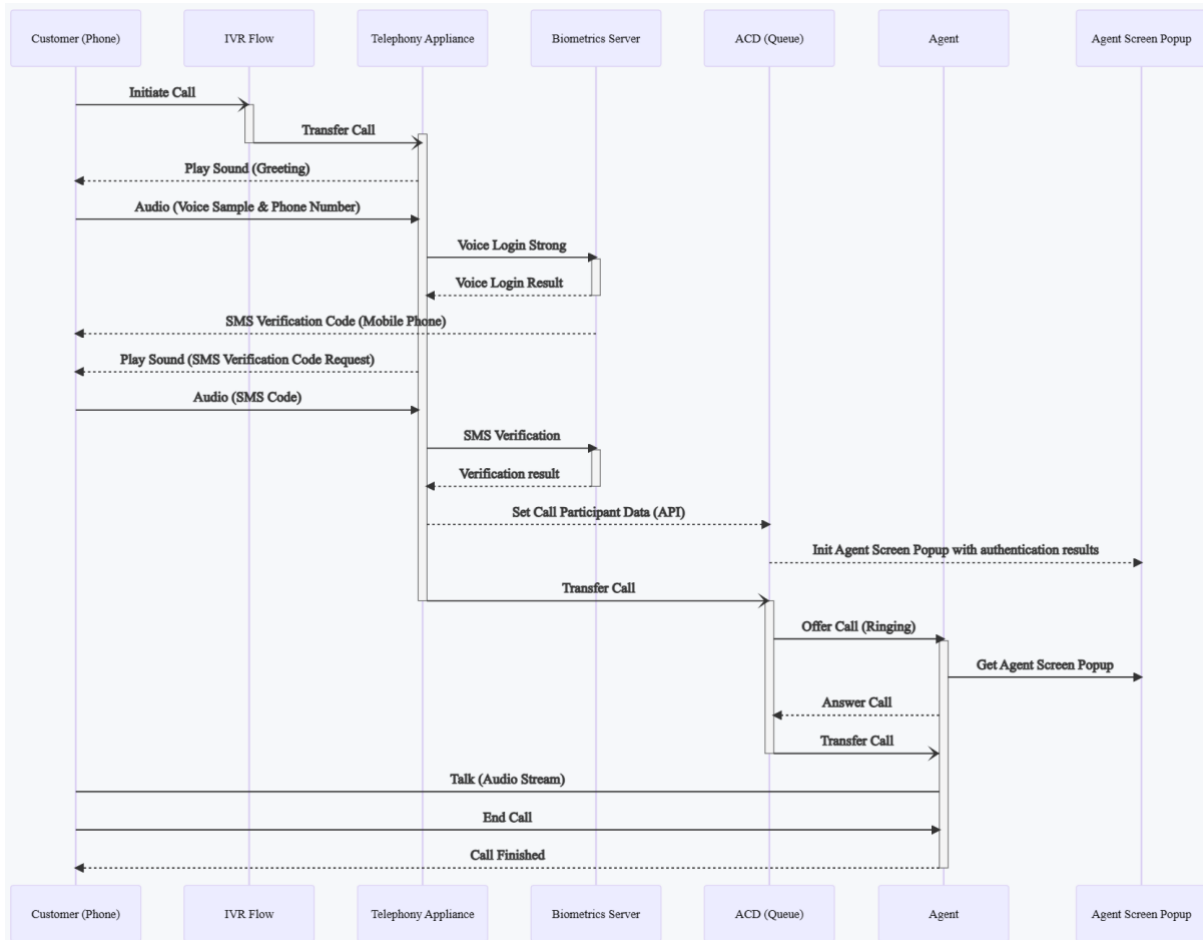
Telephony Appliance (Cloud)

Telephony Appliance is a component introduced to support Genesys Cloud versions for which AudioHook Integration is not available. As such, it requires that calls are transferred to it for authentication. Passive authentication scenarios, under this approach, are not supported.

This approach closely resembles the OMNI Voice Media Server setup but comes with several additional requirements and limitations, including:

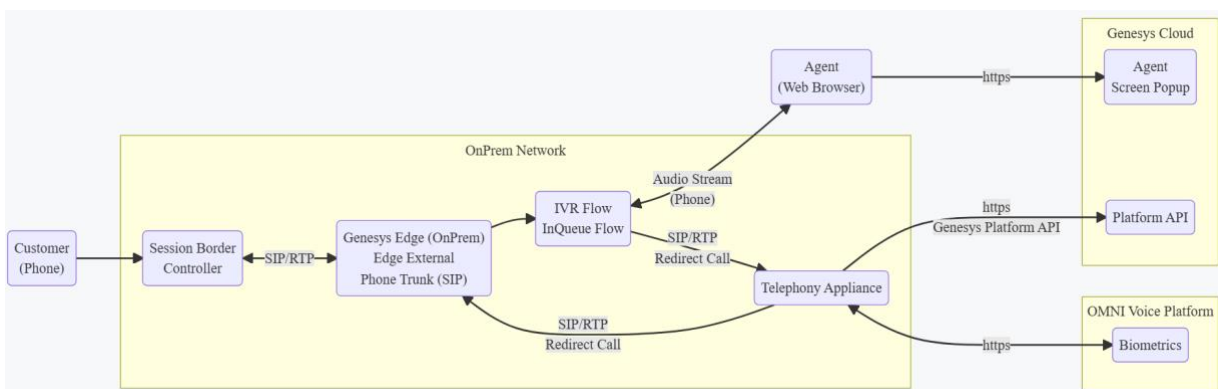
1. Configuration of a Dedicated External Phone Trunk (contact [support](#) for details).
2. Unavailability of passive mode.
3. Necessity to transfer calls to the Telephony Appliance extension for identification or authentication purposes.
4. Authentication Data transferred via the Genesys Conversation Participant Data.
5. Unavailability of Interactive Enrolment process.





Telephony Appliance (OnPrem+BYOCC)

The OMNI Voice Telephony Appliance installer can be obtained from OMNI Voice upon request. The appliance must be installed on a Windows machine. Further details on the configuration of the Telephony Appliance are available upon request.







Genesys Cloud Installation

Prerequisites

To run the OMNI Voice Media Wizard, please prepare the admin account with the following roles assigned:

1. "Master Admin" role (for importing Integrations and Architect flows)
2. "Script Designer" role

Name	Type	Description	Divisions	Group Inheritance	Unused ⓘ	Assigned
Master Admin	Standard	PureCloud Administrator	Home 			
Script Designer	Standard	Enables a user to create scripts, manage existing scripts, make scripts available for use by publishing them, and manage published scripts	Home 			

Phone Trunk Configuration

By default, Recording is disabled, which means that calls will not be recorded.

To be able to record inbound and outbound voice calls and process the audio data with AudioHook/Transcribe/OMNI Voice, you must enable and configure recording on the trunk that is hosting the calls. Once you enable recording, all calls are automatically recorded. You can use recording policies to disable recording and to configure when to save or delete recordings, but you can only enable recording at the trunk level.

1. Click **Admin**.
2. Under **Telephony**, click **Trunks**.
3. Click the **External Trunks** tab.
4. From the list, select the trunk you want to configure.
5. Under **External Trunk Configuration**, click **Media**.
6. Under **Recording**, select **Record calls on this trunk**.
7. If you must have user consent, select **Require user consent before recording**.

8. To continue recording on an external transfer, clear the **External bridged transfers** check box.

Note: By default, the **Consults**, **Holds** and **External bridged transfer** check boxes are not selected. Consider if you need to enable it depending on your infrastructure.

9. Under **Audio Format**, select the **PCMU** audio codec.
10. To record both sides of the conversation separately, select **Dual channel**.
11. Click **Save External Trunk**.

DTMF Settings

DTMF Payload

Indicate the type of payload to be used when the DTMF type is specified as RTP Events. The default value is 101. Valid range is 96-127.

DTMF Method

RTP Events

The method used to transmit dual-tone multifrequency (DTMF) signaling. Set to RTP Events for RFC 2833 or 4733 compliancy.

Recording

☒ Record calls on this trunk
 ☐ Require user consent before recording

Optional Recording

☐ Consults
Record the private conversation between the agent and supervisor.

☐ Holds
Continue recording while the call is on hold.

☐ External bridged transfers
Continue recording external-to-external transferred calls (i.e. externally bridge calls with no active participants within Genesys Cloud).

Recording Audio

☐ Level the volume on both sides of the conversation

Audio Format

For recording transcription the format must be PCMU, PCMA, L16, or Opus, and recorded with dual channels.

Codec

PCMU (G.711 μ -law)

☒ Dual channel
Saves both channels in a separate stream. Only available when using PCMU, PCMA, L16, or Opus.

Recording Beep Tone

☐ Play periodically while recording

Number of Tones

☒ Single
 ☐ Dual

Customize Tones

Play Beep Tone Every

14 seconds

Example of the External Phone Trunk Media section settings

AudioHook Integration

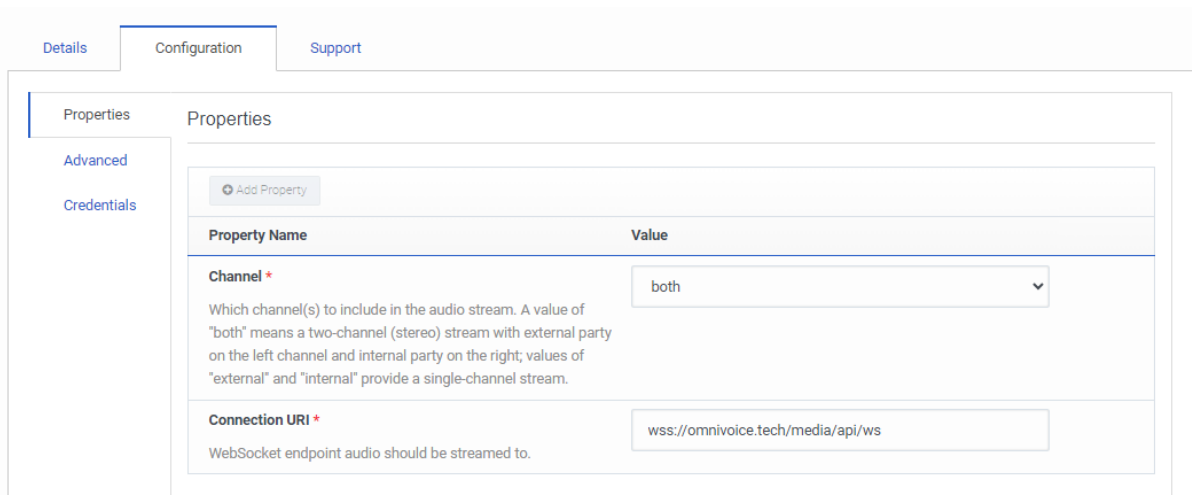
To install AudioHook Monitor in your org:

1. Log in to Genesys Cloud for the org in which to add AudioHook Monitor.
2. Click **Admin**.
3. Under **Integrations**, click **Integrations**.
4. Click **Integrations**.
5. In the **Search** box, type **AudioHook**. The card for AudioHook appears.
6. Click the **AudioHook** card. Information about the app appears.
7. Click **Purchase**.

AudioHook Monitor is now available to configure and activate from Genesys Cloud's Admin - > Integrations page.

To configure it click on the AudioHook integration.

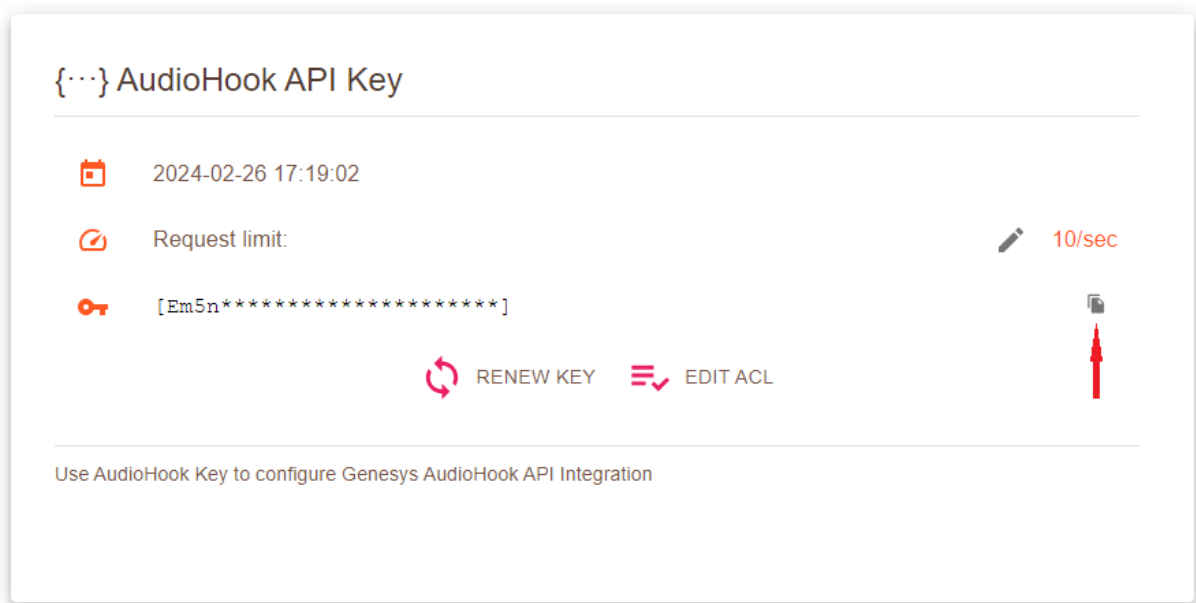
1. Set **Name** on the **Details** tab.
2. Click on the **Configuration** tab.
3. Set **Connection URI** to "wss://OMNI Voice.tech/media/api/ws".
4. Set **Channel** to "both".



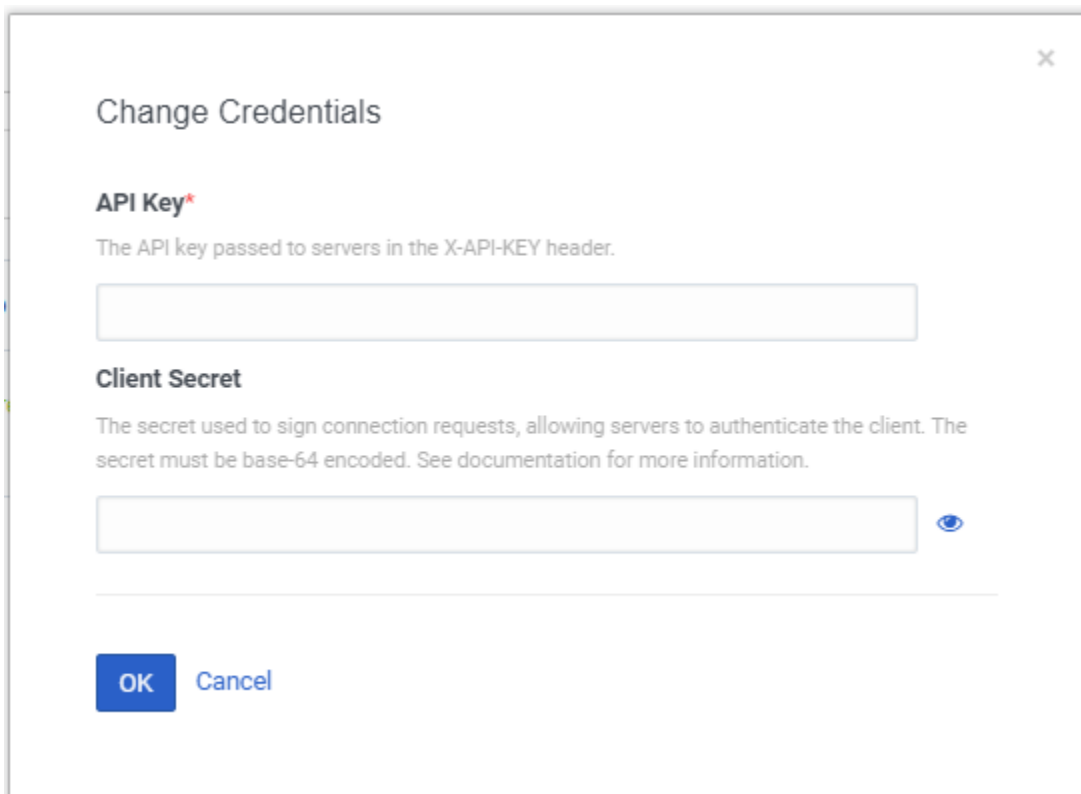
The screenshot shows the 'Configuration' tab for the AudioHook integration. On the left, there are tabs for 'Details', 'Configuration', and 'Support'. Under 'Configuration', there are sub-tabs for 'Properties', 'Advanced', and 'Credentials'. The 'Properties' sub-tab is active, displaying a table with two properties:

Property Name	Value
Channel * Which channel(s) to include in the audio stream. A value of "both" means a two-channel (stereo) stream with external party on the left channel and internal party on the right; values of "external" and "internal" provide a single-channel stream.	both
Connection URI * WebSocket endpoint audio should be streamed to.	wss://omnivoice.tech/media/api/ws

5. Login to OMNI Voice and open **Media Service** page.
6. Click on **Copy Key** button as show on the screenshot below.



7. Return to AudioHook configuration and click on the **Credentials** Section.
8. Set Credentials as on the screenshot. You should paste **API Key** from step 6. here. Leave **Client Secret** blank.



Change Credentials

API Key*

The API key passed to servers in the X-API-KEY header.




Client Secret

The secret used to sign connection requests, allowing servers to authenticate the client. The secret must be base-64 encoded. See documentation for more information.

OK Cancel

9. Save AudioHook Configuration.
10. Click on **Activate** toggle button to activate the integration. The Change Status dialog appears and asks you to confirm that you want to activate AudioHook Monitor.

11. Click **Yes**.

	Name	Category	Status	Last Updated	
	AudioHook	AudioHook	 Active	02/12/2024	

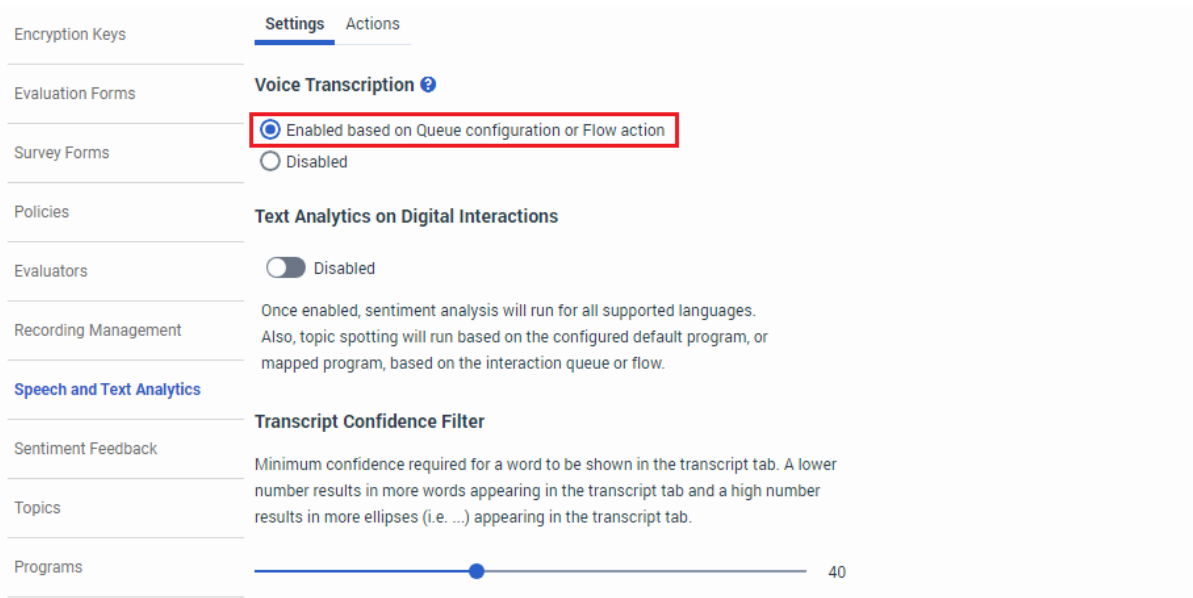
12. After a few seconds, the installation initiates a connection check to the server. If the connection check succeeds, the status changes to **Active**. If the check fails, an error appears.

13. If it gives an error, please check the configuration (steps 2-10).

Transcribe Feature

AudioHook Monitor requires enabled **Transcribe** feature. You need to enable and configure feature in order to use AudioHook:

1. Click **Admin**.
2. Under **Quality**, click **Speech and Text Analytics**.
3. Select the **Settings** tab.
4. Under **Voice Transcription**, select **Enabled based on Queue configuration or Flow action**.



Encryption Keys

Evaluation Forms

Survey Forms

Policies

Evaluators

Recording Management

Speech and Text Analytics

Sentiment Feedback

Topics

Programs

Settings Actions

Voice Transcription ?

☒ Enabled based on Queue configuration or Flow action

☐ Disabled

Text Analytics on Digital Interactions

☐ Disabled

Once enabled, sentiment analysis will run for all supported languages. Also, topic spotting will run based on the configured default program, or mapped program, based on the interaction queue or flow.

Transcript Confidence Filter

Minimum confidence required for a word to be shown in the transcript tab. A lower number results in more words appearing in the transcript tab and a high number results in more ellipses (i.e. ...) appearing in the transcript tab.

40

5. Under **Low Latency Transcription**, select **Enabled** if you want to minimize the latency of all transcripts sent through the Notification API.
6. Click **Save**.

Enable Transcribe/AudioHook on desired queues

Permission Prerequisites

- Routing > Queue > Add, Edit, Delete, Join, and View
- Routing > Queue > Readonly
- Routing > Queue Member > Manage
- Architect > UI > View

The following permissions are required to edit or view prompts in Architect (for whisper audio):

- Architect > UserPrompt > View
- Architect > UserPrompt > Edit

The following permission is required for agents to request for after call work when the setting is Agent Requested:

- Conversations > Settings > View

Configuration steps:

1. Click **Admin**.
2. Under **Contact Center**, click **Queues**. The Manage Queues page opens.
3. Click on the desired Queue.
4. Go to the **Voice** tab and enable transcription like on the screenshot below.

General
Routing
Members
Wrap-up Codes
Voice
Chat
Message
Email
Callback

Service Level

80%

20

Service Level Target (Seconds)

20

Calling Party Name

Calling Party Number

remember to input the country code

Alerting Timeout (Seconds)

8

In-queue Flow

Search for flow by name

Default Script

Search for script by name

Whisper Audio

☐ Only play whisper audio if agent is configured for auto-answer

☒ Play whisper audio for all agents

Continue Voice Recording during Queue Wait

On

Voice Transcription

On

Whisper Prompt

Search for whisper prompt

Hold Audio

If set, override org level default audio for agent-initiated hold from this queue

Search for on hold audio

Import OMNI Voice Data Actions and Architect Flow modules

1. Click **Admin**.
2. Under **Integrations** click **OAuth**.
3. Click **Add Client**.
4. Set **AppName** (ex. **OMNI VoiceWizard**).
5. Set **Grant Types** to **Token Implicit Grant (Browser)**.
6. Set **Authorized redirect URIs (one per line)** to: <https://omnivoice.tech/media/wizard>
7. Set **Scope** to *analytics, architect, authorization, presence, routing, users*

Client Details

App Name

OmniVoiceWizard

Description

Token Duration (seconds): the number of seconds, between 5mins and 48hrs, until tokens created with this client expire.

86400

Grant Types

☐ Client Credentials
☐ Code Authorization
☒ Token Implicit Grant (Browser)
☐ SAML2 Bearer

Authorized redirect URIs (one per line)

https://omnivoice.tech/media/wizard

Scope

analytics x

architect x

authorization x

presence x

routing x

users x

8. Click **Save**.

Authorization URL

https://apps.mypurecloud.com.au/admin/#/admin/oauth/authorizations/5d87d1bb-5d49-4f11-a857-712a77d3c4e7

Copy

Client ID

Copy

Client Secret

New Secret

Copy

Created By

Date

9/15/2022

Modified By

Date

7/19/2023

9. Copy **ClientID** from details.

10. Go to **OMNI Voice Media** page.

11. Click **Configuration** tab.

12. Paste ClientID to the **Implicit Grant Client Id**.

Prerequisites

1. Install Genesys Cloud AudioHook extension. Configure the credentials using the API key obtained from the "API Key" tab here.
2. Create an Implicit Grant OAuth client in Genesys Cloud.
 1. Create an OAuth client with the following settings:
 1. **Grant type:** Implicit Grant (Browser)
 2. **Authorized redirect URIs:** <https://omnivoice.tech/media/wizard>.
 3. **Scopes**
 1. analytics
 2. authorization
 3. presence
 4. routing
 5. users
 2. Note the client ID and enter it in the below field.

Genesys Implicit Grant Client Id *

Region

Asia Pacific (Sydney) ▼

Please specify your Genesys organization region

START

13. Set **Region** according to your Genesys Cloud organization region.

14. Click **Start**.

15. Wait for the import process completion.

16. Observe **Admin Integrations**. It should have **OMNI Voice Data Actions** integration.

17. Observe **Architect Common Flows**. Ensure that **OMNI Voice Flows** available.

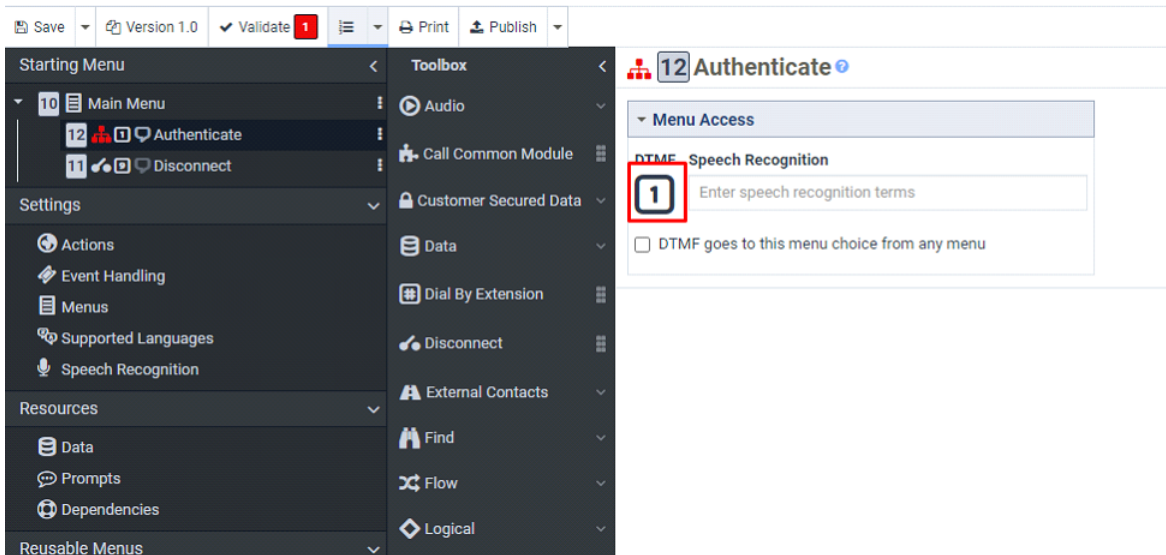
Create or modify IVR's

To provide simple and flexible approach to configure and maintain various business scenarios OMNI Voice have provisioned Architect Common Modules. Example IVR implementation using the provisioned modules is described below, the same approach should be used for adding voice biometrics to any desired IVR/Architect Flow. Please refer to the Genesys Architect manuals for further details about managing Architect flows.

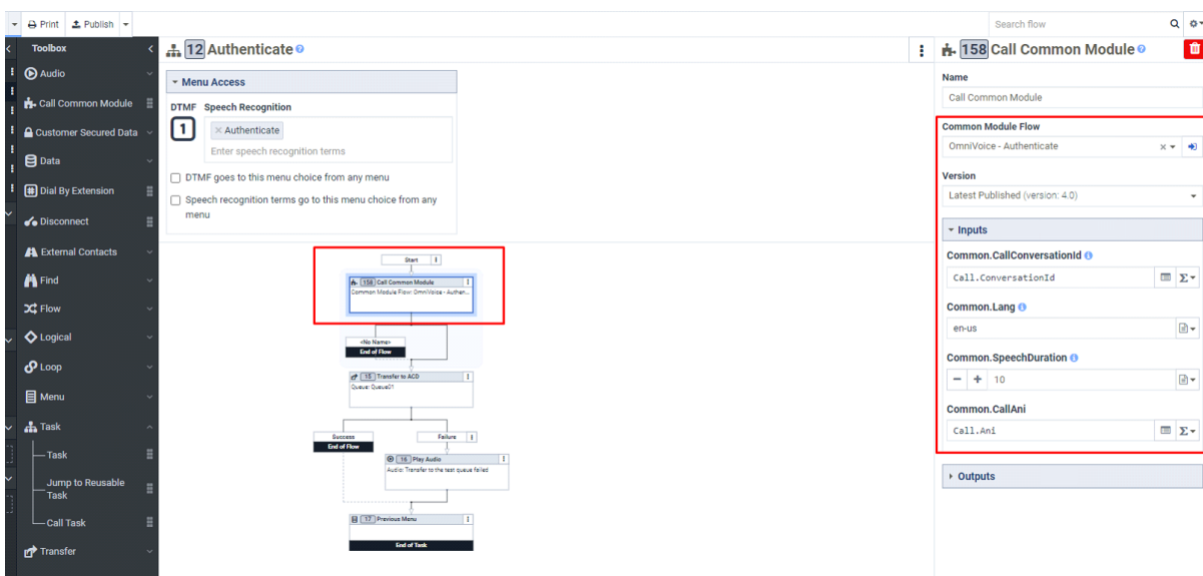
1. Open **Architect**.
2. Select **Inbound Call** flows.
3. Click on **Add** button.
4. Fill **Create new** dialog fields and confirm creation.

5. Create 4 tasks with the following names: **Authenticate, Login, Login Strong, Passive Auth & Enroll.**
6. Assign DTMF 1-4 respectively.

Test manual [Home](#) [Set description...](#)



7. Drag and drop **Call Common Module** from toolbox to the first flow action placeholder.
8. Select added box.
9. On the properties pane click on **Common module Flow** and select **OMNI Voice - Authenticate.**
10. Edit properties on the right pane as on the screenshot below.



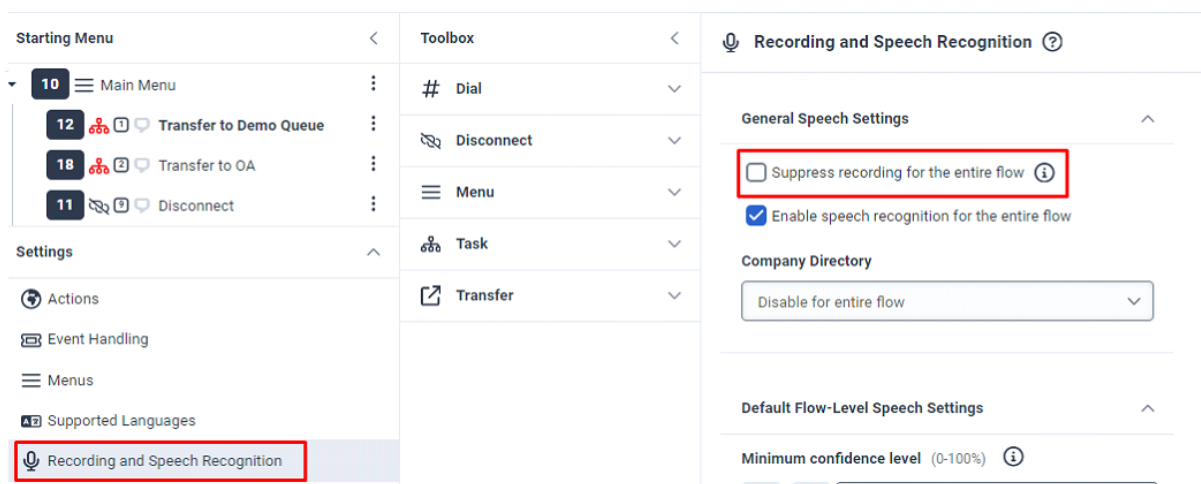
11. Repeat steps 6-10 for Login, Login Strong, Passive Auth & Enroll using the **Common Module Flow** with the corresponding name for each task:

- Login: **OMNI Voice – Voice Login**
- Login Strong: **OMNI Voice – Voice Login Strong**
- Passive Auth & Enroll: **OMNI Voice – Enable Enroll**

12. Add desired action after, in this example IVR it will be **Transfer To ACD** with selected **Queue01**. You should use the desired test queue or create a new one for testing purposes.

13. Add some sound prompts in case of transfer fail.

14. **Important.** Enable Recording for the Flow as on the screenshot below.



15. Save & Publish flow.

16. Switch from Architect to main Genesys Cloud page. Click Admin.

17. Under Telephony click DID Numbers.

18. Assign desired DID number to the new flow.

19. Call selected number to test IVR and OMNI Voice Integration.

Screen Popup Configuration

After integration, it is necessary to configure the Screen popup for queues.

1. Open ****Architect****.
2. Select **In-Queue Call** flows.

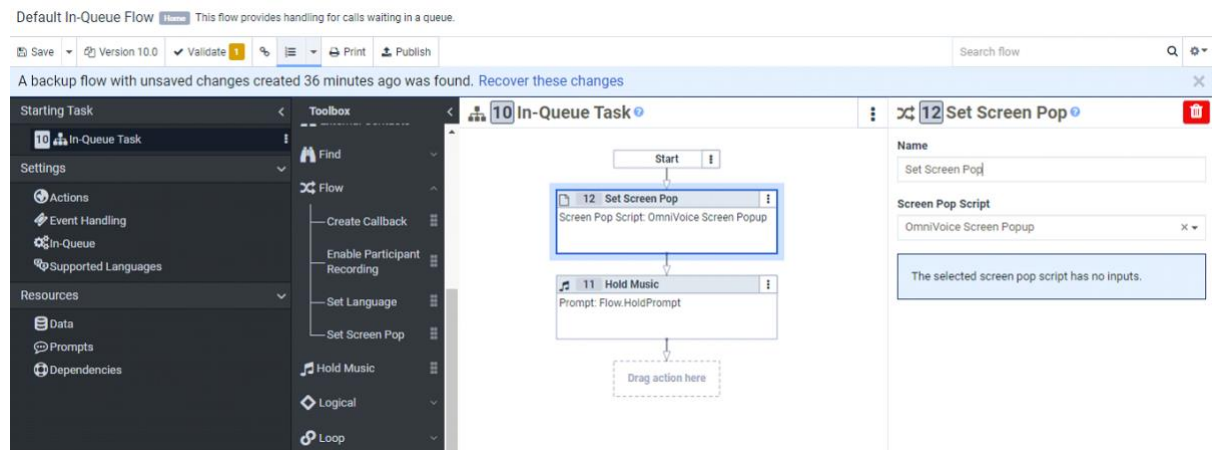
There are two possible options for using screen pop-ups for queues. Modify the existing queue flow and add the created screen pop up where needed:

3.1 Open the flow that you want to use for queue with Voice Biometrics in edit mode.

3.2 Drag and drop **Set Screen Popup** from toolbox to the first flow action placeholder.

3.3 Select added box.

3.4 On the properties pane click on **Set Screen Popup** and select **OMNI Voice Screen Popup**.



3.5 Edit properties on the right pane as on the screenshot below.

3.6 Save changes.

- Use the flow created during OMNI Voice Integration.
- Go to **Admin**.
- Select **Queues**.
- Select the name of the queue from the list for which it is necessary to use flow with VOICE Biometrics.
- Select **Voice Tab** and choose created or modified In-queue Flow.