Lingolette API manual

(document updated 2024-04-20; API version code: 1)

Table of contents

Prerequisites

Authentication

The platform currently uses the SHA256 HMAC authentication algorithm. Three HTTP request headers participate in the authentication process: `x-random` is a random string (32 bytes minimum), `x-auth-id` is your internal identifier and `x-auth-key` is the hash computed based on the two above and your secret key. Your ID and secret will be provided to you by your Lingolette manager.

The hash is computed directly from the random string using your API secret key. You may use the following online tool to test your hash: https://www.freeformatter.com/hmac-generator.html.

Versioning

You need to send a header `x-api-version` indicating desired API version code. This header is obligatory.

API wrappers

We have started adding sample API wrappers for various programming languages: https://github.com/lingolette/api-wrappers.

Data exchange

Requests are sent as HTTP POST with JSON payload, thus `Content-Type: application/json` header is expected. Endpoint URL is composed of a root URL and a collection name, for example `https://lingolette.com/api/org`. API method name to be called is passed directly in the payload:

```
{
    "method": "createUserSession",
    "data":
     {
          "userId": "ddd3c9c2-5797-4215-9390-362106f6442a"
     }
}
```

Data is returned as

```
{
    "isError": false,         // boolean showing if any error was intercepted
    "errMsg": null,           // possible execution error message
    "data":                   // data returned by the method itself
    {
        ...
    }
}
```

Organizations

org (https://lingolette.com/api/org)

Organization is an entity to manage your users inside Lingolette and to create access tokens. Tokens are valid for 30 days.

Method	Description	Input data	Output data

listUsers	List all users of your organization	—	`[{`
			`id: uuid` — user id used to access this user
			`name: str` — user displayed name
			`targetLng: TLng` — target language code
			`nativeLng: Lng` — native language code
			`languageLevel: number` — level user is at
			`createdAt: Date` — date of user creation
			`}]`

listUsersExt	Obsolete. Will be removed in Q3 2024.		

getOverview	Organization overview. Users info includes progress.	—	`{`
			`users: [{`
			`id: uuid` — user id used to access this user
			`name: str` — user displayed name
			`targetLng: TLng` — target language code
			`nativeLng: Lng` — native language code
			`languageLevel: number` — level user is at
			`createdAt: Date` — date of user creation
			`lastMeaningfulActivity: Date` — date of last activity
			`completedExerciseCount: number` — number of successfully completed exercises
			`dictionarySize: number` — active dictionary size
			`}],`
			`admins: [{`
			`id: uuid` — user id
			`name: str` — displayed name
			`}],`
			`orgInfo: {`
			`id: uuid` — organization id
			`name: str` — organization displayed name

			`allowLogin: bool` — whether students can log in via our UI `maxUsers: int` — max students allowed in your org } }

addUser	Add a new user to your organization	{
		`name: str` — user displayed name
		`targetLng: TLng` — target language code
		`nativeLng: Lng` — native language code
		`languageLevel: number` — level user is at
		}
			{
			`id: uuid` — user id used to access this user
			`name: str` — user displayed name
			`targetLng: TLng` — target language code
			`nativeLng: Lng` — native language code
			`languageLevel: LngLevel` — level user is at
			`createdAt: Date` — date of user creation
			}

removeUser	Remove an existing user from your organization.	{
		`userId: uuid` — user identifier
		}
			'ok' string

createUserSession	Create a frontend session for one of your users	{
		`userId: uuid` — user identifier
		}
			{
			`token: str` — access token
			}

After a session is created via `createUserSession` call it can be passed to the frontend as https://lingolette.com/?token=SESSION_TOKEN

Texts

text (https://lingolette.com/api/text)

This collection provides methods for working with text.

Method	Description	Input data	Output data

translate	Translates a word/text
	In case of a single word information about lemma and part of speech is fed back as well
		{
		`text: str` — input word or text
		`targetLng: TLng` — target language code
		`nativeLng: Lng` — native language code
		}
			{
			`translation: string` — chat identifier
			`lemma: ?string` — optional lemma
			`pos: ?string` — optional part of speech
			}

explain	Explains a word/phrase	{
		`text: str` — input word or phrase
		`targetLng: TLng` — target language code
		}
			{
			`explanation: string` — chat identifier
			}

Chats

chat (https://lingolette.com/api/chat)

Chat is an entity to keep communication between a user and an AI-teacher.

Method	Description	Input data	Output data

load	Loads your current chat	—	{
			`id: uuid` — chat identifier
			`messages: ChatMessage[]` — chat messages
			`tokenCount: number` — number of used tokens
			`targetLng: TLng` — chat target language
			`descriptor: str` — auxiliary chat descriptor
			}

clear	Clears your current chat	—	—

voiceInput	Takes voice input, transcribes it and sends both back to requester and into the chat	{
		`b64Data: string` — voice input, base-64 enc.
		`chatId: uuid` — chat identifier
		}
			string

Non-JSON API parts

binary (https://lingolette.com/api/binary)

Some content is not returned as JSON to improve end user experience. Usually in this case the data is server-side events (SSE).

Method	Description	Input data	Output data

startChat	Inits a chat	{
		`timeStamp: Date` — user local time
		`useVoiceOut: ?bool` — use voice output?
		`articleId: ?str` — optional article id for discussion
		}
			`server-side event`
			
			`data: [ChatCommand, ...ChatCommandArguments]`

postToChat	Takes an optional text message and produces synchronous audio/text output	{
		`input: str` — text input
		`useVoiceOut: ?bool` — use voice output?
		`useVoiceIn: ?str` — use voice input?
		}
			`server-side event`
			
			`data: [ChatCommand, ...ChatCommandArguments]`

Appendix A: types and enums

LngLevel

Code	Description
0	Unset (level unknown)
1	A1
2	A2
3	B1
4	B2
5	C1
6	C2
7	None (no prior knowledge)

ChatCommand

Code	Description
0	Close connection
1	Incoming text message, arg 1 — text message

2	Incoming voice-over, arg 1 — audio segment number, arg 2 — base64-encoded audio
3	Chat language changed, arg 1 — language code
4	Custom AI function called, documentation t.b.d

Appendix B: supported languages

All language codes are in ISO-639-1 format: https://localizely.com/iso-639-1-list

target languages (type *TLng*)

ca, de, en, es, fi, fr, it, ja, nb, ms, nl, pl, pt, ru, sv, tr, uk (full mode)
ar, fa, he, ko, lt, ro (test mode, voice input has higher error rate)

native languages (type *Lng*)

Any ISO-639-1 language.