האוניברסיטה העברית בירושלים
THE HEBREW UNIVERSITY OF JERUSALEM

UCCA API Reference - User Guide

Web API Endpoint Reference

METHOD	ENDPOINT	USAGE	RETURNS	AUTHORIZED [ADMIN/PM/ANNOTATOR/GUEST]
POST	/login	login	HTTP 200	ALL
users				
GET	/users	Get all users	[] users obj (full)	ADMIN/PM
GET	/users/{id}	Get user	user obj (full)	ADMIN/PM
POST	/users /signup	signup / create new user	user obj (full)	ADMIN/PM
PUT	/users/{id}	Update user	user obj (full)	ADMIN/PM
DELETE	/users/{id}	Delete / inactivate	HTTP 200	ADMIN
GET	/users/{id}/projects	Get user's projects	[] project obj (full)	ADMIN - all PM (projects created_by)
GET	/users/{id}/tasks	Get user's tasks	[] task obj (full)	PM (created_by, assigned) ANNOTATOR (assigned) GUEST (assigned)
categories				
GET	/categories/	Get categories	[] category obj (full)	ADMIN/PM/GUEST
GET	/categories/{id}	Get categories	category obj (full)	ADMIN/PM/GUEST
POST	/categories	create category	category obj (full)	ADMIN/PM
PUT	/categories/{id}	update categories	category obj (full)	ADMIN/PM

DELETE	/categories/{id}	Delete / inactivate	HTTP 200	ADMIN/PM
layers				
GET	/layers	Get layers	[] layers obj (full)	ADMIN/PM/GUEST
GET	/layers/{id}	Get layer	layer obj (full)	ADMIN/PM/GUEST
POST	/layers	Create Layer	layer obj (full)	ADMIN/PM
PUT	/layers/{id}	Update Layer	layer obj (full)	ADMIN/PM
DELETE	/layers/{id}	Delete / inactivate	HTTP 200	ADMIN/PM
sources				
GET	/sources	Get sources	[] source obj (full)	ADMIN/PM
GET	/sources/{id}	Get source	source obj (full)	ADMIN/PM
POST	/sources	Create source	source obj (full)	ADMIN/PM
PUT	/sources/{id}	Update source	source obj (full)	ADMIN/PM
DELETE	/sources/{id}	Delete / inactivate	HTTP 200	ADMIN/PM
passages				
GET	/passages	Get passages	[] passage obj (full)	ADMIN/PM
GET	/passages/{id}	Get passage	passage obj (full)	ADMIN/PM
POST	/passages	Create passage	passage obj (full)	ADMIN/PM
PUT	/passages/{id}	Update passage	passage obj (full)	ADMIN/PM
DELETE	/passages/{id}	Delete / inactivate	HTTP 200	ADMIN/PM

GET	/passages/{id}/tasks	Get passage's tasks	task obj (full)	ADMIN/PM
GET	/passages/{id}/projects	Get projects with task using the passage	project obj (full)	ADMIN/PM
projects				
GET	/projects	Get projects	[] project obj (full)	ADMIN/PM/ANNOTATOR/GUEST
GET	/projects/{id}	Get project	project obj (full)	ADMIN/PM
GET	/projects/{id}?fullrecursive=true	Get project full recursive + Require Auth	project obj (full)	ADMIN/PM
POST	/projects	Create project	project obj (full)	ADMIN/PM
PUT	/projects/{id}	Update project	project obj (full)	ADMIN/PM
DELETE	/projects/{id}	Delete / inactivate	HTTP 200	ADMIN/PM
~~GET~~	~~/projects/{id}/tasks~~	~~Get project's tasks~~	~~[] task obj (simp)~~	
Tasks				
GET	/tasks	**For Admin** Get tasks	[] task_in_chartobj (full)	ADMIN/PM/GUEST/ANNOTATOR
GET	/tasks/{id}	**For Admin** Get task	Task_in_chart obj (full)	ADMIN/PM
POST	/tasks	**For Admin** Create task in project	Task_in_chart obj (full)	
PUT	/tasks/{id}	**For Admin** Update task in project.	Task_in_chart obj (full)	ADMIN/PM
GET	/user_tasks/{id}	**For annotator** Get task	task obj (full)	ADMIN/PM/GUEST/ANNOTATOR
PUT	/user_tasks/{id}/{save_type}	**For annotator** Update task in		GUEST/ANNOTATOR

		project. save_type is either [autosave, draft, submit]		
DELETE	tasks/{id}	**For Admin** Delete / inactivate	HTTP 200	ADMIN/PM
General				
POST	/forgot_password	Reset password and send the new one to user email	HTTP 200	ADMIN/PM/GUEST/ ANNOTATOR
POST	/change_password	Change the password to the users params	HTTP 200	ADMIN/PM/GUEST/ ANNOTATOR
POST	/logout	update token - expired	HTTP 200	ADMIN/PM/GUEST/ ANNOTATOR

Web API Object Models

user object (full)

Key	Value Type	Value Description
id		
first_name		
last_name		
email		
organization		
affiliation		
role	ADMIN/GUEST/PM/ANNOTATOR	
created_by	user obj (simp)	
created_at		
updated_at		
is_active		

user object (simplified)

Key	Value Type	Value Description
id		
first_name		
last_name		

category object (full)

Key	Value Type	Value Description
id		
name		
description		
abbreviation		
tooltip		
is_default		
created_by	user obj (simp)	
created_at		
updated_at		
is_active		

category object (simplified)

Key	Value Type	Value Description
id		
name		

category_in_layer object

Key	Value Type	Value Description
id		
name		
was_default		was default at the time of layer creation
shortcut_key		as defined by the connecting table layers_categories
description		
abbreviation		
tooltip		
parent		

layer object (full)

Key	Value Type	Value Description
id		
name		
description		
parent	layer obj (full)	
children	array layer obj (simp)	
type	enum [REFINEMENT, COARSENING, EXTENSION, ROOT (default)]	
tooltip		
projects	array projects obj (simp)	
categories	array category_in_layer obj	
restrictions	array layer_restriction obj	
created_by	user obj (simp)	
created_at		
updated_at		
is_active		

layer object (simplified)

Key	Value Type	Value Description
id		
name		
type		

layer_restriction object

Key	Value Type	Value Description
type	enum [REQUIRE_SIBLING, REQUIRE_CHILD, FORBID_SIBLING, FORBID_CHILD, FORBID_ANY_CHILD]	VALUE (checking time) REQUIRE_SIBLING (onFinish) REQUIRE_CHILD (onFinish) FORBID_SIBLING (onFinish) FORBID_CHILD (onAdd) FORBID_ANY_CHILD (onAdd)
categories_1	array categories obj (simp)	
categories_2	array categories obj (simp)	

e.g.

```
[
        {
                "type":"REQUIRE_SIBLING",
                "Categories_1" : [{"id":1},{{"id":2}],
                "Categories_2" : [{"id":3},{{"id":4}]
        }
]
```

source object (full)

Key	Value Type	Value Description
id		
name		
description		
created_by	user obj (simp)	
created_at		
updated_at		
is_active		

source object (simplified)

Key	Value Type	Value Description
id		
name		

passage object (full)

Key	Value Type	Value Description
id		
text		
type	enum [PRIVATE, PUBLIC]	
source	source obj (simplified)	
created_by	user obj (simp)	
created_at		
updated_at		
is_active		

passage object (simplified)

Key	Value Type	Value Description
id		
type		
short_text	varchar (30)	first 30 chars of the text

project object (full)

Key	Value Type	Value Description
id		
name		
description		
layer	layer obj (simp)	
tooltip		
tasks	array task obj (simp)	
created_by	user obj (simp)	
created_at		
updated_at		
is_active		

project object (simplified)

Key	Value Type	Value Description
id		
name		

project_for_annotator object - for all types of tasks

Key	Value Type	Value Description
id		
name		
layer	layer obj (full)	
description		

task object (full) - for annotator

Key	Value Type	Value Description
id		
parent	task_in_chart obj	
children	array task obj (simp)	
type	enum [ANNOTATION, TOKENIZATION, REVIEW]	
status	enum [NOT_STARTED, ONGOING, SUBMITTED, REJECTED]	
project	project_for_annotator obj	
user	user obj (simp)	
passage	passage obj (full)	
tokens	array token obj (full)	Of the root tokenization task
annotation_units	array annotation_units obj (full)	Cloned from the parent, independent
is_demo	bool	
manager_comment		
created_by	user obj (simp)	
created_at		
updated_at		
is_active		

task object (simplified)

Key	Value Type	Value Description
id		
type		
user	user obj (simp)	
project	project obj (simp)	

task_in_chart object

Key	Value Type	Value Description
id		
parent	task obj (simp)	
children	array task obj (simp)	
type	enum [ANNOTATION, TOKENIZATION, REVIEW]	
status	enum [NOT_STARTED, ONGOING, SUBMITTED, REJECTED]	
project	project obj (simp)	
user	user obj (simp)	
passage	passage obj (simp)	
is_demo	bool	
manager_comment		
created_by	user obj (simp)	
created_at		
updated_at		

is_active		

token object (full)

Key	Value Type	Value Description
id		
start_index		
end_index		
text		
require_annotation		For determining if a token is a punctuation or not. punctuation are not require the annotator action (update only by server, regardless the client's sent value)
tokenization_task_id		the id of the original tokenization task

token object (simplified)

Key	Value Type	Value Description
id		

annotation_unit object (full)

Key	Value Type	Value Description
id		
annotation_unit_tree_id	varchar	E.g. “1.1.2”
task_id		
type	enum [IMPLICIT, REGULAR]	
is_remote_copy	bool (default false)	true if the unit is REMOTE
comment		
categories	array category obj (simp)	**Simple obj, because the full is already in the task’s project’s layer obj**
children_tokens	array tokens obj (simp)	Immediate tokens of this annotation_unit
children	annotation_unit obj (full)	
parent_id		null if the unit is the root passage. If the unit is remote copy, this is the ID of the parent
gui_status	enum [HIDDEN, COLLAPSE, OPEN (default state)]	

annotation_unit object (simplified)

Key	Value Type	Value Description
id		

Delete / Inactivate

- An asset can be deleted by admin or by owner if no other asset is connected to it.
- If another asset is connected the asset is not deleted and is_active is set to false; Thus, no new asset can be connected to it.

Response

- response for create + update requests return the simplified model

Parameters + Pagination

Pagination Params

offset
limit

Default limit is set to 100;
limit param is overriding the default limit.
if limit = -1 return all items.

Search Query:

Search by:
?{key1}={value1)&{key2}={value2)...
(OPTIONAL: ?search="...")

Activating/Inactivating an Asset

- **Project manager / admin:** An asset with is_active=false cannot be used as a data member for new assets (e.g., a task cannot be used as a parent task, a source cannot be used for a new passage etc.). Inactive assets appear with a different color in the admin screens.
- **Inactive users:** an inactive user cannot login into the system. He receives a message: "Your account has been inactivated".
- **Annotator / guest:** An annotator/guest doesn't see inactive tasks assigned to her or inactive demo tasks (even if he already started them before they turned inactive). A guest **cannot** see inactive layers and categories.
- **Dependencies:** if a project is activated/inactivated, all its tasks are also activated/inactivated. In all other cases, activating/inactivating an asset does not influence other assets. (OPTIONAL)

Response Status Codes

The API uses the following response status codes, as defined in the RFC 2616 and RFC 6585:

STATUS CODE	DESCRIPTION
200	OK - The request has succeeded. The client can read the result of the request in the body and the headers of the response.
201	Created - The request has been fulfilled and resulted in a new resource being created.
202	Accepted - The request has been accepted for processing, but the processing has not been completed.
204	No Content - The request has succeeded but returns no message body.
304	Not Modified. See Conditional requests.
400	Bad Request - The request could not be understood by the server due to malformed syntax. The message body will contain more information; see Error Details.
401	Unauthorized - The request requires user authentication or, if the request included authorization credentials, authorization has been refused for those credentials.
403	Forbidden - The server understood the request, but is refusing to fulfill it.
404	Not Found - The requested resource could not be found. This error can be due to a temporary or permanent condition.

429	Too Many Requests - Rate limiting has been applied.
500	Internal Server Error. You should never receive this error because our clever coders catch them all ... but if you *are* unlucky enough to get one, please report it to us through a comment at the bottom of this page.
502	Bad Gateway - The server was acting as a gateway or proxy and received an invalid response from the upstream server.
503	Service Unavailable - The server is currently unable to handle the request due to a temporary condition which will be alleviated after some delay. You can choose to resend the request again.

Error Details

The API uses two different formats to describe an error.

Authentication error object

Whenever the application makes requests to the API which are related to authentication or authorization, e.g. retrieving an access token or refreshing an access token, the error response follows RFC 6749 on The OAuth 2.0 Authorization Framework.

KEY	VALUE TYPE	VALUE DESCRIPTION
error	string	A high level description of the error as specified in RFC 6749 Section 5.2.
error_description	string	A more detailed description of the error as specified in RFC 6749 Section 4.1.2.1.

Below is an example of a failing request to refresh an access token.

```
$ curl -H "Authorization: Basic Yjc...cK" -d grant_type=refresh_token -d
refresh_token=AQD...f0 "https://accounts.spotify.com/api/token"
```

```
{
    "error": "invalid_client",
    "error_description": "Invalid client secret"
}
```

Regular error object

Apart from the response code, unsuccessful responses return information about the error as an error JSON object containing the following information:

KEY	VALUE TYPE	VALUE DESCRIPTION
status	integer	The HTTP status code (also returned in the response header; see Response Status Codes for more information).
message	string	A short description of the cause of the error.

Here, for example is the error that occurs when trying to fetch information for a non-existent track:

```
$ curl -i "https://api.spotify.com/v1/tracks/2KrxsD86ARO5beq7Q0Drfqa"

HTTP/1.1 400 Bad Request
{
    "error": {
        "status": 400,
        "message": "invalid id"
    }
}
```