Table of Contents

Curl commands for kickstarting

While working with eLabFTW's API, before implementing a new feature it's wise to test if something is even possible through **curl** commands. We'll be referring to this action as *kickstarting* a feature.

On eLabFTW v5.0.4

Creating an empty sample

After exporting the API key for your account as a Bash variable (`export ELAB_KEY=8-8bd3f…`), run the following command:

```
curl -H "Content-Type: application/json" -H "Authorization: $ELAB_KEY" \
    -d '{ "category_id": 10, "tags": ["primo campione", "altro tag"] }' \
    -k -X POST https://old.elabftw.net/api/v2/items/
```

- The header (`-H or --header`) section tells eLab that the content is in json format and that the authentication must occur via API key.
- The data (`-d` or `--data`) section contains the payload, which unfortunately for eLabFTW 5.0.4 is limited to category ID (10 for "Sample" in OUR VERY OWN instance of eLab) and tags.
- `-k` or `--insecure` makes curl skip the verification step and proceed without checking; necessary if your instance of eLab has a self-signed SSL certificate - or no certificate at all.
- The method (`-X` or `--request`) section decides the method to use when starting the transfer; HTTP POST is used to create new content on eLab.

- Replace `https://old.elabftw.net/api/v2/items/` with the url to the API of your instance of eLabFTW (e.g. "https://elabftw.selfhosted.it:8080/api/v2/items").

NOTE	In Bash single (') and double (") quotation marks are slightly different, as single quotes treat the content literally and do not expand variables (i.e. `'Authorization: $API_KEY'` will try to authenticate you to the server literally as **`$API_KEY`** instead of using your key - the content of the variable).

Opening an item by ID

If you know the ID of the item you wish to get data about, the curl command would be:

```
ITEM_ID=123
curl -H "Content-Type: application/json" -H "Authorization: $ELAB_KEY" \
    -k -X GET https://old.elabftw.net/api/v2/items/$ITEM_ID
```

The standard output is "minified", unless jq is added to the pipeline:

```
curl -H "Content-Type: application/json" -H "Authorization: $ELAB_KEY" \
    -k -X GET https://old.elabftw.net/api/v2/items/$ITEM_ID | jq | less
```

On eLabFTW v5.1.15

Creating an empty sample

v5.1.x has huge improvements for item creation and update. After exporting the API key for your account as a Bash variable (`export ELAB_KEY=8-8bd3f⋯`), run the following command:

```
curl -H "Content-Type: application/json" -H "Authorization: $ELAB_KEY" \
    -d '{ "template": 10, "title":"My new sample", "body":"<p>Brief description in raw
HTML</p>", "tags": ["primo campione", "altro tag"] }' \
    -k -X POST https://demo.elabftw.net/api/v2/items/
```

- The header (`-H or --header`) section tells eLab that the content is in json format and that the authentication must occur via API key.
- The data (`-d` or `--data`) section contains the payload; in v5.1.15 "category_id" has been replaced by "template" and many more fields have been added, which means the user is now capable of populating an item with a single command much like how it's always been the case for the experiments.
- `-k` or `--insecure` makes curl skip the verification step and proceed without checking; necessary if your instance of eLab has a self-signed SSL certificate - or no certificate at all.
- The method (`-X` or `--request`) section decides the method to use when starting the transfer; HTTP POST is used to create new content on eLab.

- Replace `https://demo.elabftw.net/api/v2/items/` with the url to the API of your instance of eLabFTW (e.g. "https://elabftw.selfhosted.it:8080/api/v2/items").

Creating a sample from JSON file

Assuming you have a JSON file compliant to your eLabFTW instance, you can run the previous command in a "quieter" way:

```
curl -H "Content-Type: application/json" -H "Authorization: $ELAB_KEY" \
    -d '@/path/to/sample.json' \
    -k -X POST https://demo.elabftw.net/api/v2/items/
```

NOTE	The path to the JSON file can be relative (e.g. `@tests/example.json`). The `@` symbol is required.

Visualize all items

Item visualization works the same as in v5.0.4.

```
curl -H "Content-Type: application/json" -H "Authorization: $ELAB_KEY" \
    -k -X GET https://demo.elabftw.net/api/v2/items?limit=9999
```

Note that this only goes up to ten thousands items, and I've yet to find a way to show every item regardless of the total number in the database. But it's probably useless since the items with the highest STD-ID's will always be on top of the list anyways. Even if we ever DO create more than 10k items on the database and it somehow doesn't collapse it's almost impossible that the most recently created item is somehow lower than 10k other items.

Opening an item by ID

```
ITEM_ID=123
curl -H "Content-Type: application/json" -H "Authorization: $ELAB_KEY" \
    -k -X GET https://demo.elabftw.net/api/v2/items/$ITEM_ID
```

The standard output is "minified", unless jq is added to the pipeline:

```
curl -H "Content-Type: application/json" -H "Authorization: $ELAB_KEY" \
    -k -X GET https://old.elabftw.net/api/v2/items/$ITEM_ID | jq | less
```

Looking for items with a specific title, body or elabid

Let's suppose we want to look up an item by title, body or elabid. The appropriate key to use is `q` (*query*):

```
curl -H "Content-Type: application/json" -H "Authorization: $ELAB_KEY" \
    -k -X GET https://demo.elabftw.net/api/v2/items?q='Na-25' | jq | less
```

This returns every item whose title, body or elabid contains the string "Na-25". It seems to be caps-insensitive.

Selecting metadata/extra_field/STD-ID values from every item with 'Na' in its name

```
curl -H "Content-Type: application/json" -H "Authorization: $ELAB_KEY" \
    -k -X GET "https://demo.elabftw.net/api/v2/items?q=na&limit=9999" -s | \
    jq -r '.[] | "\(.metadata | fromjson | .extra_fields["STD-ID"].value)"' \
    2>/dev/null | grep -v null
```

Where `-s` runs curl in *silent mode*, while `jq` is used to process the string downloaded by the command.

- `jq -r '.[]'` iterates over every element of the array.
- `\(.metadata | fromjson | .extra_fields["STD-ID"].value)`:
  - `fromjson` converts the string (minified) to JSON.
  - `.extra_fields["STD-ID"].value` extracts the values of STD-ID for every item.
- `2>/dev/null` redirects errors from jq to /dev/null.
- `grep -v null` excludes from output null values of the STD-ID.

Debugging with curl

Oftentimes executing a certain command returns a different result from what's expected. What follows is a "journal" of curl prompts I've tried to achieve a certain result which instead returned something wrong or unexpected.

On eLabFTW v5.1.15

Posting metadata in "extra fields" [Items]

Link to GitHub Issue: elabftw / issues / 5480.

WARNING	The issue has been closed as the problem is solved in v5.2.0-alpha.

I want to create a new item on the demo instance (demo.elabftw.net) via API using curl; I copy the item model directly from the official API documentation with no editing and paste it in a variable `$MYJSON`:

```
export MYJSON='{
```

```
  "body": "<h1>Section title</h1><p>Main text of resource</p>",
  "canread": "{\"base\": 30, \"teams\": [], \"users\": [], \"teamgroups\": []}",
  "canwrite": "{\"base\": 20, \"teams\": [], \"users\": [], \"teamgroups\": []}",
  "content_type": 1,
  "metadata": "{ \"extra_fields\": { \"For example\": { \"type\": \"text\", \"value\":
\"With a value\", \"required\": true, \"description\": \"An extra field of type text\"
} } }",
  "rating": 0,
  "status": 1,
  "template": 1,
  "tags": [
    "TIRF",
    "Nikon",
    "mandatory booking"
  ],
  "title": "TIRF microscope"
}'
```

After that I launch curl with the following options - where $ELAB_KEY is my personal API key:

```
curl -H "Content-Type: application/json" -H "Authorization: $ELAB_KEY" \
    -k -X POST https://demo.elabftw.net/api/v2/items/ \
    -d "$MYJSON"
```

All is good, I don't get any error and the new item is created on the demo. Title, tags, template/category and status are exactly what I want; however:

- The body field is clear;
- The metadata/extra fields are clear - or set to the template's default.

I've also tried many small changes, like:

- Storing my JSON in a file instead of a Bash variable;
- Pasting the contents of the JSON directly into my terminal while running curl;
- Editing the model to fit a different template's default extra fields:

```
{
  "body": "<h1>Section title</h1><p>Main text of resource</p>",
  "canread": "{\"base\": 30, \"teams\": [], \"users\": [], \"teamgroups\": []}",
  "canwrite": "{\"base\": 20, \"teams\": [], \"users\": [], \"teamgroups\": []}",
  "content_type": 1,
  "metadata": "{\"extra_fields\": {\"Concentration\": {\"type\": \"number\",
\"value\": \"20\", \"description\": \"in µg/µL\"}, \"Growth temperature\": {\"type\":
\"text\", \"value\": \"50\", \"description\": \"in °C\"}}}",
  "rating": 3,
  "status": 1,
  "template": 5,
```

```
  "tags": [
    "TIRF",
    "Nikon",
    "mandatory booking"
  ],
  "title": "TIRF microscope"
}
```

While the original example contains a single extra field called "For example", my edited file contains two extra fields supported officially on the eLabFTW's demo instance (**Concentration** and **Growth temperature**) for the *Plasmid* template. I've also changed the template value from 1 to 5 (Plasmid) and the rating from 0 (no rating) to 3, discovering that not even the rating field is set to the value I provide.