The Journal of Open Source Software

pynxtools: A Python framework for generating and validating NeXus files in experimental data workflows

Sherjeel Shabih [1*], **Lukas Pielsticker** [1,2*], **Florian Dobener** [1,3], **Andrea Albino** [1], **Theodore Chang** [1], **Carola Emminger** [1,4], **Lev Ginzburg** [1], **Ron Hildebrandt** [1,4], **Markus Kühbach** [1], **Rubel Mozumder** [1], **Tommaso Pincelli** [5], **Martin Aeschlimann** [3], **Marius Grundmann** [4], **Walid Hetaba** [2], **Carlos-Andres Palma** [1,6], **Laurenz Rettig** [5], **Markus Scheidgen** [1], **José Antonio Márquez Prieto** [1], **Claudia Draxl** [1], **Sandor Brockhauser** [1], **Christoph Koch** [1], and **Heiko B. Weber** [7]

1 Physics Department and CSMB, Humboldt-Universität zu Berlin, Zum Großen Windkanal 2, D-12489 Berlin, Germany **2** Department Heterogeneous Reactions, Max Planck Institute for Chemical Energy Conversion, Stiftstraße 34-36, D-45470 Mülheim an der Ruhr, Germany **3** Department of Physics, RPTU Kaiserslautern-Landau, Erwin-Schrödinger-Str. 46, D-67663 Kaiserslautern, Germany **4** Felix Bloch Institute for Solid State Physics, Leipzig University, Linnestr. 5, D-04103 Leipzig, Germany **5** Department of Physical Chemistry, Fritz Haber Institute of the Max Planck Society, Faradayweg 4-6, D-14195 Berlin, DE **6** Institute of Physics, Chinese Academy of Sciences, No.8, 3rd South Street, Zhongguancun, Haidian District, Beijing, China **7** Lehrstuhl für Angewandte Physik, Friedrich-Alexander-Universität Erlangen-Nürnberg, Staudtstr. 7, D-91058 Erlangen, Germany ***** These authors contributed equally.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 14 October 2025
Published: unpublished

Summary

Scientific data across physics, materials science, and materials engineering often lacks adherence to FAIR principles (Barker et al., 2022; Jacobsen et al., 2020; M. D. Wilkinson et al., 2016; S. R. Wilkinson et al., 2025) due to incompatible instrument-specific formats and diverse standardization practices. `pynxtools` is a Python software development framework with a command line interface (CLI) that standardizes data conversion for scientific experiments in materials science to the NeXus format (Klosowski et al., 1997; Könnecke, 2006; Könnecke et al., 2015) across diverse scientific domains. NeXus defines data storage specifications for different experimental techniques through application definitions. `pynxtools` provides a fixed, versioned set of NeXus application definitions that ensures convergence and alignment in data specifications across, among others, atom probe tomography, electron microscopy, optical spectroscopy, photoemission spectroscopy, scanning probe microscopy, and X-ray diffraction. Through its modular plugin architecture `pynxtools` provides conversion of data and metadata from instruments and electronic lab notebooks to these unified definitions, while performing validation to ensure data correctness and NeXus compliance. `pynxtools` can be integrated directly into Research Data Management Systems (RDMS) to facilitate parsing and normalization. We detail one example for the RDM system NOMAD. By simplifying the adoption of NeXus, the framework enables true data interoperability and FAIR data management across multiple experimental techniques.

Statement of need

Achieving FAIR (Findable, Accessible, Interoperable, and Reproducible) data principles in experimental physics and materials science requires consistent implementation of standardized data formats. NeXus provides comprehensive data specifications for structured storage of scientific data. pynxtools simplifies the use of NeXus for developers and researchers by providing guided workflows and automated validation to ensure complete compliance. Existing solutions (Jemian et al., 2025; Könnecke et al., 2024) provide individual capabilities, but none offers a comprehensive end-to-end workflow for proper NeXus adoption. pynxtools addresses this critical gap by providing a framework that enforces complete NeXus application definition compliance through automated validation, detailed error reporting for missing required data points, and clear implementation pathways via configuration files and extensible plugins. This approach transforms NeXus from a complex specification into a practical solution, enabling researchers to achieve true data interoperability without deep technical expertise in the underlying standards.

Dataconverter and validation

The dataconverter, core module of pynxtools, combines instrument output files and data from electronic lab notebooks into NeXus-compliant HDF5 files. The converter performs three key operations: extracting experimental data through specialized readers, validating against NeXus application definitions to ensure compliance with existence and format constraints, and writing valid NeXus/HDF5 output files.

The dataconverter provides a command-line interface (CLI) for generating NeXus files, supporting both built-in readers for general-purpose functionality and technique-specific reader plugins, which are distributed as separate Python packages.

For developers, the dataconverter provides an abstract reader class for building plugins that process experiment-specific formats and populate the NeXus specification. It passes a Template, a subclass of Python's dictionary, to the reader as a form to fill. The Template ensures structural compliance with the chosen NeXus application definition and organizes data by NeXus's required, recommended, and optional levels.

The dataconverter validates reader output against the selected NeXus application definition, checking for instances of required concepts, complex dependencies (like inheritance and nested group rules), and data integrity (type, shape, constraints). It validates required concepts, reporting errors for any violations, and issues warnings for invalid data, facilitating reliable and practical NeXus file generation.

All reader plugins are tested using the pynxtools.testing suite, which runs automatically via GitHub CI to ensure compatibility with the dataconverter, the NeXus specification, and integration across plugins.

The dataconverter includes eln_mapper that creates either a fillable YAML file or a NOMAD (Scheidgen et al., 2023) ELN schema based on a selected NeXus application definition.

NeXus reader and annotator

read_nexus enables semantic access to NeXus files by linking data items to NeXus concepts, allowing applications to locate relevant data without hardcoding file paths. It supports concept-based queries that return all data items associated with a specific NeXus vocabulary term. Each data item is annotated by traversing its group path and resolving its corresponding NeXus concept, included inherited definitions.

Items not part of the NeXus schema are explicitly marked as such, aiding in validation and debugging. Targeted documentation of individual data items is supported through path-specific annotation. The tool also identifies and summarizes the file's default plottable data based on the NXdata definition.

NOMAD integration

While pynxtools works independently, it can also be integrated directly into any Research Data Management System (RDMS). The package works as a plugin within the NOMAD platform (Draxl & Scheffler, 2019; Scheidgen et al., 2023) out of the box. This enables data in the NeXus format to be integrated into NOMAD's metadata model, making it searchable and interoperable with other data from theory and experiment. The plugin consists of several key components (so called entry points):

pynxtools extends NOMAD's data schema, known as Metainfo (Ghiringhelli et al., 2017), by integrating NeXus definitions as a NOMAD Schema Package. This integration introduces NeXus-specific quantities and enables interoperability by linking to other standardized data representations within NOMAD. The dataconverter is integrated into NOMAD, making the conversion of data to NeXus accessible via the NOMAD GUI. The dataconverter also processes manually entered NOMAD ELN data in the conversion.

The NOMAD Parser module in pynxtools (NexusParser) extracts structured data from NeXus HDF5 files to populate NOMAD with Metainfo object instances as defined by the pynxtools schema package. This enables ingestion of NeXus data directly into NOMAD. Parsed data is post-processed using NOMAD's Normalization pipeline. This includes automatic handling of units, linking references (including sample and instrument identifiers defined elsewhere in NOMAD), and populating derived quantities needed for advanced search and visualization.

pynxtools contains an integrated Search Application for NeXus data within NOMAD, powered by Elasticsearch (Elasticsearch B.V., 2025). This provides a search dashboard whereby users can efficiently filter uploaded data based on parameters like experiment type, upload timestamp, and domain- and technique-specific quantities. The entire pynxtools workflow (conversion, parsing, and normalization) is exemplified in a representative NOMAD Example Upload that is shipped with the package. This example helps new users understand the workflow and serves as a template to adapt the plugin to new NeXus applications.

Funding

The work is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - project 460197019 (FAIRmat).

Acknowledgements

We acknowledge the following software packages our package depends on: h5py (Collette & al., 2008), numpy (Harris et al., 2020), click (The Pallets development team, 2014) , CFF (Druskat et al., 2021), xarray (S. Hoyer & Hamman, 2017), (Stephan Hoyer et al., 2025), pandas (The pandas development team, 2020), scipy (McKinney, 2010), lxml (Behnel et al., 2005), mergedeep (Clarke, 2019), Atomic Simulation Environment (Larsen et al., 2017), pint (The Pint development team, 2012).

References

Barker, M., Chue Hong, N. P., Katz, D. S., Lamprecht, A.-L., Martinez-Ortiz, C., Psomopoulos, F., Harrow, J., Castro, L. J., Gruenpeter, M., & Martinez, P. A. et. al. (2022). Introducing the FAIR principles for research software. *Scientific Data*, *9*(1), 622. https://doi.org/10.1038/s41597-022-01710-x

Behnel, S., Faassen, M., & Bicking, I. (2005). *lxml: XML and HTML with python*. https://lxml.de

Clarke, T. (2019). *Mergedeep: A deep merge function for python*. https://github.com/clarketm/mergedeep

Collette, A., & al., et. (2008). *h5py: HDF5 for python*. https://github.com/h5py/h5py

Draxl, C., & Scheffler, M. (2019). The NOMAD laboratory: From data sharing to artificial intelligence. *Journal of Physics: Materials*, *2*(3), 036001. https://doi.org/10.1088/2515-7639/ab13bb

Druskat, S., Spaaks, J. H., Chue Hong, N., Haines, R., Baker, J., Bliven, S., Willighagen, E., Pérez-Suárez, D., & Konovalov, O. (2021). *Citation file format*. https://doi.org/10.5281/zenodo.1003149

Elasticsearch B.V. (2025). Elasticsearch-py: Official python client for elasticsearch. In *GitHub repository*. GitHub. https://github.com/elastic/elasticsearch-py

Ghiringhelli, L. M., Carbogno, C., Levchenko, S., Mohamed, F., Huhs, G., Lüders, M., Oliveira, M., & Scheffler, M. (2017). Towards efficient data exchange and sharing for big-data driven materials science: Metadata and data formats. *Npj Computational Materials*, *3*(1), 46. https://doi.org/10.1038/s41524-017-0048-5

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hoyer, S., & Hamman, J. (2017). Xarray: N-D labeled arrays and datasets in Python. *J. Open Res. Software*. https://doi.org/10.5334/jors.148

Hoyer, Stephan, Roos, M., Joseph, H., Magin, J., Cherian, D., Fitzgerald, C., Hauser, M., Fujii, K., Maussion, F., Imperiale, G., Clark, S., Kleeman, A., Nicholas, T., Kluyver, T., Westling, J., Munroe, J., Amici, A., Barghini, A., Banihirwe, A., … Littlejohns, O. (2025). *Xarray* (Version v2025.03.1). Zenodo. https://doi.org/10.5281/zenodo.15110615

Jacobsen, A., Miranda Azevedo, R. de, Juty, N., Batista, D., Coles, S., Cornet, R., Courtot, M., Crosas, M., Dumontier, M., & Evelo, C. T. et. al. (2020). FAIR principles: Interpretations and implementation considerations. *Data Intelligence*, *2*(1-2), 10–29. https://doi.org/10.1162/dint_r_00024

Jemian, P. R., Ching, D., De Nolf, W., & Stöckli, P. (2025). *Prjemian/punx*. https://github.com/prjemian/punx

Klosowski, P., Koennecke, M., Tischler, J. Z., & Osborn, R. (1997). NeXus: A common format for the exchange of neutron and synchroton data. *Physica B: Condensed Matter*, *241-243*, 151–153. https://doi.org/10.1016/S0921-4526(97)00865-X

Könnecke, M. (2006). The state of the NeXus data format. *Physica B: Condensed Matter*, *385-386*, 1343–1345. https://doi.org/10.1016/j.physb.2006.06.106

Könnecke, M., Akeroyd, F. A., Bernstein, H. J., Brewster, A. S., Campbell, S. I., Clausen, B., Cottrell, S., Hoffmann, J. U., Jemian, P. R., & Männicke, D. et. al. (2015). The NeXus data format. *Applied Crystallography*, *48*(1), 301–305. https://doi.org/10.1107/S1600576714027575

Könnecke, M., Bernstein, H. J., Richter, T., Caswell, T. A., Jemian, P. R., & Levinsen, Y. (2024). *Nexusformat/cnxvalidate*. https://github.com/nexusformat/cnxvalidate

Larsen, A. H., Mortensen, J. J., Blomqvist, J., Castelli, I. E., Christensen, R., Dułak, M., Friis, J., Groves, M. N., Hammer, B., Hargus, C., Hermes, E. D., Jennings, P. C., Jensen, P. B., Kermode, J., Kitchin, J. R., Kolsbjerg, E. L., Kubal, J., Kaasbjerg, K., Lysgaard, S., … Jacobsen, K. W. (2017). The atomic simulation environment—a python library for working with atoms. *Journal of Physics: Condensed Matter*, *29*(27), 273002. https://doi.org/10.1088/1361-648X/aa680e

McKinney, Wes. (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt & Jarrod Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 56–61). https://doi.org/10.25080/Majora-92bf1922-00a

Scheidgen, M., Himanen, L., Ladines, A. N., Sikter, D., Nakhaee, M., Fekete, Á., Chang, T., Golparvar, A., Márquez, J. A., Brockhauser, S., Brückner, S., Ghiringhelli, L. M., Dietrich, F., Lehmberg, D., Denell, T., Albino, A., Näsström, H., Shabih, S., Dobener, F., … Draxl, C. (2023). NOMAD: A distributed web-based platform for managing materials science research data. *Journal of Open Source Software*, *8*(90), 5388. https://doi.org/10.21105/joss.05388

The Pallets development team. (2014). *Click: Command line interface creation kit*. https://github.com/pallets/click

The pandas development team. (2020). *Pandas-dev/pandas: pandas*. Zenodo. https://doi.org/10.5281/zenodo.3509134

The Pint development team. (2012). *Pint: Operate and manipulate physical quantities in Python*. https://github.com/hgrecco/pint. https://github.com/hgrecco/pint

Wilkinson, M. D., Dumontier, M., Aalbersberg, Ij. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., Silva Santos, L. B. da, & Bourne, P. E. et. al. (2016). The FAIR guiding principles for scientific data management and stewardship. *Scientific Data*, *3*(1), 1–9. https://doi.org/10.1038/sdata.2016.18

Wilkinson, S. R., Aloqalaa, M., Belhajjame, K., Crusoe, M. R., Paula Kinoshita, B. de, Gadelha, L., Garijo, D., Gustafsson, O. J. R., Juty, N., Kanwal, S., Khan, F. Z., Köster, J., Peters-von Gehlen, K., Pouchard, L., Rannow, R. K., Soiland-Reyes, S., Soranzo, N., Sufi, S., Sun, Z., … Goble, C. (2025). Applying the FAIR principles to computational workflows. *Sci. Data*, *12*(1), 328. https://doi.org/10.1038/s41597-025-04451-9