Integration of an electronic lab notebook into a measurement control program

Hanno Dierke
Physikalisch-Technische Bundesanstalt, Bundesallee 100, 38116 Braunschweig
mailto:hanno.dierke@ptb.de

For a reliable and traceable documentation of experimental results, suitable tools and processes are required. At PTB, an electronic laboratory book can be used to store and make available metadata, e.g. on environmental conditions and measurement parameters, in addition to the actual measurement results. This article describes the automatic creation of laboratory book entries directly from the measurement programme using the PTB's MTF measuring system as an example.

1 Introduction

The safeguarding of measurement data is a central component of good research practice. Reliable and traceable documentation ensures that experimental results remain verifiable and reproducible (cf. [1], Guideline 12: Documentation). Appropriate tools and processes are required to store and make available not only the actual measurement results, but also metadata such as environmental conditions and measurement parameters.

By integrating an electronic laboratory notebook (ELN) into the control system of a monitoring station, the quality of data storage and documentation can be significantly improved. An electronic lab notebook promotes traceability of test procedures by seamlessly linking measurement results, parameters and metadata. ELNs usually also support the FAIR data principles (**F**indability, **A**ccessibility, **I**nteroperability, and **R**eusability [2]), for example by assigning metadata, tags and persistent identifiers.

After thorough testing of several ELN software candidates, PTB has chosen eLabFTW for this purpose [3]. The software is open source and can be installed locally to address privacy concerns. It is a powerful tool for storing experimental and metadata, as well as information about the resources used in the experiment. In addition, other information such as data sheets, drivers or user manuals can be stored.

A software-based solution has many advantages over a paper-based laboratory book at the measuring system: the electronic laboratory book is available from anywhere, the design is responsive so that it can be used with mobile devices, too. It also offers an extensive search function, as well as a powerful tagging system, which enables users to quickly locate previous experiments, protocols, or inventory items. Templates and custom fields can be defined to match specific workflow needs, helping to standardize documentation practices. It also offers options for setting access rights so that entries can be edited jointly in a team.

A Python package is available [4] that enables the user to access the lab notebook using the eLabFTW API v2 (a package for API v1 exists, but is deprecated). This contribution shows how the automated creation of lab book entries can be carried out by a measurement programme. An exemplary Python script, available as a Gitlab project [5] simulates the behaviour of the measurement programme at PTB's MTF measuring system.

2 Automatic creation of lab notebook records

The automatic generation of ELN entries is demonstrated using the implementation in the control system of the MTF measuring system at PTB [6]. In this setup, the MTF of a test object can be determined at the focal point (position on the optical axis with the highest MTF value), during the measurement along the optical axis ("on-axis scan"), as well as in the image field of the test object ("off-axis scans"). The latter measurements, in particular, usually take several hours, so it is useful to create the ELN entry at the start of a measurement and to mark it accordingly on successful completion. This is described below using extracts from the control programme:

2.1 Initialisation

The first step is to define and establish a connection to the lab notebook. This requires an API key, which can be generated in the ELN user settings. The key is stored in a JSON file that is only accessible to the user and is read from there by the application.

```
API_HOST_URL = '<ELN_URL>'
with open(os.path.join('<path>','key.json')) as
     keyfile:
    API_KEY = json.load(keyfile)['key']
```

Using the `elabapi_python` package, an API instance is created and configured:

```
config = elabapi_python.Configuration()
config.api_key['api_key'] = API_KEY
config.api_key_prefix['api_key'] = 'Authorization'
config.host = API_HOST_URL

self.api_client = elabapi_python.ApiClient(config)
```

DGaO Proceedings 2025 – *http://www.dgao-proceedings.de* – ISSN: 1614-8436 – urn:nbn:de:0287-2025-P036
submitted: 02.07.2025 – published: 30.09.2025

This client then provides several functions for the different tasks for the communication with the ELN:

```
# create and edit ELN experiments:
experimentsApi =
    elabapi_python.ExperimentsApi(self.api_client)
# add tags to ELN experiments:
tagsApi = elabapi_python.TagsApi(self.api_client)
# ... and some more API functions to use templates
    and add ressources (e.g. devices used in the
    measurements), links, files to the ELN entries
...
```

The actual ELN entry then is created using

```
response =
    experimentsApi.post_experiment_with_http_info()
```

In the response to that command the unique ID of the ELN entry is included, thus this command should be preferred over the `post_experiment` command also suitable for creating ELN entries.

2.2 Collect and store the metadata

The measurement program then collects some information, the experimental setup data used for the measurement (e.g. field angle, scan length, step width, ...), and metadata such as the description of the SUT and it's orientation in the measurement setup, and stores them in dictionaries `exp_base_dict` and `exp_metadata_dict`:

```
# main text describing the measurement (usually empty
    for the automatically created ELN entry:
main_text = "<add additional infos here>"
exp_base_dict = {"body": f"<H1>Main text
    header</H1>{main_text}",
    "category": 2, "status": 90, ...}
exp_metadata_dict = {
    # define extra field groups
    "elabftw": {
        "display_main_text": True,
        "extra_fields_groups":[
            {"id":1, "name": "General information"},
            {"id":2, "name": "Experimental setup"},
            {"id":3, "name": "..."},]},

    "extra_fields":{ #define and fill the extra fields
        # General information
        "Measurement ID": {
            "type": "text", "value": f"{<ID
    variable>:d}",
            "group_id":1, "position":0,
    "description":"<description>"},
        # Experimental setup
        "SUT": {"type": "text", "value": str(<SUT
    variable>),
                "group_id": 2, "position": 0}, ...
        # Measurement parameters
        "Scan length": {"type": "number", "unit":
    "mm", "value": f"{<scan length variable>}",
            "group_id": 3, "position": 0}, ...
```

The metadata dictionary must be converted to a JSON string before adding it to the `exp_base_dict` dictionary. The base dictionary then is transferred to the electronic lab notebook:

```
exp_metadata = json.dumps(exp_metadata_dict)
# add to base dictionary
exp_base_dict['title'] = '<title string>'
exp_base_dict['metadata'] = exp_metadata

experimentsApi.patch_experiment(self.exp_id,
    body=exp_base_dict)
```

During the measurement, updates can be automatically written to the lab book entry or intermediate states of the data files can be uploaded if necessary.

Finally, when the measurement has been successfully completed, the `store_to_elabftw()` function is called with an `update_only` option to set the correct status and add data files and plots if available:

```
experimentsApi.patch_experiment(self.exp_id,
    body={'status': 91}) # '91' is "successful"
# look for uploadable files
...
uploadable_files = [os.path.join(upload_files_path,f)
    for f in os.listdir(upload_files_path) if
    upload_filetype in f]
if len(uploadable_files)>0:
    for i in range(len(uploadable_files)):
        uploadsApi.post_upload('experiments',
    self.exp_id, file = uploadable_files[i])
        tagsApi.post_tag('experiments', self.exp_id,
    body={'tag': 'data file'})
# ... the same for plot, omitted here
```

3 Conclusion and outlook

This article presents the automated creation of lab notebook entries by the control program of the MTF measurement setup at PTB to conveniently and reliably store measurement metadata. For measurement setups that are not connected to a network and therefore do not have access to an electronic laboratory notebook, an offline method for storing the metadata is required. The ELN file format is widely recognised and can be imported into a variety of ELN solutions. This file format is actually a ZIP archive of a directory structure following the ResearchObject Crate specification [7], including a JSON file describing the contents and file locations. Initial attempts to create offline lab notebook records were successful, but did not include all of the features provided by the ELN software.

References

[1] Deutsche Forschungsgemeinschaft (DFG), "Guidelines for Safeguarding Good Research Practice," Accessed: 2025-05-09, URL https://doi.org/10.5281/zenodo.3923601.

[2] M. D. Wilkinson *et al.*, "The FAIR Guiding Principles for scientific data management and stewardship," Scientific Data **3**(1), 160018 (2016). URL https://doi.org/10.1038/sdata.2016.18.

[3] Deltablot, "eLabFTW website," Accessed: 2025-05-09, URL https://elabftw.net.

[4] "eLabFTW API v2 package in the Python Package Index," Accessed: 2025-05-19, URL https://pypi.org/project/elabapi-python/.

[5] H. Dierke, "eLabFTW automatisation GitLab project," Accessed: 2025-05-09, URL https://gitlab1.ptb.de/hdierke/elabftw-python-beispiel.

[6] H. Dierke and M. Schake, "Bestimmung des Fokuspunkts bei Messung der Modulationstransferfunktion MTF," in *DGaO-Proceedings 2023*, 124 B43 (DGaO, 2023). URL urn:nbn:de:0287-2023-B043-4.

[7] P. Sefton *et al.*, *"Research Object Crate" specification for aggregating and describing research data with associated metadata* (2023). URL https://doi.org/10.5281/zenodo.7867028.