FLUXX

Fluxx API Documentation Page

Table of Contents

Overview

The Fluxx API documentation page provides you with how to create access tokens, the endpoints that you can use in your API calls and some examples of API calls for each class.

Requirements for API Documentation Page Access

- To access the API documentation page, the user must either:
    - Be an **Admin**
    - Have a profile with the **Super User** category
    - Have a profile with the **Developer** category

How to Access the API Documentation Page

To access the documentation page, the user **must be logged in** to your Fluxx environment. From there, you will need to append one of the following paths to the end of your URL:

- Version One: /api/rest/v1/doc
    - This version was primarily used to handle data in bulk and the type of output is defined by the parameter style or 'api_style' and it's three values (full, compact, and detail) as seen in the Admin Panel field configuration.

Version One (v1) of the Fluxx API will be retired soon. Clients should no longer be using v1 for integrations and should start moving to Version Two (v2).

- Version Two: /api/rest/v2/doc
    - This version will return more granular results defined in the 'cols' parameter and will return values for the fields provided.
    - Production Example: https://example.fluxx.io/api/rest/v2/doc
    - Pre-production Example: https://example.preprod.fluxxlabs.com/api/rest/v2/doc

---

Break Down of Samples

List/Search
URI: /:model_type
REST method: GET
This endpoint allows you to retrieve data in increments of 25 per page.

Alternate URI: /:model_type/list
REST method: POST
This endpoint also allows you to retrieve data in increments of 25 per page and has the added ability to use the filter parameter using a POST method.

Fetch
URI: /:model_type/:model_id
REST method: GET
This endpoint allows you to retrieve data of a single record.

Create
URI: /:model_type
REST method: POST
This endpoint allows you create records/documents. A data payload must be provided.

We highly recommend including the created_by field in create API calls as this can help identify which records were created by the API (vs a user).

Update
URI: /:model_type/:model_id
REST method: PUT
This endpoint allows you update existing records/documents in the system. A data payload must be provided.

We highly recommend including the updated_by field in update API calls as this can help identify which records were created by the API (vs a user).

Delete
URI: /:model_type/:model_id
REST method: DELETE
This endpoint allows you delete existing records/documents in the system.

General Notes

- Dates are to be formatted in iso8601.
  - Example, YYYY-MM-DDTHH:MM:SSZ
- If you want to remove a value from a record use "".
  - Example:

  -
    ```
    curl --globoff -X PUT "https://<your Fluxx site>/api/rest/v2/grant_request/1" -F 'data={"project_title":"","updated_by_id":263502}' -F 'cols=["program_organization_id","program_id","project_title","updated_by"]' --header 'Authorization: Bearer <--token-->'
    ```
- If you want to remove a related record, use "id":"".
  - Example:

  -
    ```
    curl --globoff -X PUT "https://<your Fluxx site>/api/rest/v2/grant_request/1" -F 'data={"grantee_signatory_id":{"id":""},"updated_by_id":263502}' -F 'cols=["program_organization_id","program_id","grantee_signatory_id","updated_by"]' --header 'Authorization: Bearer <--token-->'
    ```

Best Practices

- **Containing Values** – When it comes to containing the values in your query, our best practice is using single quote to enclose the entire URL, and using double quotes when containing items like keys and values within the URL.
- **Nested Lists** – When using parameters like "*relation*", you will need to use two levels of lists. The first level using curly braces '{ }' to enclose the entire list and square braces '[ ]' to enclose sub lists, such as fields from a specific model type.
  - *Example: {"bank_accounts":["owner_user_id","organization_owner_id"], "request_transaction_ids":["amount_paid","due_at"]}*