LAIKA API v1.01

Revision history

Version	Changes
v0.1	First version
v0.2	General aspects: added Groups to collections. General aspects: modified endpoint table. Users/retrieve_user: modified output including group, in case of collaborator. Licenses: added general introduction of different types of licenses. Licenses/create_license: modified input/output to manage group licenses. Licenses/list_license: modified input/output to manage group licenses, added filter by user or group. Licenses/stop_license: modified input to differentiate between immediate and renewal stop. Groups: created all the endpoints
v0.3	Users/create_user: modified HTTP status codes. Users/stop_user: changed from POST to PATCH. Users/stop_user: modified HTTP status codes. Users/modify_user: changed from POST to PUT. Users/modify_user: modified HTTP status codes. Users/retrieve_user: changed from POST to GET. Users/retrieve_user: modified HTTP status codes. Users/list_user: changed from POST to GET. Users/list_user: modified HTTP status codes. Licenses/create_license: changed parameter/output data types. Licenses/create_license: modified HTTP status codes. Licenses/stop_license: changed from POST to PATCH. Licenses/stop_license: changed parameter/output data types. Licenses/stop_license: modified HTTP status codes. Licenses/retrieve_license: changed from POST to GET. Licenses/retrieve_license: changed parameter/output data types. Licenses/retrieve_license: modified HTTP status codes. Licenses/list_license: changed from POST to GET. Licenses/list_license: changed parameter/output data types. Licenses/list_license: modified HTTP status codes. Groups/modify_group: changed from POST to PUT. Groups/modify_group: modified HTTP status codes. Groups/retrieve_group: changed from POST to GET. Groups/retrieve_group: changed parameter/output data types. Groups/retrieve_group: modified HTTP status codes. Groups/list_group: changed from POST to GET. Groups/list_group: modified HTTP status codes.

	Patients/create_patient: changed parameter/output data types. Patients/create_patient: modified HTTP status codes. Patients/modify_patient: changed from POST to PUT. Patients/modify_patient: changed parameter/output data types. Patients/modify_patient: modified HTTP status codes. Patients/retrieve_patient: changed from POST to GET. Patients/retrieve_patient: changed parameter/output data types. Patients/retrieve_patient: modified HTTP status codes. Patients/list_patient: changed from POST to GET. Patients/list_patient: modified HTTP status codes. Medical records/create_medical_record: modified HTTP status codes. Sessions/create_session: changed parameter/output data types. Sessions/create_session: modified HTTP status codes. Sessions/retrieve_session: changed from POST to GET. Sessions/retrieve_session: changed parameter/output data types. Sessions/retrieve_session: modified HTTP status codes. Sessions/list_session: changed from POST to GET. Sessions/list_session: modified HTTP status codes. Sessions/stop_session: changed from POST to PATCH. Sessions/stop_session: modified HTTP status codes.
v0.4	Introduced differentiation between "system level authentication" and "user level authentication". Creation of "admin" endpoints. All endpoints modified accordingly.
v0.5	Move the keys (User/Company level) to the headers Introduced "Query Parameters" for the "GET" end points Change refresh token auth Api from "PATCH" to the "GET"
v0.6	Users/admin-modify-user: removed email from the field, because not possible to change email Users/modify-user: removed email from the field, because not possible to change email
v0.7	Authentication mechanism changes for external system: from API_key string to OAuth2.0. New endpoints for external system authentication. All endpoints affected.
v0.8	Users/admin-modify-user: removed

	Users/modify-user: changed from PUT to PATCH Users/admin-stop-user: not available yet, note added
v0.9	Licenses are now purchased directly by users via the LAIKA platform. The platform prompts users to enter their payment credentials during the transaction. The API has been updated to generate a redirect link guiding users to complete the license purchase process. This change affects all license and group-related endpoints and caused Removal of several other endpoints for Sessions.
v0.10	User/admin-create-user: added parameter "restrictedCollaboratorsAccess" Licenses/create-license: removed "pay-per-use" plan
v1.00	**First in production release** User/admin-create-user: international phone prefix needed
V1.01	Corrected typo in /admin-create-patient for "species" parameter (capitalized wording)

General aspects

This document serves as an overview of the LAIKA APIs designed for integration with external services. These APIs facilitate the exchange of data and user interaction within the LAIKA ecosystem.

Authentication

To utilize LAIKA APIs, external systems must first register with LAIKA to establish an API profile. Each registered system is required to obtain **API profile credentials**, provided by AITEM, which serve as authentication parameters for the associated API profile when accessing the LAIKA APIs.

API profile credentials consist in the following fields:

- External system ID
- External system secret

To access LAIKA APIs endpoints, it is required to authenticate the external system which is requiring the access. This mechanism is referred as "**system level authentication**". Every endpoint requires OAuth2.0 web token, this token can be obtained through dedicated LAIKA API endpoints. Calling these endpoints will provide two different tokens:

- **external system access token** allows access to the endpoints; it is valid for a short period of time.
- **external system refresh token** allows to generate a new valid external system access token; it is valid for longer period.

When both access and refresh tokens are expired, new request through LAIKA APIs is required with API profile credentials to obtain valid ones.

Access to data via API credentials is restricted to the permissions granted by the associated API profile.

Both credentials and tokens must be securely stored by the external system, without sharing them with other systems or entities. If these credentials are compromised or lost, AITEM should be promptly notified to deactivate the current credentials and issue new ones; this procedure will make all previously obtained tokens invalid.

Depending on the accessed endpoint, can be required additional authentication of the user who is requiring the access. Also in this case, an OAuth2.0 web token is requested, this token can be retrieved through dedicated LAIKA API endpoints. Two different tokens are used:

- **user access token** allows access to the endpoints; it is valid for a short period of time.
- **user refresh token** allows to generate a new valid user access token; it is valid for longer period.

When both access and refresh tokens are expired, new user login is required with user credentials. This mechanism is referred as "**user level authentication**" and restricts the access to only the data associated with the authenticated user.

For both admin level and user level authentications OAuth2.0 standard (https://oauth.net/2/) is applied to tokens generation and management, with "Client Credential" grant type (https://oauth.net/2/grant-types/client-credentials/).

Authentication is regulated by the following endpoints:

- **/ext-sys-login**: allows external system authentication using their API profile credentials (ID and secret), returns valid external system access and refresh tokens. This endpoint is used when external system tokens are not available or expired.
- **/ext-sys-refresh-token**: generates new valid external system access token, from a valid external system refresh token. This endpoint is used when external system access token is not valid.
- **/user-login**: allows user authentication using their credentials (email and secret), returns valid user access and refresh tokens. This endpoint is used when user tokens are not available or expired.
- **/user-refresh-token**: generates new valid user access token, from a valid user refresh token. This endpoint is used when access token is not valid.

Exemplary flow for admin level authenticated access to API:

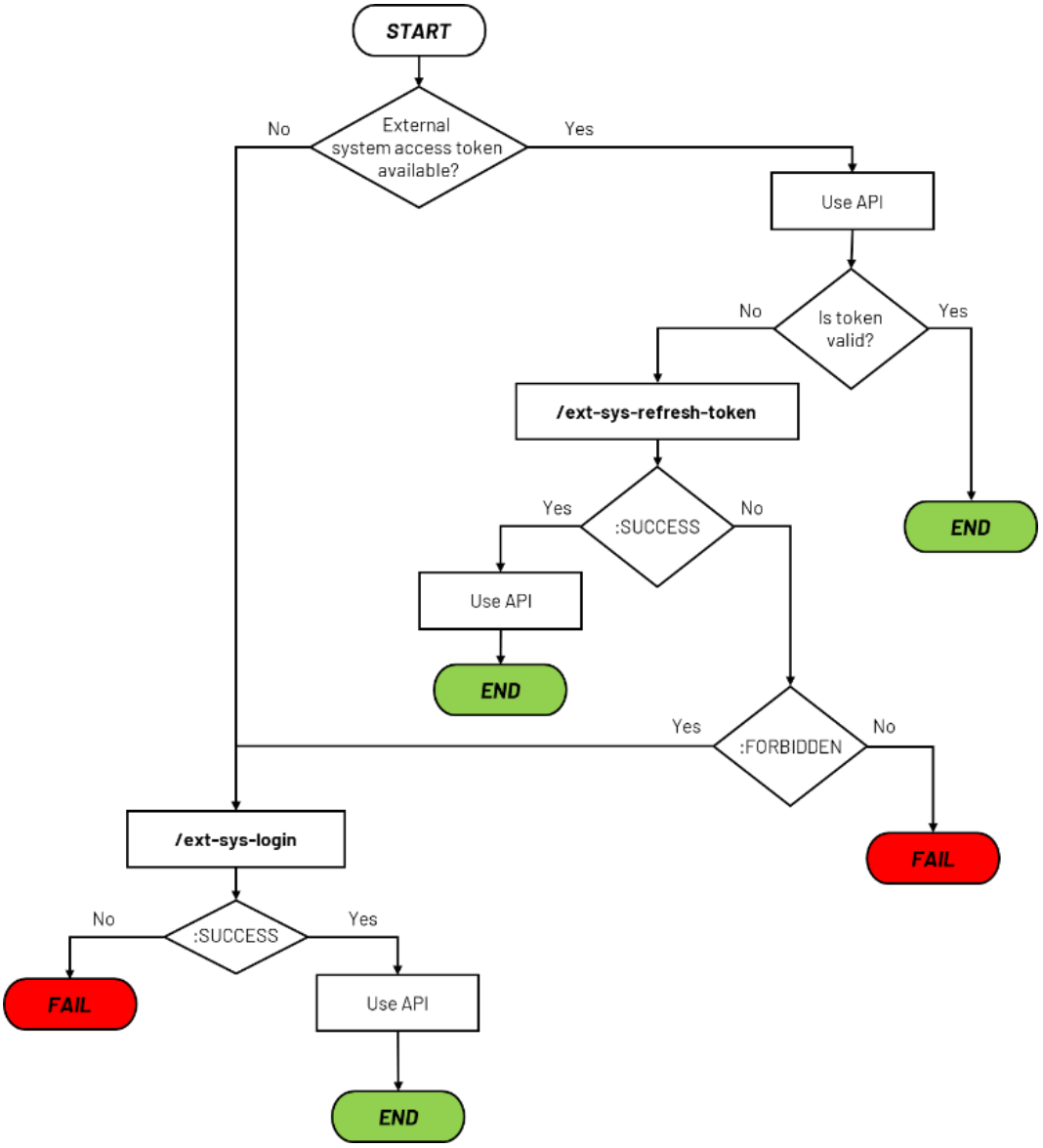

Figure 1: Example of admin level authenticated access to LAIKA API. Note: for example purpose only, actual implementation requires complete error management by external service.

Exemplary flow for user level authenticated access to API:

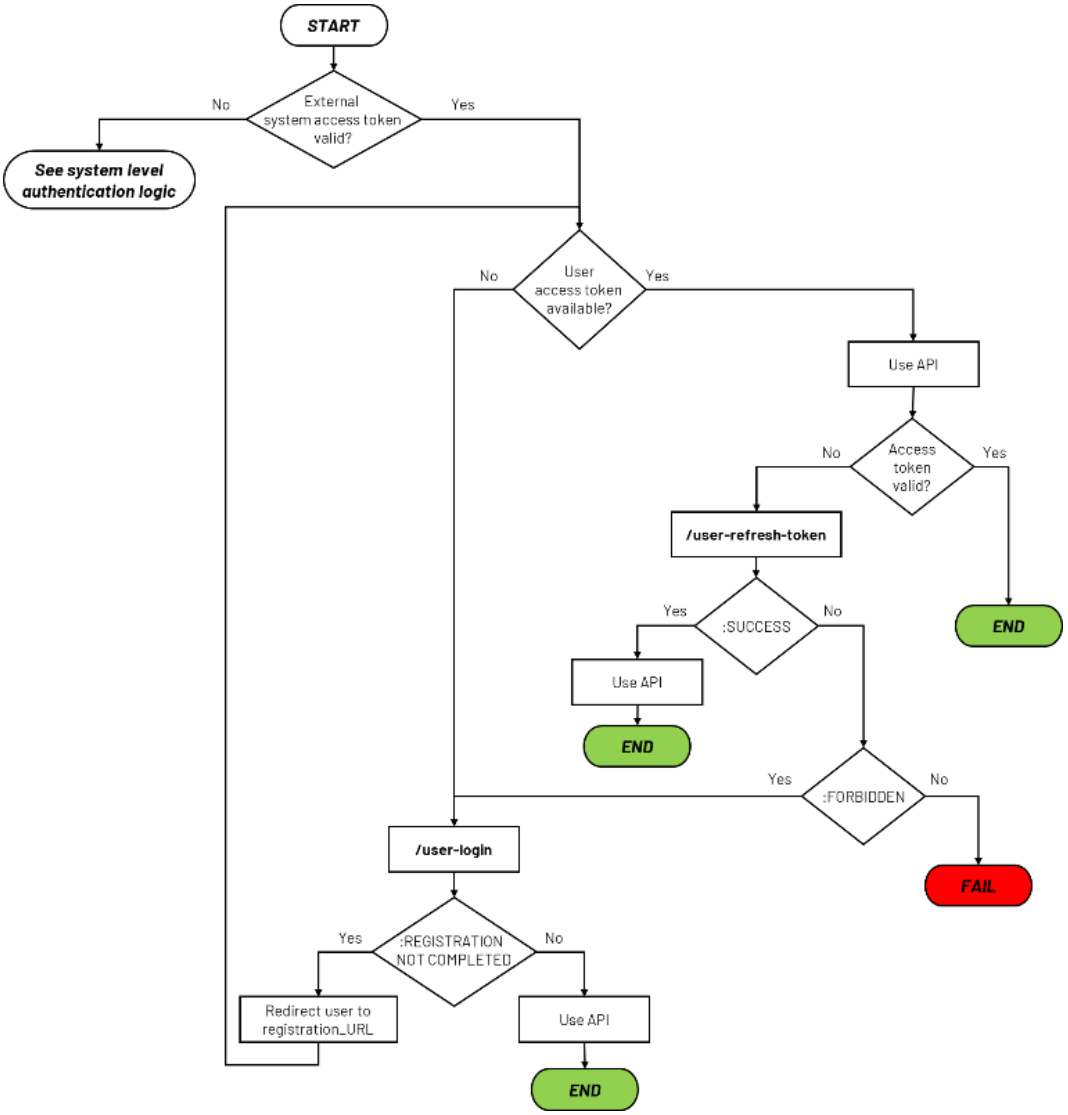

Figure 2: Example of user level authenticated access to LAIKA API. Note: for example purpose only, actual implementation requires complete error management by external service.

/ext-sys-login [POST]

This endpoint allows external system authentication using their credentials in form of external system ID and secret.

If API profile credentials are valid, it returns valid access and refresh tokens that can be used in the other endpoints, otherwise error is returned.

Input:

Body:

Argument	Type	Description
{		
"externalSystem":		
{		
"ID"	string	External system ID
"Secret"	string	External system secret
}		
}		

Output:

Argument	Type	Description
{		
"status"	Number	Status codes
"message"	string	"SUCCESS" if external system logged in successfully. "INVALID INPUT" if wrong input. "FORBIDDEN" if invalid API profile credentials "ERROR" if internal error.
"data":		
{		
"externalSystemAccessToken"	string/null	- External system access token - Null in case of errors
"externalSystemRefreshToken"	string/null	- External system refresh token - Null in case of errors
}		
}		

Status codes:

Code	Description
200	Login was successful
403	Login denied
422	Invalid input
500	Internal error

/ext-sys-refresh-token [GET]

This endpoint allows to generate a new external system access token from a valid external system refresh token.

In case the provided refresh token is expired or not valid, "FORBIDDEN" error message is returned, and new external system login is needed.

If refresh token is valid, new external system access token is returned.

Input:

Headers:

Argument	Type	Description
"externalSystemRefreshToken"	string	External system refresh token

Output:

Argument	Type	Description
{		
"status":	Number	Status codes
"message":	string	"SUCCESS" if login was successful. "FORBIDDEN" external system refresh token is not valid. "ERROR" if internal error.
"data":		
{		
"externalSystemAccessToken"	string/null	- External system access token - Null in case of errors
}		
}		

Status codes:

Code	Description
200	External system login was successful
403	refresh token is not valid
500	internal error

/user-login [POST]

This endpoint allows user authentication using their credentials in form of email and secret.

If user credentials are valid, and user registration is complete, it returns valid access and refresh tokens that can be used in the other endpoints.

If user credentials are valid, but user registration is not completed (e.g. verification code not entered), this endpoint returns error (REGISTRATION NOT COMPLETED) and the URL of the page that allows the user to complete the registration procedure. External service should redirect the user to this page to finalize the onboarding.

If user credentials are valid, but user is not associated to the external service, or user credentials are not valid, error is returned.

Input:

Headers:

Argument	Type	Description
“externalSystemAccessToken”	string	External system access token

Body:

Argument	Type	Description
{		
“user”:		
{		
“email”	string	User email
“secret”	string	User password
}		
}		

Output:

Argument	Type	Description
{		
“status”	Number	Status codes
“message”	string	“SUCCESS” if user logged in successfully. “INVALID INPUT” if wrong input. “NOT AUTHORIZED” if invalid external system access token. “REGISTRATION NOT COMPLETED” if user registration has not been completed. “FORBIDDEN” if invalid user credentials “ERROR” if internal error.
“data”:		
{		
“userAccessToken”	string/null	- User access token - Null in case of errors
“userRefreshToken”	string/null	- User refresh token - Null in case of errors
“registrationURL”	string/null	- URL of registration completion page, in case procedure has not been completed. User should be redirect to this address to complete registration. - Null otherwise
}		
}		

Status codes:

Code	Description
200	Login was successful
401	Invalid external system access token
403	Login denied
422	Invalid input
500	Internal error

/user-refresh-token [GET]

This endpoint allows to generate a new User access token from a valid User refresh token.

In case the provided user refresh token is expired or not valid, "FORBIDDEN" error message is returned, and new user login is needed.

If user refresh token is valid, but user is not associated to the external service, error is returned.

Input:

Headers:

Argument	Type	Description
"externalSystemAccessToken"	string	External system access token
"userRefreshToken"	string	User refresh token

Output:

Argument	Type	Description
{		
"status":	Number	Status codes
"message":	string	"SUCCESS" if login was without any problem. "NOT AUTHORIZED" if invalid external system access token. "FORBIDDEN" user refresh token is not valid. "ERROR" if internal error.
"data":		
{		
"userAccessToken"	string/null	- Updated User access token - Null in case of errors
}		
}		

Status codes:

Code	Description
200	user login was successful
401	invalid external system access token
403	refresh token is not valid
500	internal error

LAIKA API architecture

The LAIKA APIs are structured into several collections:

- **Users**: Represents registered users within LAIKA who seek assistance.
- **Licenses**: Represents LAIKA licenses, each tied to a single user and featuring start and expiration dates. Various license types, with corresponding costs and services, are available based on user preferences. Users without valid licenses can only view past interactions but cannot initiate new requests or upload new data.
- **Groups**: Represents a set of registered users within LAIKA, who share a license and data. The owner of the group is the one who also owns the license and the rights to

define which users are part of the group or not. Only users with valid license type (e.g., "multi-user") can create and manage groups. User invited by the owner to be part of the group, also referred as "collaborators" do not need to own a valid license, since their permission to act on LAIKA is given by group level license.

- **Patients**: Represents individuals receiving care, associated with a single user who creates them in LAIKA (referred to as the owner). Visibility of patient-related data is limited to the owner, or a designated group of users specified by the owner.
- **Medical Records**: Holds patient data such as laboratory results or reports, linked to a specific patient.
- **Sessions**: Represents consultancy sessions within LAIKA, comprising messages exchanged between LAIKA and the user. Each session pertains to a single patient and is accessible only to the owner or a designated group of users specified by the owner.

For each collection, there are five potential actions available:

- **Create**: Add a new object to the collection.
- **Stop**: Deactivate the specified object.
- **Modify**: Alter the properties of the specified object.
- **Retrieve**: Retrieve the properties of the specified object.
- **List**: Provide the complete list of accessible objects within the collection.

Stopping, modifying, and retrieving objects within collections are only permissible for objects created using the corresponding API profile associated with the provided API profile credentials.

Depending on the collection involved, the type of action and the ownership of the object involved, actions can be classified depending on the authentication level required:

- **Admin**: action is performed by the external system administrator; these actions are identified by the prefix "admin". Authentication is required at external system level using only the external system access token.
- **User**: action is performed by the user through the external system; these actions modify or access to objects owned by the user, such as licences, patients, medical records and sessions. Are required both external system level, using external system access token, and user level authentication, using user access token.

The available actions for each collection, and corresponding endpoints, are outlined in the following table:

Collection	Create	Stop	Modify	Retrieve	List
Users	admin-create-user	admin-stop-user	modify-user	admin-retrieve-user retrieve-user	admin-list-user
Licenses	create-license	stop-license		retrieve-license retrieve-active-license	list-license
Group			modify-group		
Patients	create-patient		modify-patient	retrieve-patient	list-patient
Medical records	create-medical-record				
Sessions	create-retrieve-session			create-retrieve-session	

All the endpoints accept arguments and provide output in JSON format.

Users

/admin-create-user [POST]

This endpoint allows external service to create a new user. If valid input is provided, it returns a URL to a webpage where the user can complete the registration (e.g., choose password, mail verification).

It is possible to set the user profile such that their collaborators (i.e. other users invited into their group – applicable only to a subset of licenses, e.g. multi-user) will receive limited access to LAIKA. They will only be able to see some sections, such as legal and login, but not others, such as patient list, integrations or plans. To regulate this behaviour the optional parameter "restrictedCollaboratorsAccess" can be set accordingly.

NOTE: This endpoint returns "**SUCCESS**" message regardless of the user has or has not completed the registration procedure on the provided webpage. In case user registration has not been completed, calling the endpoint ***/user-login-auth*** will return **"REGISTRATION NOT COMPLETED"** message, with URL to a webpage where the user can conclude the process.

Input:

Headers:

Argument	**Type**	**Description**
"externalSystemAccessToken"	string	External system access token

Body:

Argument	**Type**	**Description**
{		
"user":		
{		
"firstName"	string	User first name
"lastName"	string	User last name
"email"	string	User email
"phoneNumber"	string	User phone number – Include international prefix
"restrictedCollaboratorsAccess"	boolean	If True, user collaborators have restricted access to LAIKA platform [optional]
}		
}		

Output:

Argument	**Type**	**Description**
{		
"status"	Number	Status codes
"message"	string	"SUCCESS" if user added correctly. "INVALID INPUT" if wrong input. "INVALID USER" if user already exist.

		"NOT AUTHORIZED" if invalid external system access token. "ERROR" if internal error.
"data":		
{		
"registrationURL"	string/null	- URL of registration page. User should be redirect to this address to complete registration. - Null otherwise
}		
}		

Status codes:

Code	Description
200	user added correctly
401	invalid external system access token
403	user already exists
400	invalid input
500	internal error

/modify-user [PATCH]

This endpoint allows user to modify some of their information.

If user is not associated to the external service, error is returned.

Input:

Headers:

Argument	Type	Description
"externalSystemAccessToken"	string	External system access token
"userAccessToken"	String	User access token

Body:

Argument	Type	Description
{		
"user":		
{		
"firstName"	string/null	User first name [optional]
"lastName"	string/null	User last name [optional]
"phoneNumber"	string/null	User phone number [optional]
}		
}		

Output:

Argument	Type	Description
{		
"status"	Number	Status codes

“message”	string	“SUCCESS” if user modified correctly. “INVALID INPUT” if wrong input. “NOT AUTHORIZED” if invalid external system access token. “FORBIDDEN” if invalid user Access Token. “ERROR” if internal error.
“data”	NULL	
}		

Status codes:

Code	Description
200	user modified correctly
401	invalid external system access token/USERACCESSTOKEN
400	invalid input
500	internal error

/admin-retrieve-user [GET]

This endpoint allows external service to retrieve user information.

If user is not associated to the external service, error is returned.

Input:

Headers:

Argument	Type	Description
“externalSystemAccessToken”	string	External system access token

Query Parameters:

Argument	Type	Description
“userId”	string	User unique identifier

Output:

Argument	Type	Description
{		
“status”	number	Status codes
“message”	string	“SUCCESS” if user retrieved correctly. “INVALID INPUT” if wrong input. “INVALID USER” if user doesn’t exist. “NOT AUTHORIZED” if invalid external system access token. “ERROR” if internal error.
“data”:		
{		
“user”:		
{		
“firstName”	string	User first name

"lastName"	string	User last name
"email"	string	User email
"phoneNumber"	string	User phone number
},		
"groupId"	string/null	- Group unique identifier if applicable. - Null otherwise
}		

Status codes:

Code	Description
200	user retrieved correctly
400	invalid input
401	invalid external system access token
404	user doesn't exist
500	internal error

/retrieve-user [GET]

This endpoint allows user to retrieve some of their information.

If user is not associated to the external service, error is returned.

Input:

Headers:

Argument	Type	Description
"externalSystemAccessToken"	string	External system access token
"userAccessToken"	string	User access token

Output:

Argument	Type	Description
{		
"status"	Number	Status codes
"message"	string	"SUCCESS" if user retrieved correctly. "NOT AUTHORIZED" if invalid external system access token. "FORBIDDEN" if invalid user Access Token. "ERROR" if internal error.
"data":		
{		
"user":		
{		
"firstName"	string	User first name
"lastName"	string	User last name
"email"	string	User email
"phoneNumber"	string	User phone number
}		
"groupId"	string/null	- Group unique identifier if applicable. - Null otherwise

}		

Status codes:

Code	Description
200	user retrieved correctly
401	invalid external system access token/user Access Token
500	internal error

/admin-list-user [GET]

This endpoint allows external service to retrieve the list of their associated users.

Input:

Headers:

Argument	Type	Description
"externalSystemAccessToken"	string	External system access token

Output:

Argument	Type	Description
{		
"status"	Number	Status codes
"message"	string	"SUCCESS" if user added correctly. "NOT AUTHORIZED" if invalid external system access token. "ERROR" if internal error.
"data":		
{		
"users": [ ]	List of string	- List of user unique identifiers if it is retrieved correctly. - Empty otherwise
}		
}		

Status codes:

Code	Description
200	users list retrieved correctly
401	invalid external system access token
500	internal error

Licenses

Within LAIKA, users have the option to subscribe to various types of licenses, offering flexibility to suit their needs. Two distinct options are available:

- "***unlimited***": Users pay a fixed fee every 30 days, granting them the ability to open an unlimited number of sessions within that period.
- "***multi-user***": This option allows a user to act as the owner of a group, with the fee determined by the size of the group (i.e.: fixed + cost per collaborator). The group owner manages the collaborators by both adding or removing them.

Only "multi-user" license type allows the creation of groups.

/create-license [POST]

This endpoint enables the user to obtain a license. If the user has not previously completed a license purchase, the endpoint will return a URL to redirect the user to the LAIKA platform to finalize the transaction. Otherwise, the endpoint will assign a new license to the user and return empty URL.

If the user does not exist or is not associated with the external service, an error response will be returned.

If the license type is a "multi-user" (group) license, the user asking for the license will be designated as the group owner.

NOTE: This endpoint does not check for duplicate license, neither schedule it in the future. Please, verify that there is not already an active license, before to activate a new one.

Input:

Headers:

Argument	**Type**	**Description**
"externalSystemAccessToken"	string	External system access token
"userAccessToken"	String	User access token

Body:

Argument	**Type**	**Description**
{		
"user":		
{		
"invoiceCompany"	string	Name of the company to indicate in the invoice
"invoiceAddress"	string	Address of the company to indicate in the invoice
"invoiceVAT"	string	VAT number of the company to indicate in the invoice
"invoiceSDI"	string	SDI number of the company to indicate in the invoice
"userProfessionalRegisterDistrict"	string	User professional register district
"userProfessionalRegisterNumber"	string	User professional register number
}		
"license":		

{		
"product"	string	User selected type of license
"groupSize"	integer/null	- Maximum number of collaborators in the group - Null if license "product" is not enabled for groups
}		
}		

Output:

Argument	**Type**	**Description**
{		
"status":	Number	Status codes
"message":	string	"SUCCESS" if license has been created. "COMPLETION REQUIRED" if license can be created, but transaction completion is required" "INVALID INPUT" if wrong input. "INVALID USER" if user doesn't exist. "NOT AUTHORIZED" if invalid external system access token. "ERROR" if internal error.
"data":		
{		
"completionURL":	string/null	- URL to complete license purchase transaction. User should be redirect to this address to complete license creation. - Null if not required
}		
}		

Status codes:

Code	**Description**
200	license can be created
401	invalid external system access token/user Access Token
400	invalid input
500	internal error

/stop-license [PATCH]

NOTE: This endpoint disables automatic renewal at end of current invoicing time.

Input:

Headers:

Argument	**Type**	**Description**
"externalSystemAccessToken"	string	External system access token
"userAccessToken"	String	User access token

Body:

Argument	Type	Description
{		
“licenseId”	string	License unique identifier
}		

Output:

Argument	Type	Description
{		
“status”:	Number	Status codes
“message”:	string	“SUCCESS” if license automatic renewal has been disabled correctly. “INVALID INPUT” if wrong input. “INVALID LICENSE” if license is not valid or not associated to the user. “NOT AUTHORIZED” if invalid external system access token. “FORBIDDEN” if invalid user Access Token. “ERROR” if internal error.
“data”:	NULL	
}		

Status codes:

Code	Description
200	license terminated correctly
400	invalid input
401	invalid external system access token/user Access Token
404	invalid license
500	internal error

/retrieve-license [GET]

Input:

Headers:

Argument	Type	Description
“externalSystemAccessToken”	string	External system access token
“userAccessToken”	String	User access token

Query Parameters:

Argument	Type	Description
“licenseId”	string	License unique identifier

Output:

Argument	Type	Description
{		
“status”	Number	Status codes

“message”	string	“SUCCESS” if license retrieved correctly. “INVALID INPUT” if wrong input. “INVALID LICENSE” if license is not valid or not associated to the user. “NOT AUTHORIZED” if invalid external system access token. “FORBIDDEN” if invalid user Access Token. “ERROR” if internal error.
“data”:		
{		
“license”:		
{		
“userId”	string/null	- User associated with license if license retrieved correctly. It indicates the owner if is a group license - Null otherwise
“product”	string/null	- Type of license if license retrieved correctly. - Null otherwise
“startDate”	datetime/null	- License starting date, if license retrieved correctly. - Null otherwise
“expirationDate”	datetime/null	- License expiration date if license retrieved correctly. - Null otherwise
“automaticRenewal”	boolean/null	- “Yes”, “No” if license retrieved correctly. - Null otherwise
“status”	string/null	- “not started”, “active”, “expired” if license retrieved correctly. - Null otherwise
“groupId”	string/null	- Group unique identifier if applicable and license created correctly. - Null otherwise
“groupSize”	integer/null	- Maximum number of collaborators in the group if applicable - Null otherwise
}		
}		

Status codes:

Code	**Description**
200	license retrieved correctly
400	invalid input
401	invalid external system access token/user Access Token
404	invalid license
500	internal error

/retrieve-active-license [GET]

Input:

Headers:

Argument	Type	Description
"externalSystemAccessToken"	string	External system access token
"userAccessToken"	String	User access token

Output:

Argument	Type	Description
{		
"status"	Number	Status codes
"message"	string	"SUCCESS" if user have one active license and has been retrieved correctly. "NONE" if user doesn't have any active license "INVALID INPUT" if wrong input. "NOT AUTHORIZED" if invalid external system access token. "ERROR" if internal error.
"data":		
{		
"license":	object/null	- License object if one active license is associated with user - NULL otherwise
{		
"licenseId"	string/null	- License unique identifier - Null otherwise
"product"	string/null	- Type of license if license retrieved correctly. - Null otherwise
"startDate"	datetime/null	- License starting date, if license retrieved correctly. - Null otherwise
"expirationDate"	datetime/null	- License expiration date if license retrieved correctly. - Null otherwise
"automaticRenewal"	boolean/null	- "Yes", "No" if license retrieved correctly. - Null otherwise
"status"	string/null	- "not started", "active", "expired" if license retrieved correctly. - Null otherwise
"groupId"	string/null	- Group unique identifier if applicable and license created correctly. - Null otherwise
"groupSize"	integer/null	- Maximum number of collaborators in the group if applicable - Null otherwise
}		
}		

Status codes:

Code	Description
200	Active license retrieved correctly, or no active licenses found
400	invalid input
401	invalid external system access token/user Access Token
500	internal error

/list-license [GET]

This endpoint allows user to list all the license associated with them.

Input:

Headers:

Argument	Type	Description
"externalSystemAccessToken"	string	External system access token
"userAccessToken"	String	User access token

Output:

Argument	Type	Description
{		
"status":	number	Status codes
"message":	string	"SUCCESS" if user added correctly. "NOT AUTHORIZED" if invalid external system access token. "FORBIDDEN" if invalid user Access Token. "ERROR" if internal error.
"data":		
{		
"licenses": [ ]	array	- List of licenses if it is retrieved correctly. - Empty otherwise
[		
{		
"licenseId"	string	License unique identifier
"userId"	string	User unique identifier. It indicates the owner if is a group license.
"status"	string	"not started", "active", "expired"
"groupId"	string/null	- Group unique identifier if applicable and license created correctly. - Null otherwise
},..		
]		
}		

Status codes:

Code	Description
200	license list retrieved correctly
401	invalid external system access token/user Access Token
500	internal error

Groups

A new group is created when a new license of type "multi-user" is created. Its unique identifier can be retrieved using **/admin-retrieve-active-license** or **/retrieve-active-license** endpoints, passing the owner user or one of the collaborators.

/modify-group [PUT]

This endpoint allows a user to manage the composition of a group by adding or removing collaborators. Group management actions are handled directly within the LAIKA platform. Users can only modify groups that they own.

The endpoint returns the URL to the specific group management section on LAIKA, where the user can perform these modifications.

Group capacity is defined at license creation. To modify it, is necessary to create a new license (see /create-license).

Input:

Headers:

Argument	**Type**	**Description**
"externalSystemAccessToken"	string	External system access token
"userAccessToken"	String	User access token

Output:

Argument	**Type**	**Description**
{		
"status"	Number	Status code
"message"	string	"SUCCESS" if user can modify a group. "INVALID USER" if user doesn't own any active group "NOT AUTHORIZED" if invalid external system access token. "FORBIDDEN" if invalid user Access Token. "ERROR" if internal error. "INVALID INPUT" if wrong input.
"data"		
{		
"groupManagementURL"	string/null	- URL of group management page. User should be redirect to this address to manage group. - Null otherwise
}		
}		

Status codes:

Code	**Description**
200	user can modify a group

401	invalid external system access token/user Access Token
403	user is not owner of a group
400	invalid input
500	internal error

Patients

/create-patient [POST]

Input:

Headers:

Argument	**Type**	**Description**
"externalSystemAccessToken"	string	External system access token
"userAccessToken"	String	User access token

Body:

Argument	**Type**	**Description**
{		
"patient":		
{		
"name"	string	Patient name
"owner":		
{		
"firstName"	string	Owner first name
"lastName"	string	Owner last name
}		
"gender"	string	'M' for males 'F' for females
"sterilized"	string	'unknown' if information not available 'no' if not sterilized. 'yes' otherwise
"species"	string	'Cat', 'Dog'
"breed"	string/null	NULL if not defined
"birthdate"	datetime	date of patient birth
}		
}		

Output:

Argument	**Type**	**Description**
{		
"status"	Number	Status code
"message"	string	"SUCCESS" if patient created correctly. "INVALID INPUT" if wrong input. "NOT AUTHORIZED" if invalid external system access token. "FORBIDDEN" if invalid user Access Token.

		"ERROR" if internal error.
"data"		
{		
"patientId"	string/NULL	- Patient unique identifier if patient added correctly. - Empty otherwise
}		
}		

Status codes:

Code	Description
200	patient created correctly
401	invalid external system access token/user Access Token
400	invalid input
500	internal error

/modify-patient [PATCH]

Input:

Headers:

Argument	Type	Description
"externalSystemAccessToken"	string	External system access token
"userAccessToken"	String	User access token

Body:

Argument	Type	Description
{		
"patientId"	string	Patient unique identifier
"patient":		
{		
"name"	string/null	Patient name [optional]
"owner":		
{		
"firstName"	string/null	Owner first name [optional]
"lastName"	string/null	Owner last name [optional]
}	string/null	
"gender"	string/null	'M' for males [optional] 'F' for females
"sterilized"	string/null	'unknown' if information not available [optional] 'no' if not sterilized. 'yes' otherwise
"species"	string/null	'cat', 'dog' [optional]
"breed"	string/null	empty if not defined [optional]
"birthdate"	datetime/null	date of patient birth
}		
}		

Output:

Argument	Type	Description
{		
"status"	number	Status code
"message"	string	"SUCCESS" if patient modified correctly. "INVALID PATIENT" if patient doesn't exist or is not associated with user. "NOT AUTHORIZED" if invalid external system access token. "FORBIDDEN" if invalid user Access Token. "ERROR" if internal error. "INVALID INPUT" if wrong input.
"data":	NULL	
}		

Status codes:

Code	Description
200	patient modified correctly
401	invalid external system access token/user Access Token
404	patient doesn't exist user is not associated to patient
400	invalid input
500	internal error

/retrieve-patient [GET]

Input:

Headers:

Argument	Type	Description
"externalSystemAccessToken"	string	External system access token
"userAccessToken"	String	User access token

Query Parameters:

Argument	Type	Description
{		
"patientId"	string	Patient unique identifier
}		

Output:

Argument	Type	Description
{		
"status"	Number	Status code
"message"	string	"SUCCESS" if patient modified correctly. "INVALID PATIENT" if patient doesn't exist or is not associated with user. "NOT AUTHORIZED" if invalid external system access token.

		"FORBIDDEN" if invalid user Access Token. "ERROR" if internal error. "INVALID INPUT" if wrong input.
"data"		
{		
"patient":		
{		
"name"	string	Patient name
"owner":		
{		
"firstName"	string	Owner first name
"lastName"	string	Owner last name
}		
"gender"	string	'M' for males 'F' for females
"sterilized"	string	'unknown' if information not available 'no' if not sterilized. 'yes' otherwise
"species"	string	'cat', 'dog'
"breed"	string/null	null if not defined
"birthdate"	datetime	date of patient birth
}		
}		
}		

Status codes:

Code	Description
200	patient retrieved correctly
401	invalid external system access token/user Access Token
404	patient doesn't exist user is not associated to patient
400	invalid input
500	internal error

/list-patient [GET]

Input:

Headers:

Argument	Type	Description
"externalSystemAccessToken"	string	External system access token
"userAccessToken"	string	User access token

Output:

Argument	Type	Description
{		
"status"	Number	Status code
"message"	string	"SUCCESS" if patients list is retrieved correctly.

		"NOT AUTHORIZED" if invalid external system access token. "FORBIDDEN" if invalid user Access Token. "ERROR" if internal error. "INVALID INPUT" if wrong input.
"data"		
{		
"patients": [ ]	list of string	- List of patients' unique identifiers if it is retrieved correctly. - Empty otherwise
}		

Status codes:

Code	Description
200	patients list retrieved correctly
400	invalid input
401	invalid external system access token/user Access Token
500	internal error

Medical records

/create-medical-record [POST]

Input:

Headers:

Argument	Type	Description
"externalSystemAccessToken"	string	External system access token
"userAccessToken"	String	User access token

Body:

Argument	Type	Description
{		
"patientId"	string	Patient unique identifier
"type"	string	Type of medical record
"sampleDate"	datetime	Sample collection date
"resultDate"	datetime	Results date
"laboratoryData":	array	List of laboratory results - can be empty
[		
{		
"analyte"	string	Analyte name
"result":		
{		
"value"	string	Value associated to analyte
"unit"	string	Unit of measure of the analyte and range

"normRangeMin"	string	Norm Range minimum value associated to analyte
"normRangeMax"	string	Norm Range maximum value associated to analyte
}		
},...		
]		
"findings":	array	List of descriptive findings – can be empty
[		
{		
"subject"	string	Subject of the finding
"content"	string	Content/description of the finding
},...		
]		
}		

Output:

Argument	**Type**	**Description**
{		
"status"	Number	Status code
"message"	string	"SUCCESS" if medical record has been created correctly. "INVALID PATIENT" if patient doesn't exist or is not associated with user. "NOT AUTHORIZED" if invalid external system access token. "FORBIDDEN" if invalid user Access Token. "ERROR" if internal error. "INVALID INPUT" if wrong input.
"data"	Null	
}		

Status codes:

Code	**Description**
200	medical record has been created correctly
401	invalid external system access token/user Access Token
404	patient doesn't exist user is not associated to patient
400	invalid input
500	internal error

Sessions

/create-retrieve-session [POST]

This endpoint allows a user to initiate or retrieve a consultation session on the LAIKA platform. Each consultation session remains active for **72 hours**.

Upon invocation, the endpoint performs the following logic:

- **If an active session already exists** for the specified patient, it returns the existing session's URL.
- **If no active session exists**, it creates a new consultation session and returns the URL to access it.

In both cases, the session is conducted on the **LAIKA platform**, and the user must be redirect to the returned URL to proceed with the consultation.

If patient is not associated with user, endpoint will return an error.

Input:

Headers:

Argument	**Type**	**Description**
"externalSystemAccessToken"	string	External system access token
"userAccessToken"	String	User access token

Body:

Argument	**Type**	**Description**
{		
"patientId"	string	Patient unique identifier
"language"	string	"ITA", "ENG", "FRA", "SPA"
}		

Output:

Argument	**Type**	**Description**
{		
"status"	Number	Status code
"message"	string	"SUCCESS" if session has been started correctly. "INVALID PATIENT" if patient doesn't exist or is not associated with user. "NOT AUTHORIZED" if invalid external system access token. "USER NOT AUTHORIZED" if user is not authorized. "FORBIDDEN" if invalid user Access Token. "ERROR" if internal error. "INVALID INPUT" if wrong input.
"data":		
{		
"URL"	string/null	- URL to session web interface if session has been started correctly. - Null otherwise.
}		
}		

Status codes:

Code	**Description**
200	session has been returned correctly
401	invalid external system access token/user Access Token
403	user is not authorized (e.g. license expiration)

404	patient doesn't exist user is not associated to patient
400	invalid input
500	internal error