how_to_use_api_functions

October 15, 2024

1 How to use nomad-utility-workflows to perform NOMAD API Calls

Imports for the following examples:

[1]:
```python
import time
from pprint import pprint

from decouple import config as environ

from nomad_utility_workflows.utils.core import get_authentication_token
from nomad_utility_workflows.utils.datasets import (
    create_dataset,
    delete_dataset,
    get_dataset_by_id,
    retrieve_datasets,
)
from nomad_utility_workflows.utils.entries import (
    download_entry_by_id,
    get_entries_of_my_uploads,
    get_entries_of_upload,
    get_entry_by_id,
    query_entries,
)
from nomad_utility_workflows.utils.uploads import (
    delete_upload,
    edit_upload_metadata,
    get_all_my_uploads,
    get_upload_by_id,
    publish_upload,
    upload_files_to_nomad,
)
from nomad_utility_workflows.utils.users import (
    get_user_by_id,
    search_users_by_name,
    who_am_i,
)
```

1.1 NOMAD URLs

The NOMAD URL specifies the base address of the API for the NOMAD deployment of interest. Typically, this URL is structured as `https://<deployment_base_path>/api/v1`.

By default, nomad-utility-workflows uses the Test deployment of NOMAD to make API calls. This is simply a safety mechanism so that users do not accidentally publish something during testing.

All API functions allow the user to specify the URL with the optional keyword argument `url`. If you want to use the central NOMAD URLs, you can simply set `url` equal to "prod", "staging", or "test", which correspond to the following deployments (see full URLs below):

- prod: the official NOMAD deployment.
  - Updated most infrequently (as advertised in #software-updates on the NOMAD Discord Server)
- staging: the beta version of NOMAD.
  - Updated more frequently than prod in order to integrate and test new features.
- test: a test NOMAD deployment.
  - The data is occassionally wiped, such that test publishing can be made.

Note that the prod and staging deployments share a common database, and that publishing on either will result in publically available data.

Alternatively to these short names, the user can use the `url` input to specify the full API address to some alternative NOMAD deployment, e.g., an Oasis.

For reference, the full addresses of the above-mentioned central NOMAD deployments (including api suffix) are:

[2]:
```python
from nomad_utility_workflows.utils.core import (
    NOMAD_PROD_URL,
    NOMAD_STAGING_URL,
    NOMAD_TEST_URL,
)

print(NOMAD_PROD_URL, NOMAD_STAGING_URL, NOMAD_TEST_URL)
```

```
https://nomad-lab.eu/prod/v1/api/v1 https://nomad-lab.eu/prod/v1/staging/api/v1
https://nomad-lab.eu/prod/v1/test/api/v1
```

1.2 Authentication

Some API calls, e.g., making uploads or accessing your own non-published uploads, require an authentication token. To generate this token, nomad-utility-workflows expects that your NOMAD credentials are stored in a `.env` file in the plugin root directory in the format:

```
NOMAD_USERNAME="<your_nomad_username>"
NOMAD_PASSWORD="<your_nomad_password>"
```

You can access these explicitly with:

[3]:
```
NOMAD_USERNAME = environ('NOMAD_USERNAME')
NOMAD_PASSWORD = environ('NOMAD_PASSWORD')
NOMAD_USERNAME
```

[3]:
```
'JFRudzinski'
```

Use `get_authentication_token()` with your credentials to explicitly obtain and store a token:

[ ]:
```
token = get_authentication_token(
    username=NOMAD_USERNAME, password=NOMAD_PASSWORD, url='test'
)
token
```

In practice, you do not need to obtain a token yourself when using nomad-utility-workflows. A token will automatically be obtained for API calls that require authentication. However, you may want to do the token generation yourself for custom API calls (see `Writing your own wrappers` below.)

1.2.1 NOMAD User Metadata

nomad-utility-workflows uses the `NomadUser()` class to store the following user metadata:

```
class NomadUser:
    user_id: str
    name: str
    first_name: str
    last_name: str
    username: str
    affiliation: str
    affiliation_address: str
    email: str
    is_oasis_admin: bool
    is_admin: bool
    repo_user_id: str
    created: dt.datetime
```

You can retrieve your own personal info with the `who_am_i()` function:

[4]:
```
nomad_user_me = who_am_i(url='test')
nomad_user_me
```

[4]:
```
NomadUser(name='Joseph Rudzinski')
```

Similarly, you can query NOMAD for other users with `search_users_by_name()`:

[5]:
```
nomad_users = search_users_by_name('Rudzinski', url='test')
nomad_users
```

[5]:
```
[NomadUser(name='Joseph Rudzinski'), NomadUser(name='Joseph Rudzinski')]
```

In the case of multiple matches or for robustly identifying particular users, e.g., coauthors, in the future, it may be useful to store their `user_id`—a persistent identifier for each user account. Then, in the future you can use `get_user_by_id()` to grab the user info:

```
[6]: nomad_user = get_user_by_id(nomad_users[0].user_id, url='test')
     nomad_user
```

```
[6]: NomadUser(name='Joseph Rudzinski')
```

1.2.2 Uploading Data

nomad-utility-workflows uses the `NomadUpload()` class to store the following upload metadata:

```
class NomadUpload:
    upload_id: str
    upload_create_time: dt.datetime
    main_author: NomadUser
    process_running: bool
    current_process: str
    process_status: str
    last_status_message: str
    errors: list[Any]
    warnings: list[Any]
    coauthors: list[str]
    coauthor_groups: list[Any]
    reviewers: list[NomadUser]
    reviewer_groups: list[Any]
    writers: list[NomadUser]
    writer_groups: list[Any]
    viewers: list[NomadUser]
    viewer_groups: list[Any]
    published: bool
    published_to: list[Any]
    with_embargo: bool
    embargo_length: float
    license: str
    entries: int
    n_entries: int
    upload_files_server_path: str
    publish_time: dt.datetime
    references: list[str]
    datasets: list[str]
    external_db: str
    upload_name: str
    comment: str
    url: str
    complete_time: dt.datetime
```

You can make an upload using the `upload_files_to_nomad()` function with input filename=<path_to_a_zip_file_with_your_upload_data>, as follows:

```
[7]: test_upload_fnm = './test.zip' # a dummy upload file containing a single empty␣
     ↪json file
```

```
[8]: upload_id = upload_files_to_nomad(filename=test_upload_fnm, url='test')
     upload_id
```

```
[8]: 'RdA_3ZsOTMqbtAhYLivVsw'
```

1.2.3 Checking the upload status

The returned upload_id can then be used to directly access the upload, e.g., to check the upload status, using get_upload_by_id():

```
[9]: nomad_upload = get_upload_by_id(upload_id, url='test')

     pprint(nomad_upload)
```

```
NomadUpload(upload_id='RdA_3ZsOTMqbtAhYLivVsw',
            upload_create_time=datetime.datetime(2024, 10, 15, 20, 2, 10,
378000),
            main_author=NomadUser(name='Joseph Rudzinski'),
            process_running=False,
            current_process='process_upload',
            process_status='SUCCESS',
            last_status_message='Process process_upload completed successfully',
            errors=[],
            warnings=[],
            coauthors=[],
            coauthor_groups=[],
            reviewers=[],
            reviewer_groups=[],
            writers=[NomadUser(name='Joseph Rudzinski')],
            writer_groups=[],
            viewers=[NomadUser(name='Joseph Rudzinski')],
            viewer_groups=[],
            published=False,
            published_to=[],
            with_embargo=False,
            embargo_length=0.0,
            license='CC BY 4.0',
            entries=1,
            n_entries=None,
upload_files_server_path='/nomad/test/fs/staging/R/RdA_3ZsOTMqbtAhYLivVsw',
            publish_time=None,
            references=None,
            datasets=None,
            external_db=None,
            upload_name=None,
            comment=None,
            url='https://nomad-lab.eu/prod/v1/test/api/v1',
            complete_time=datetime.datetime(2024, 10, 15, 20, 2, 11, 320000))
```

One common usage of this function is to ensure that an upload has been processed successfully before making a subsequent action on it, e.g., editing the metadata or publishing. For this purpose, one could require the `process_running==False` or `process_status='SUCCESS'`, e.g.:

```
    import time

    max_wait_time = 20 * 60  # 20 minutes in seconds
    interval = 2 * 60  # 2 minutes in seconds
    elapsed_time = 0

    while elapsed_time < max_wait_time:
        nomad_upload = get_upload_by_id(upload_id, url='test')

        # Check if the upload is complete
        if nomad_upload.process_status == 'SUCCESS':
            break

        # Wait for 2 minutes before the next call
        time.sleep(interval)
        elapsed_time += interval
    else:
        raise TimeoutError("Maximum wait time of 20 minutes exceeded. Upload is not complete."
```

1.2.4 Editing the upload metadata

After your upload is processed successfully, you can add coauthors, references, and other comments, as well as link to a dataset and provide a name for the upload. Note that the coauthor is specified by an email address that should correspond to the email linked to the person's NOMAD account, which can be accessed from `NomadUser.email`. The metadata should be stored as a dictionary as follows:

```
metadata = {
    "metadata": {
    "upload_name": '<new_upload_name>',
    "references": ["https://doi.org/xx.xxxx/xxxxxx"],
    "datasets": '<dataset_id>',
    "embargo_length": 0,
    "coauthors": ["coauthor@affiliation.de"],
    "comment": 'This is a test upload...'
    },
}
```

For example:

```
[10]: metadata_new = {'upload_name': 'Test Upload', 'comment': 'This is a test upload.
       ↪..'}
      edit_upload_metadata(upload_id, url='test', **metadata_new)
```

```
[10]: {'upload_id': 'RdA_3ZsOTMqbtAhYLivVsw',
       'data': {'process_running': True,
        'current_process': 'edit_upload_metadata',
        'process_status': 'PENDING',
        'last_status_message': 'Pending: edit_upload_metadata',
        'errors': [],
        'warnings': [],
        'complete_time': '2024-10-15T20:02:11.320000',
        'upload_id': 'RdA_3ZsOTMqbtAhYLivVsw',
        'upload_create_time': '2024-10-15T20:02:10.378000',
        'main_author': '8f052e1f-1906-41fd-b2eb-690c03407788',
        'coauthors': [],
        'coauthor_groups': [],
        'reviewers': [],
        'reviewer_groups': [],
        'writers': ['8f052e1f-1906-41fd-b2eb-690c03407788'],
        'writer_groups': [],
        'viewers': ['8f052e1f-1906-41fd-b2eb-690c03407788'],
        'viewer_groups': [],
        'published': False,
        'published_to': [],
        'with_embargo': False,
        'embargo_length': 0,
        'license': 'CC BY 4.0',
        'entries': 1,
        'upload_files_server_path':
      '/nomad/test/fs/staging/R/RdA_3ZsOTMqbtAhYLivVsw'}}
```

Before moving on, let's again check that this additional process is complete:

```
[11]: nomad_upload = get_upload_by_id(upload_id, url='test')

      pprint(nomad_upload.process_status == 'SUCCESS')
      pprint(nomad_upload.process_running is False)
```

```
True
True
```

1.2.5 Accessing individual entries of an upload

During the upload process, NOMAD automatically identfies representative files that indicate the presence of data that can be parsed with the plugins included within a given deployment. This means that each upload can contain multiple *entries*—the fundamental unit storage within the NOMAD database.

You can query the individual entries within a known upload with `get_entries_of_upload()`, which then returns the metadata within the `NomadEntry()` class of nomad-utility-worklfows:

```
class NomadEntry:
    entry_id: str
    upload_id: str
    references: list[str]
    origin: str
    quantities: list[str]
    datasets: list[NomadDataset]
    n_quantities: int
    nomad_version: str
    upload_create_time: dt.datetime
    nomad_commit: str
    section_defs: list[NomadSectionDefinition]
    processing_errors: list[Any]
    results: dict
    entry_name: str
    last_processing_time: dt.datetime
    parser_name: str
    calc_id: str
    published: bool
    writers: list[NomadUser]
    sections: list[str]
    processed: bool
    mainfile: str
    main_author: NomadUser
    viewers: list[NomadUser]
    entry_create_time: dt.datetime
    with_embargo: bool
    files: list[str]
    entry_type: str
    authors: list[NomadUser]
    license: str
    domain: str
    optimade: dict
    comment: str
    upload_name: str
    viewer_groups: list[Any]
    writer_groups: list[Any]
    text_search_contents: list[str]
    publish_time: dt.datetime
    entry_references: list[dict]
    url: str
```

Let's try this out with our test upload. In this case, the upload is *not* published and located in the *private* `Your Uploads` section of the NOMAD deployment. To access the uploads there, we need to set `with_authentication=True`:

```
[12]: entries = get_entries_of_upload(upload_id, url='test', with_authentication=True)
      pprint(f'Entries within upload_id={upload_id}:')
      for entry in entries:
          pprint(f'entry_id={entry.entry_id}')
```

```
'Entries within upload_id=RdA_3ZsOTMqbtAhYLivVsw:'
'entry_id=Htbl78lHDSNAKbvPjEgEN_6sOcxF'
```

To query an entry directly using the `entry_id`, use `get_entry_by_id()`:

```
[13]: entry = get_entry_by_id(entries[0].entry_id, url='test',␣
      ↪with_authentication=True)
      entry
```

```
[13]: NomadEntry(entry_id='Htbl78lHDSNAKbvPjEgEN_6sOcxF',
      upload_id='RdA_3ZsOTMqbtAhYLivVsw', references=[], origin='Joseph Rudzinski',
      n_quantities=0, nomad_version='1.3.7.dev55+ge83de27b3',
      upload_create_time=datetime.datetime(2024, 10, 15, 20, 2, 10, 378000,
      tzinfo=datetime.timezone(datetime.timedelta(0), '+0000')), nomad_commit='',
      processing_errors=[], entry_name='test.archive.json',
      last_processing_time=datetime.datetime(2024, 10, 15, 20, 2, 10, 752000,
      tzinfo=datetime.timezone(datetime.timedelta(0), '+0000')),
      parser_name='parsers/archive', calc_id='Htbl78lHDSNAKbvPjEgEN_6sOcxF',
      published=False, writers=[NomadUser(name='Joseph Rudzinski')], processed=True,
      mainfile='test.archive.json', main_author=NomadUser(name='Joseph Rudzinski'),
      entry_create_time=datetime.datetime(2024, 10, 15, 20, 2, 10, 543000,
      tzinfo=datetime.timezone(datetime.timedelta(0), '+0000')), with_embargo=False,
      entry_type=None, license='CC BY 4.0', domain=None, comment='This is a test
      upload…', upload_name='Test Upload', text_search_contents=[],
      publish_time=None, entry_references=None, url='https://nomad-
      lab.eu/prod/v1/test/api/v1')
```

You can download the full (meta)data stored in an entry using `download_entry_by_id()`. This will return the entire archive as a dictionary. If you supply a `zip_file_name` (including the desired local path), the raw data of the entry will also be downloaded and saved to a zip file. Otherwise, only the archive will be downloaded.

```
[15]: test = download_entry_by_id(
          entry.entry_id, url='test', zip_file_name='./raw_entry_data.zip',␣
      ↪with_authentication=True
      )
      test
```

```
[15]: {'processing_logs': [{'event': 'Executing celery task',
         'proc': 'Entry',
         'process': 'process_entry',
         'process_worker_id': 'BOiybXorRqW5XImFf0SyoA',
         'parser': 'parsers/archive',
 'logger': 'nomad.processing',
 'timestamp': '2024-10-15 20:02.10',
 'level': 'DEBUG'},
{'exec_time': '0.001298666000366211',
 'input_size': '3',
 'event': 'parser executed',
 'proc': 'Entry',
 'process': 'process_entry',
 'process_worker_id': 'BOiybXorRqW5XImFf0SyoA',
 'parser': 'parsers/archive',
 'step': 'parsers/archive',
 'logger': 'nomad.processing',
 'timestamp': '2024-10-15 20:02.10',
 'level': 'INFO'},
{'normalizer': 'MetainfoNormalizer',
 'step': 'MetainfoNormalizer',
 'event': 'normalizer completed successfully',
 'proc': 'Entry',
 'process': 'process_entry',
 'process_worker_id': 'BOiybXorRqW5XImFf0SyoA',
 'parser': 'parsers/archive',
 'logger': 'nomad.processing',
 'timestamp': '2024-10-15 20:02.10',
 'level': 'INFO'},
{'exec_time': '0.0011508464813232422',
 'input_size': '3',
 'event': 'normalizer executed',
 'proc': 'Entry',
 'process': 'process_entry',
 'process_worker_id': 'BOiybXorRqW5XImFf0SyoA',
 'parser': 'parsers/archive',
 'normalizer': 'MetainfoNormalizer',
 'step': 'MetainfoNormalizer',
 'logger': 'nomad.processing',
 'timestamp': '2024-10-15 20:02.10',
 'level': 'INFO'},
{'normalizer': 'ResultsNormalizer',
 'step': 'ResultsNormalizer',
 'event': 'normalizer completed successfully',
 'proc': 'Entry',
 'process': 'process_entry',
 'process_worker_id': 'BOiybXorRqW5XImFf0SyoA',
 'parser': 'parsers/archive',
 'logger': 'nomad.processing',
 'timestamp': '2024-10-15 20:02.10',
 'level': 'INFO'},
{'exec_time': '0.0011796951293945312',
  'input_size': '3',
  'event': 'normalizer executed',
  'proc': 'Entry',
  'process': 'process_entry',
  'process_worker_id': 'BOiybXorRqW5XImFfOSyoA',
  'parser': 'parsers/archive',
  'normalizer': 'ResultsNormalizer',
  'step': 'ResultsNormalizer',
  'logger': 'nomad.processing',
  'timestamp': '2024-10-15 20:02.10',
  'level': 'INFO'},
 {'exec_time': '0.002213716506958008',
  'event': 'entry metadata saved',
  'proc': 'Entry',
  'process': 'process_entry',
  'process_worker_id': 'BOiybXorRqW5XImFfOSyoA',
  'parser': 'parsers/archive',
  'logger': 'nomad.processing',
  'timestamp': '2024-10-15 20:02.10',
  'level': 'INFO'},
 {'exec_time': '0.07823586463928223',
  'event': 'entry metadata indexed',
  'proc': 'Entry',
  'process': 'process_entry',
  'process_worker_id': 'BOiybXorRqW5XImFfOSyoA',
  'parser': 'parsers/archive',
  'logger': 'nomad.processing',
  'timestamp': '2024-10-15 20:02.10',
  'level': 'INFO'}],
'metadata': {'upload_id': 'RdA_3ZsOTMqbtAhYLivVsw',
 'upload_create_time': '2024-10-15T20:02:10.378000+00:00',
 'entry_id': 'Htbl78lHDSNAKbvPjEgEN_6sOcxF',
 'entry_name': 'test.archive.json',
 'entry_hash': 't6Zf68GLfrWxWRAIQu7QAY8LVmlL',
 'entry_create_time': '2024-10-15T20:02:10.543000+00:00',
 'parser_name': 'parsers/archive',
 'mainfile': 'test.archive.json',
 'text_search_contents': [],
 'files': ['test.archive.json'],
 'published': False,
 'with_embargo': False,
 'embargo_length': 0,
 'license': 'CC BY 4.0',
 'processed': True,
 'last_processing_time': '2024-10-15T20:02:10.752287+00:00',
 'processing_errors': [],
 'nomad_version': '1.3.7.dev55+ge83de27b3',
'nomad_commit': '',
'references': [],
'main_author': '8f052e1f-1906-41fd-b2eb-690c03407788',
'coauthors': [],
'coauthor_groups': [],
'entry_coauthors': [],
'reviewers': [],
'reviewer_groups': [],
'datasets': [],
'n_quantities': 34,
'quantities': ['',
 'metadata',
 'metadata.coauthor_groups',
 'metadata.coauthors',
 'metadata.datasets',
 'metadata.embargo_length',
 'metadata.entry_coauthors',
 'metadata.entry_create_time',
 'metadata.entry_hash',
 'metadata.entry_id',
 'metadata.entry_name',
 'metadata.entry_timestamp',
 'metadata.entry_timestamp.timestamp',
 'metadata.entry_timestamp.token',
 'metadata.entry_timestamp.token_seed',
 'metadata.entry_timestamp.tsa_server',
 'metadata.files',
 'metadata.last_processing_time',
 'metadata.license',
 'metadata.main_author',
 'metadata.mainfile',
 'metadata.nomad_commit',
 'metadata.nomad_version',
 'metadata.parser_name',
 'metadata.processed',
 'metadata.processing_errors',
 'metadata.published',
 'metadata.quantities',
 'metadata.references',
 'metadata.reviewer_groups',
 'metadata.reviewers',
 'metadata.section_defs',
 'metadata.section_defs.definition_id',
 'metadata.section_defs.definition_qualified_name',
 'metadata.section_defs.used_directly',
 'metadata.sections',
 'metadata.upload_create_time',
   'metadata.upload_id',
   'metadata.with_embargo',
   'results',
   'results.properties'],
  'sections': ['nomad.datamodel.datamodel.EntryArchive',
   'nomad.datamodel.datamodel.EntryMetadata',
   'nomad.datamodel.datamodel.RFC3161Timestamp',
   'nomad.datamodel.results.Properties',
   'nomad.datamodel.results.Results'],
  'entry_timestamp': {'token_seed': 't6Zf68GLfrWxWRAIQu7QAY8LVmlL',
   'token': 'MIIEQwYJKoZIhvcNAQcCoIIENDCCBDACAQMxDTALBglghkgBZQMEAgEwfAYLKoZIhvc
NAQkQAQSgbQRrMGkCAQEGDCsGAQQBga0hgiwWATAvMAsGCWCGSAFlAwQCAQQgYnRB2tk2mTRtMamyedr
2QFd3bb0lFM56N52xD8rv/0ECFFIumicGBkskyLovIs3OBywh3EXbGA8yMDI0MTAxNTIwMDIxMFoxgg0
cMIIDmAIBATCBnjCBjTELMAkGA1UEBhMCREUxRTBDBgNVBAoMPFZlcmVpbiB6dXIgRm9lcmRlcnVuZyB
laW5lcyBEZXV0c2NoZW4gRm9yc2NodW5nc25ldHplcyBlLiBWLjEQMA4GA1UECwwHREZOLVBLSTElMCM
GA1UEAwwcREZOLVZlcmVpbiBHbG9iYWwgSXNzdWluZyBDQQIMKQLVczMPeOL0nrS5MAsGCWCGSAFlAwQ
CAaCB0TAaBgkqhkiG9w0BCQMxDQYLKoZIhvcNAQkQAQQwHAYJKoZIhvcNAQkFMQ8XDTI0MTAxNTIwMDI
xMFowKwYJKoZIhvcNAQk0MR4wHDALBglghkgBZQMEAgGhDQYJKoZIhvcNAQELBQAwLwYJKoZIhvcNAQk
EMSIEIMF/vhL+ddm6reCVHQgPz5FLJYZKE3ag+HqzfGTfwTvPMDcGCyqGSIb3DQEJEAIvMSgwJjAkMCI
EILYIjb3dCJjTSQeNfCMyp07MhBQMoINZ8CNXJUbPboLkMA0GCSqGSIb3DQEBCwUABIICAJSu4GSAVDG
NwuA+Kr5Qhi7rrcQcZpAcA2TOotKVS9b8wDyCE+J7IwobbVDIVURsa0b9QsReNUZHc+U9TmlWGprwY1j
1BVy+ccXNg2U2Uf0dMrnOzVWfPNAoMT1iv9tK3kMwq2Gal35yh/Arrp+XYMmLfKFRZpzNcjz0TFokjCF
brPxreLCHgSnPWy3VHncNWghyMg6Nxq0rUwvV7dp7cjt1R9Ky6eHdxQvyMmVuSRbTetg+B43KkrYXoqG
NGVEqiaScZWJswM5jFApNquUZnOuRl4/bhABNGWpLXQRxDz2r4Bqvgz4DfQt/8EPg7YvWNWNzfw+oPXg
h5dDqKW9DiKoa0U2J1+/YKnBcdJefDsyZHnOcXOAIaX/f8k9Wg90v+c/WrbSjbfYMMhJBGqKYgU6DAeh
7p7DZQNeDS4qspVWaTb10zgxmnkCbtqzmlzJvY4pOnjirVHfMFyXxL1rkXG91swhjK6FnL/okQKVT2tG
QHC4I15VnEmrhit/ptXJHpWABO6TKJCW4oCgUdxUBPMT3bY7RaT3Fe+XzjjRuau0dVd2bYIbUTlrGFBU
jJf+9Zuj145FdqdSjezx7NaIy7zsF1wV/e9feu6vbu3kEu32hZOT7Agw/Ryqrqgx4KuJuel0lwTljeQV
HUfblr6yhwNDzdxufsQyAbJAvRUS2tyMk',
   'tsa_server': 'http://zeitstempel.dfn.de',
   'timestamp': '2024-10-15T20:02:10+00:00'},
  'section_defs': [{'definition_qualified_name':
'nomad.datamodel.data.ArchiveSection',
    'definition_id': '7047cbff9980abff17cce4b1b6b0d1c783505b7f',
    'used_directly': True},
   {'definition_qualified_name': 'nomad.datamodel.datamodel.EntryArchive',
    'definition_id': '510c3beb8699d7d23a29bb0cf45540286916c20c',
    'used_directly': True},
   {'definition_qualified_name': 'nomad.datamodel.datamodel.EntryMetadata',
    'definition_id': '6edfa503af63b84d6a6021c227d00137b4c1cc9c',
    'used_directly': True},
   {'definition_qualified_name': 'nomad.datamodel.datamodel.RFC3161Timestamp',
    'definition_id': '1e3e9dd7b802b04343f46305a7d0f58663d8110a',
    'used_directly': True},
   {'definition_qualified_name': 'nomad.datamodel.results.Properties',
    'definition_id': '3d0188853e1806435f95f9a876b83ed98ad38713',
    'used_directly': True},
   {'definition_qualified_name': 'nomad.datamodel.results.Results',
    'definition_id': '1caea35fada02e0b6861ceec2dd928595fc824db',
    'used_directly': True}]},
 'results': {'properties': {}},
 'm_ref_archives': {}}
```

1.3 Publishing Uploads

Once the processing of your upload is successful and you have added/adjusted the appropriate metadata, you can publish your upload with `publish_upload()`, making it publicly available on the corresponding NOMAD deployment.

Note that once the upload is published you will no longer be able to make changes to the raw files that you uploaded. However, the upload metadata (accessed and edited in the above example) can be changed after publishing.

[16]:
```
published_upload = publish_upload(nomad_upload.upload_id, url='test')
published_upload
```

[16]:
```
{'upload_id': 'RdA_3ZsOTMqbtAhYLivVsw',
 'data': {'process_running': True,
  'current_process': 'publish_upload',
  'process_status': 'PENDING',
  'last_status_message': 'Pending: publish_upload',
  'errors': [],
  'warnings': [],
  'complete_time': '2024-10-15T20:03:51.605000',
  'upload_id': 'RdA_3ZsOTMqbtAhYLivVsw',
  'upload_name': 'Test Upload',
  'upload_create_time': '2024-10-15T20:02:10.378000',
  'main_author': '8f052e1f-1906-41fd-b2eb-690c03407788',
  'coauthors': [],
  'coauthor_groups': [],
  'reviewers': [],
  'reviewer_groups': [],
  'writers': ['8f052e1f-1906-41fd-b2eb-690c03407788'],
  'writer_groups': [],
  'viewers': ['8f052e1f-1906-41fd-b2eb-690c03407788'],
  'viewer_groups': [],
  'published': False,
  'published_to': [],
  'with_embargo': False,
  'embargo_length': 0,
  'license': 'CC BY 4.0',
  'entries': 1,
  'upload_files_server_path':
'/nomad/test/fs/staging/R/RdA_3ZsOTMqbtAhYLivVsw'}}
```

1.4 Finding and Creating Datasets

Although uploads can group multiple entries together, they are limited by the maximum upload size and act more as a practical tool for optimizing the transfer of data to the NOMAD repository. For scientifically relevant connections between entries, NOMAD uses *Datasets* and *Workflows*.

You can easily create a dataset with `create_dataset()`:

```
[18]: dataset_id = create_dataset('test dataset', url='test')
      dataset_id
```

```
[18]: 'NCKd75f9R9S8rnkd-GBZlg'
```

The returned `dataset_id` can then be used to add individual entries (or all entries within an upload) to the dataset by including it in the upload/entry metadata, using the method described above:

```
[19]: metadata_new = {'dataset_id': dataset_id}
      edit_upload_metadata(upload_id, url='test', **metadata_new)
```

```
[19]: {'upload_id': 'RdA_3ZsOTMqbtAhYLivVsw',
       'data': {'process_running': True,
        'current_process': 'edit_upload_metadata',
        'process_status': 'PENDING',
        'last_status_message': 'Pending: edit_upload_metadata',
        'errors': [],
        'warnings': [],
        'complete_time': '2024-10-15T20:09:28.769000',
        'upload_id': 'RdA_3ZsOTMqbtAhYLivVsw',
        'upload_name': 'Test Upload',
        'upload_create_time': '2024-10-15T20:02:10.378000',
        'main_author': '8f052e1f-1906-41fd-b2eb-690c03407788',
        'coauthors': [],
        'coauthor_groups': [],
        'reviewers': [],
        'reviewer_groups': [],
        'writers': ['8f052e1f-1906-41fd-b2eb-690c03407788'],
        'writer_groups': [],
        'viewers': ['8f052e1f-1906-41fd-b2eb-690c03407788'],
        'viewer_groups': [],
        'published': True,
        'published_to': [],
        'publish_time': '2024-10-15T20:09:28.757000',
        'with_embargo': False,
        'embargo_length': 0,
        'license': 'CC BY 4.0',
        'entries': 1}}
```

[20]:
```
nomad_upload = get_upload_by_id(upload_id, url='test')

pprint(nomad_upload.process_status == 'SUCCESS')
pprint(nomad_upload.process_running is False)
```

```
True
True
```

You can also retrieve the dataset metadata using the `dataset_id` with `get_dataset_by_id()`. The returned `NomadDataset()` class contains the following attributes:

```
class NomadDataset:
    dataset_id: str
    dataset_create_time: dt.datetime
    dataset_name: str
    dataset_type: str
    dataset_modified_time: dt.datetime
    user: NomadUser
    doi: str
    pid: int
    m_annotations: dict
```

[21]:
```
nomad_dataset = get_dataset_by_id(dataset_id, url='test')
nomad_dataset
```

[21]:
```
NomadDataset(dataset_id='NCKd75f9R9S8rnkd-GBZlg',
dataset_create_time=datetime.datetime(2024, 10, 15, 20, 10, 17, 568000),
dataset_name='test dataset', dataset_type='owned',
dataset_modified_time=datetime.datetime(2024, 10, 15, 20, 10, 17, 568000),
user=NomadUser(name='Joseph Rudzinski'), doi=None, pid=None, m_annotations=None)
```

Alternatively, you can search for datasets, e.g., by `user_id` or `dataset_name`, using `retrieve_datasets()`:

[22]:
```
my_datasets = retrieve_datasets(
    user_id=nomad_user_me.user_id, url='test', max_datasets=20
)
pprint(my_datasets)
```

```
[NomadDataset(dataset_id='NCKd75f9R9S8rnkd-GBZlg',
              dataset_create_time=datetime.datetime(2024, 10, 15, 20, 10, 17,
568000),
              dataset_name='test dataset',
              dataset_type='owned',
              dataset_modified_time=datetime.datetime(2024, 10, 15, 20, 10, 17,
568000),
              user=NomadUser(name='Joseph Rudzinski'),
              doi=None,
              pid=None,
                m_annotations=None)]
```

To get the list of entries contained within a dataset, use `query_entries()`:

[23]:
```python
dataset_entries = query_entries(dataset_id=dataset_id, url='test')
for entry in dataset_entries:
    pprint(f'entry_id={entry.entry_id}, upload_id={entry.upload_id}')
```

```
'entry_id=Htbl78lHDSNAKbvPjEgEN_6sOcxF, upload_id=RdA_3ZsOTMqbtAhYLivVsw'
```

There is no "publishing" action for datasets. Instead, when the dataset is complete (i.e., you are ready to lock the contents of the dataset), you can *assign a DOI*. There is currently no API action for this within nomad-utility-workflows. You must go to the GUI of the relevant deployment, go to `PUBLISH > Datasets`, find the dataset, and then click the "assign a DOI" banner icon to the right of the dataset entry.

1.5 Deleting Uploads and Datasets

You can delete uploads and datasets using `delete_upload()` and `delete_dataset()` as demonstrated in the following examples (along with the previously explained workflow of uploading, editing, etc.). Note that the wait times in these examples are arbitrary. One should optimize these for specific use cases.

upload, check for success, delete, check for success:

[25]:
```python
# Make a dummy upload
upload_id = upload_files_to_nomad(filename=test_upload_fnm, url='test')


max_wait_time = 15  # 15 seconds
interval = 5  # 5 seconds
elapsed_time = 0

while elapsed_time < max_wait_time:
    # Get the upload
    nomad_upload = get_upload_by_id(upload_id, url='test')

    # Check if the upload is complete
    if nomad_upload.process_status == 'SUCCESS':
        break

    # Wait for 5 seconds before the next call
    time.sleep(interval)
    elapsed_time += interval
else:
    raise TimeoutError(
        'Maximum wait time of 15 seconds exceeded. Upload is not complete.'
    )

# Delete the upload
```

```
delete_upload(upload_id, url='test')

# Wait for 5 seconds to make sure deletion is complete
time.sleep(5)

# Check if the upload was deleted
try:
    get_upload_by_id(upload_id, url='test')
except Exception:
    pprint(f'Upload with upload_id={upload_id} was deleted successfully.')
```

```
'Upload with upload_id=zpq-JTzWQJ63jtSOlbueKA was deleted successfully.'
```

create dataset, check for success, delete, check for success:

[26]:
```
# Make a dummy dataset
dataset_id = create_dataset('dummy dataset', url='test')

# Wait for 5 seconds to make sure dataset is created
time.sleep(5)

# Ensure the dataset was created
dummy_dataset = get_dataset_by_id(dataset_id, url='test')
assert dummy_dataset.dataset_id == dataset_id

# Delete the upload
delete_dataset(dataset_id, url='test')

# Wait for 5 seconds to make sure deletion is complete
time.sleep(5)

# Check if the dataset was deleted
try:
    get_dataset_by_id(dataset_id, url='test')
except Exception:
    pprint(f'Dataset with dataset_id={dataset_id} was deleted successfully.')
```

```
'Dataset with dataset_id=eT2WPkfCQgmwNadsurYz0A was deleted successfully.'
```

1.6 Useful Wrappers

nomad-utility-workflows contains a few useful wrapper functions to help users query all of their uploads and corresponding entries:

[27]:
```
get_all_my_uploads(url='test')
```

[27]:
```
[NomadUpload(upload_id='bQa5SGDQQ8auQUBb5AaYHw',
upload_create_time=datetime.datetime(2024, 10, 14, 10, 48, 40, 994000),
main_author=NomadUser(name='Joseph Rudzinski'), process_running=False,
current_process='publish_upload', process_status='SUCCESS',
last_status_message='Process publish_upload completed successfully', errors=[],
warnings=[], coauthors=[], coauthor_groups=[], reviewers=[], reviewer_groups=[],
writers=[NomadUser(name='Joseph Rudzinski')], writer_groups=[],
viewers=[NomadUser(name='Joseph Rudzinski')], viewer_groups=[], published=True,
published_to=[], with_embargo=False, embargo_length=0.0, license='CC BY 4.0',
entries=1, n_entries=None, upload_files_server_path=None,
publish_time=datetime.datetime(2024, 10, 14, 10, 48, 55, 806000),
references=None, datasets=None, external_db=None, upload_name='Test Upload',
comment=None, url='https://nomad-lab.eu/prod/v1/test/api/v1',
complete_time=datetime.datetime(2024, 10, 14, 10, 48, 55, 818000)),
 NomadUpload(upload_id='DN61X4r7SCyzm5q1kxcEcw',
upload_create_time=datetime.datetime(2024, 10, 14, 10, 55, 12, 410000),
main_author=NomadUser(name='Joseph Rudzinski'), process_running=False,
current_process='publish_upload', process_status='SUCCESS',
last_status_message='Process publish_upload completed successfully', errors=[],
warnings=[], coauthors=[], coauthor_groups=[], reviewers=[], reviewer_groups=[],
writers=[NomadUser(name='Joseph Rudzinski')], writer_groups=[],
viewers=[NomadUser(name='Joseph Rudzinski')], viewer_groups=[], published=True,
published_to=[], with_embargo=False, embargo_length=0.0, license='CC BY 4.0',
entries=1, n_entries=None, upload_files_server_path=None,
publish_time=datetime.datetime(2024, 10, 14, 10, 55, 23, 52000),
references=None, datasets=None, external_db=None, upload_name='Test Upload',
comment=None, url='https://nomad-lab.eu/prod/v1/test/api/v1',
complete_time=datetime.datetime(2024, 10, 14, 10, 55, 23, 65000)),
 NomadUpload(upload_id='z4QvhZ7qSCmgIFv_qJqlyQ',
upload_create_time=datetime.datetime(2024, 10, 14, 20, 20, 38, 757000),
main_author=NomadUser(name='Joseph Rudzinski'), process_running=False,
current_process='edit_upload_metadata', process_status='SUCCESS',
last_status_message='Process edit_upload_metadata completed successfully',
errors=[], warnings=[], coauthors=['7c85bdf1-8b53-40a8-81a4-04f26ff56f29'],
coauthor_groups=[], reviewers=[], reviewer_groups=[],
writers=[NomadUser(name='Joseph Rudzinski'), NomadUser(name='Joseph
Rudzinski')], writer_groups=[], viewers=[NomadUser(name='Joseph Rudzinski'),
NomadUser(name='Joseph Rudzinski')], viewer_groups=[], published=True,
published_to=[], with_embargo=False, embargo_length=0.0, license='CC BY 4.0',
entries=1, n_entries=None, upload_files_server_path=None,
publish_time=datetime.datetime(2024, 10, 15, 6, 18, 27, 700000),
references=None, datasets=None, external_db=None, upload_name='Test Upload',
comment=None, url='https://nomad-lab.eu/prod/v1/test/api/v1',
complete_time=datetime.datetime(2024, 10, 15, 6, 22, 33, 45000)),
 NomadUpload(upload_id='GJdVAOCxRVe-Cwo3qMz9Kg',
upload_create_time=datetime.datetime(2024, 10, 15, 10, 48, 44, 337000),
main_author=NomadUser(name='Joseph Rudzinski'), process_running=False,
current_process='edit_upload_metadata', process_status='SUCCESS',
last_status_message='Process edit_upload_metadata completed successfully',
errors=[], warnings=[], coauthors=[], coauthor_groups=[], reviewers=[],
reviewer_groups=[], writers=[NomadUser(name='Joseph Rudzinski')],
writer_groups=[], viewers=[NomadUser(name='Joseph Rudzinski')],
viewer_groups=[], published=True, published_to=[], with_embargo=False,
embargo_length=0.0, license='CC BY 4.0', entries=1, n_entries=None,
upload_files_server_path=None, publish_time=datetime.datetime(2024, 10, 15, 10,
49, 24, 4000), references=None, datasets=None, external_db=None,
upload_name='Test Upload', comment=None, url='https://nomad-
lab.eu/prod/v1/test/api/v1', complete_time=datetime.datetime(2024, 10, 15, 10,
49, 30, 962000)),
 NomadUpload(upload_id='RdA_3ZsOTMqbtAhYLivVsw',
upload_create_time=datetime.datetime(2024, 10, 15, 20, 2, 10, 378000),
main_author=NomadUser(name='Joseph Rudzinski'), process_running=False,
current_process='edit_upload_metadata', process_status='SUCCESS',
last_status_message='Process edit_upload_metadata completed successfully',
errors=[], warnings=[], coauthors=[], coauthor_groups=[], reviewers=[],
reviewer_groups=[], writers=[NomadUser(name='Joseph Rudzinski')],
writer_groups=[], viewers=[NomadUser(name='Joseph Rudzinski')],
viewer_groups=[], published=True, published_to=[], with_embargo=False,
embargo_length=0.0, license='CC BY 4.0', entries=1, n_entries=None,
upload_files_server_path=None, publish_time=datetime.datetime(2024, 10, 15, 20,
9, 28, 757000), references=None, datasets=None, external_db=None,
upload_name='Test Upload', comment=None, url='https://nomad-
lab.eu/prod/v1/test/api/v1', complete_time=datetime.datetime(2024, 10, 15, 20,
10, 33, 141000))]
```

[28]:
```
get_entries_of_my_uploads(url='test')
```

[28]:
```
[NomadEntry(entry_id='ycdeXhPDG-nIgEQlqBfzIEKPWCvy',
upload_id='bQa5SGDQQ8auQUBb5AaYHw', references=[], origin='Joseph Rudzinski',
n_quantities=34, nomad_version='1.3.7.dev55+ge83de27b3',
upload_create_time=datetime.datetime(2024, 10, 14, 10, 48, 40, 994000,
tzinfo=datetime.timezone(datetime.timedelta(0), '+0000')), nomad_commit='',
processing_errors=[], entry_name='test.archive.json',
last_processing_time=datetime.datetime(2024, 10, 14, 10, 48, 42, 415000,
tzinfo=datetime.timezone(datetime.timedelta(0), '+0000')),
parser_name='parsers/archive', calc_id='ycdeXhPDG-nIgEQlqBfzIEKPWCvy',
published=True, writers=[NomadUser(name='Joseph Rudzinski')], processed=True,
mainfile='test.archive.json', main_author=NomadUser(name='Joseph Rudzinski'),
entry_create_time=datetime.datetime(2024, 10, 14, 10, 48, 41, 672000,
tzinfo=datetime.timezone(datetime.timedelta(0), '+0000')), with_embargo=False,
entry_type=None, license='CC BY 4.0', domain=None, comment='This is a test
upload…', upload_name='Test Upload', text_search_contents=[],
publish_time=None, entry_references=None, url='https://nomad-
lab.eu/prod/v1/test/api/v1'),
 NomadEntry(entry_id='7A6lJb-14xR9lxXO8kjuYt5-vxg2',
upload_id='DN61X4r7SCyzm5q1kxcEcw', references=[], origin='Joseph Rudzinski',
n_quantities=34, nomad_version='1.3.7.dev55+ge83de27b3',
upload_create_time=datetime.datetime(2024, 10, 14, 10, 55, 12, 410000,
tzinfo=datetime.timezone(datetime.timedelta(0), '+0000')), nomad_commit='',
processing_errors=[], entry_name='test.archive.json',
last_processing_time=datetime.datetime(2024, 10, 14, 10, 55, 12, 808000,
tzinfo=datetime.timezone(datetime.timedelta(0), '+0000')),
parser_name='parsers/archive', calc_id='7A6lJb-14xR9lxXO8kjuYt5-vxg2',
published=True, writers=[NomadUser(name='Joseph Rudzinski')], processed=True,
mainfile='test.archive.json', main_author=NomadUser(name='Joseph Rudzinski'),
entry_create_time=datetime.datetime(2024, 10, 14, 10, 55, 12, 563000,
tzinfo=datetime.timezone(datetime.timedelta(0), '+0000')), with_embargo=False,
entry_type=None, license='CC BY 4.0', domain=None, comment='This is a test
upload…', upload_name='Test Upload', text_search_contents=[],
publish_time=None, entry_references=None, url='https://nomad-
lab.eu/prod/v1/test/api/v1'),
 NomadEntry(entry_id='jWSpYURP5GgPtgF9LXZJpNlDv-GL',
upload_id='z4QvhZ7qSCmgIFv_qJqlyQ', references=[], origin='Joseph Rudzinski',
n_quantities=0, nomad_version='1.3.7.dev55+ge83de27b3',
upload_create_time=datetime.datetime(2024, 10, 14, 20, 20, 38, 757000,
tzinfo=datetime.timezone(datetime.timedelta(0), '+0000')), nomad_commit='',
processing_errors=[], entry_name='test.archive.json',
last_processing_time=datetime.datetime(2024, 10, 14, 20, 20, 39, 272000,
tzinfo=datetime.timezone(datetime.timedelta(0), '+0000')),
parser_name='parsers/archive', calc_id='jWSpYURP5GgPtgF9LXZJpNlDv-GL',
published=True, writers=[NomadUser(name='Joseph Rudzinski'),
NomadUser(name='Joseph Rudzinski')], processed=True,
mainfile='test.archive.json', main_author=NomadUser(name='Joseph Rudzinski'),
entry_create_time=datetime.datetime(2024, 10, 14, 20, 20, 38, 982000,
tzinfo=datetime.timezone(datetime.timedelta(0), '+0000')), with_embargo=False,
entry_type=None, license='CC BY 4.0', domain=None, comment='This is a test
upload…edited', upload_name='Test Upload', text_search_contents=[],
publish_time=datetime.datetime(2024, 10, 15, 6, 18, 27, 700000,
tzinfo=datetime.timezone(datetime.timedelta(0), '+0000')),
entry_references=None, url='https://nomad-lab.eu/prod/v1/test/api/v1'),
 NomadEntry(entry_id='MVBIMEZOuIzH7-QFU2TtMIM6LLPp', upload_id='GJdVAOCxRVe-
Cwo3qMz9Kg', references=[], origin='Joseph Rudzinski', n_quantities=0,
nomad_version='1.3.7.dev55+ge83de27b3',
upload_create_time=datetime.datetime(2024, 10, 15, 10, 48, 44, 337000,
tzinfo=datetime.timezone(datetime.timedelta(0), '+0000')), nomad_commit='',
processing_errors=[], entry_name='test.archive.json',
last_processing_time=datetime.datetime(2024, 10, 15, 10, 48, 45, 206000,
tzinfo=datetime.timezone(datetime.timedelta(0), '+0000')),
parser_name='parsers/archive', calc_id='MVBIMEZOuIzH7-QFU2TtMIM6LLPp',
published=True, writers=[NomadUser(name='Joseph Rudzinski')], processed=True,
mainfile='test.archive.json', main_author=NomadUser(name='Joseph Rudzinski'),
entry_create_time=datetime.datetime(2024, 10, 15, 10, 48, 44, 741000,
tzinfo=datetime.timezone(datetime.timedelta(0), '+0000')), with_embargo=False,
entry_type=None, license='CC BY 4.0', domain=None, comment='This is a test
upload…', upload_name='Test Upload', text_search_contents=[],
publish_time=datetime.datetime(2024, 10, 15, 10, 49, 24, 4000,
tzinfo=datetime.timezone(datetime.timedelta(0), '+0000')),
entry_references=None, url='https://nomad-lab.eu/prod/v1/test/api/v1'),
 NomadEntry(entry_id='Htbl78lHDSNAKbvPjEgEN_6sOcxF',
upload_id='RdA_3ZsOTMqbtAhYLivVsw', references=[], origin='Joseph Rudzinski',
n_quantities=0, nomad_version='1.3.7.dev55+ge83de27b3',
upload_create_time=datetime.datetime(2024, 10, 15, 20, 2, 10, 378000,
tzinfo=datetime.timezone(datetime.timedelta(0), '+0000')), nomad_commit='',
processing_errors=[], entry_name='test.archive.json',
last_processing_time=datetime.datetime(2024, 10, 15, 20, 2, 10, 752000,
tzinfo=datetime.timezone(datetime.timedelta(0), '+0000')),
parser_name='parsers/archive', calc_id='Htbl78lHDSNAKbvPjEgEN_6sOcxF',
published=True, writers=[NomadUser(name='Joseph Rudzinski')], processed=True,
mainfile='test.archive.json', main_author=NomadUser(name='Joseph Rudzinski'),
entry_create_time=datetime.datetime(2024, 10, 15, 20, 2, 10, 543000,
tzinfo=datetime.timezone(datetime.timedelta(0), '+0000')), with_embargo=False,
entry_type=None, license='CC BY 4.0', domain=None, comment='This is a test
upload…', upload_name='Test Upload', text_search_contents=[],
publish_time=datetime.datetime(2024, 10, 15, 20, 9, 28, 757000,
tzinfo=datetime.timezone(datetime.timedelta(0), '+0000')),
entry_references=None, url='https://nomad-lab.eu/prod/v1/test/api/v1')]
```

1.6.1 Writing Your Own Wrappers

In `nomad_utility_workflows.utils.core` you will find the core NOMAD API functions `get_nomad_request()`, `post_nomad_request()`, and `delete_nomad_request()`. Using these as a basis, along with the NOMAD API Dashboard, you can easily extend the `nomad-utility-workflows` module for making more specific queries within your specialized workflows.