How to navigate the code

NOMAD is a complex project with lots of parts. This guide gives you a rough overview about the codebase and ideas about what to look at first.

Git Projects

There is one main NOMAD project (https://gitlab.mpcdf.mpg.de/nomad-lab/nomad-FAIR){:target="_blank"} (and its fork on GitHub (https://github.com/nomad-coe/nomad){:target="_blank"}). This project contains all the framework and infrastructure code. It instigates all checks, builds, and deployments for the public NOMAD service, the NOMAD Oasis, and the `nomad-lab` Python package. All contributions to NOMAD have to go through this project eventually.

All (Git) projects that NOMAD depends on are either a Git submodule (you find them all in the `dependencies` directory or its subdirectories) or they are listed as PyPI packages in the `pyproject.toml` of the main project (or one of its submodules).

You can also have a look at the list of parsers (../../reference/parsers.md) and built-in plugins (../../reference/plugins.md) that constitute the majority of these projects. The only other projects are MatID (https://github.com/nomad-coe/matid){:target="_blank"}, DOS fingerprints (https://gitlab.mpcdf.mpg.de/nomad-lab/nomad-dos-fingerprints){:target="_blank"}, and the NOMAD Remote Tools Hub (https://gitlab.mpcdf.mpg.de/nomad-lab/nomad-remote-tools-hub){:target="_blank"}.

!!! note

The GitLab organization nomad-lab (https://gitlab.mpcdf.mpg.de/nomad-lab){:target="_blank"} and the

GitHub organizations for FAIRmat (https://github.com/fairmat-nfdi){:target="_blank"} and the

NOMAD CoE (https://github.com/nomad-coe){:target="_blank"} all represent larger infrastructure and

research projects, and they include many other Git projects that are not related.

When navigating the codebase, only follow the submodules.

Python code

There are three main directories with Python code:

- `nomad`: The actual NOMAD code. It is structured into more subdirectories and modules.
- `tests`: Tests (pytest (https://docs.pytest.org){:target="_blank"}) for the NOMAD code.

  It follows the same module structure, but Python files are prefixed with `test_`.
- `examples`: A few small Python scripts that might be linked in the documentation.

The `nomad` directory contains the following "main" modules. This list is not extensive but should help you to navigate the codebase:

- `app`: The FastAPI (https://fastapi.tiangolo.com/){:target="_blank"} APIs: v1 and v1.2 NOMAD APIs,

  OPTIMADE (https://www.optimade.org/){:target="_blank"}, DCAT (https://www.w3.org/TR/vocab-dcat-2/){:target="_blank"},

  h5grove (https://github.com/silx-kit/h5grove){:target="_blank"}, and more.
- `archive`: Functionality to store and access archive files. This is the storage format

  for all processed data in NOMAD. See also the docs on

  structured data (../../explanation/data.md).

- `cli`: The command line interface (based on Click (https://click.palletsprojects.com){:target="_blank"}).
  Subcommands are structured into submodules.
- `config`: NOMAD is configured through the `nomad.yaml` file. This contains all the
  (Pydantic (https://docs.pydantic.dev/){:target="_blank"}) models and default config parameters.
- `datamodel`: The built-in schemas (e.g. `nomad.datamodel.metainfo.workflow` used to construct
  workflows). The base sections and section for the shared entry structure.
  See also the docs on the datamodel (../../explanation/data.md) and
  processing (../../explanation/basics.md).
- `metainfo`: The Metainfo system, e.g. the schema language that NOMAD uses.
- `normalizing`: All the normalizers. See also the docs on
  processing (../../explanation/basics.md#normalizing).
- `parsing`: The base classes for parsers, matching functionality, parser initialization,
  some fundamental parsers like the *archive* parser. See also the docs on
  processing (../../explanation/basics.md#parsing).
- `processing`: It's all about processing uploads and entries. The interface to
  Celery (https://docs.celeryq.dev/en/stable/){:target="_blank"} and MongoDB (https://www.mongodb.com).
- `units`: The unit and unit conversion system based on
  Pint (https://pint.readthedocs.io){:target="_blank"}.
- `utils`: Utility modules, e.g. the structured logging system
  (structlog (https://www.structlog.org/){:target="_blank"}), id generation, and hashes.
- `files.py`: Functionality to maintain the files for uploads in staging and published.
  The interface to the file system.

- `search.py`: The interface to Elasticsearch (https://www.elastic.co/guide/en/enterprise-search/current/start.html){:target="_blank"}.

GUI code

The NOMAD UI is written as a React (https://react.dev/){:target="_blank"} single-page application (SPA). It uses (among many other libraries) MUI (https://mui.com/){:target="_blank"}, Plotly (https://plotly.com/python/){:target="_blank"}, and D3 (https://d3js.org/){:target="_blank"}. The GUI code is maintained in the `gui` directory. Most relevant code can be found in `gui/src/components`. The application entry point is `gui/src/index.js`.

Documentation

The documentation is based on MkDocs (https://www.mkdocs.org/){:target="_blank"}. The important files and directories are:

- `docs`: Contains all the Markdown files that contribute to the documentation system.
- `mkdocs.yml`: The index and configuration of the documentation. New files have to be added here as well.
- `nomad/mkdocs.py`: Python code that defines macros (https://mkdocs-macros-plugin.readthedocs.io/){:target="_blank"} which can be used in Markdown.

Other top-level directories

- `dependencies`: Contains all the submodules, e.g. the parsers.
- `ops`: Contains artifacts to run NOMAD components, e.g. `docker-compose.yaml`

files,

  and our Kubernetes Helm chart.

- `scripts`: Contains scripts used during the build or for certain development tasks.