NWB Storage

Release v1.0.0

Jan 07, 2020

Table of Contents

CHAPTER 1

NWB:N Storage

1.1 What is the role of data storage?

The NWB:N format specification defined using the NWB:N specification language describes how to organize large collections of neuroscience data using basic primitives, e.g., Files, Groups, Datasets, Attributes, and Links to describe and hierarchically group data. The role of the data storage then is to store large collections of neuroscience data. In other words, the role of the storage is to map NWB:N primitives (and types, i.e., neurodata_types) to persistent storage. For an overview of the various components of the NWB:N project see here .

1.2 How are NWB:N files stored?

The NWB:N format currently uses HDF5 as primary storage mechanism. The mapping of the NWB:N format to HDF5 files is described in more detail in Section 2.

1.3 Are backends other than HDF5 supported?

NWB:N currently only officially supports HDF5 as main storage backend. However, the PyNWB API has been designed to enable the design of custom read/write backends for the API, enabling other storage backends to be mapped to NWB:N.

CHAPTER 2

HDF5

The NWB:N format currently uses the Hierarchical Data Format (HDF5) as the primary mechanism for data storage. HDF5 was selected for the NWB format because it met several of the project's requirements. First, it is a mature data format standard with libraries available in multiple programming languages. Second, the format's hierarchical structure allows data to be grouped into logical self-documenting sections. Its structure is analogous to a file system in which its "groups" and "datasets" correspond to directories and files. Groups and datasets can have attributes that provide additional details, such as authorities' identifiers. Third, its linking feature enables data stored in one location to be transparently accessed from multiple locations in the hierarchy. The linked data can be external to the file. Fourth, HDF5 is widely supported across programming languages (e.g., C, C++, Python, MATLAB, R among others) and tools, such as, HDFView, a free, cross-platform application, can be used to open a file and browse data. Finally, ensuring the ongoing accessibility of HDF-stored data is the mission of The HDF Group, the nonprofit that is the steward of the technology.

2.1 Format Mapping

Here we describe the mapping of NWB primitives (e.g., Groups, Datasets, Attributes, Links, etc.) used by the NWB format and specification to HDF5 storage primitives. As the NWB:N format was designed with HDF5 in mind, the high-level mapping between the format specification and HDF5 is quite simple:

Table 2.1: Mapping of groups

NWB Primitive	HDF5 Primitive
Group	Group
Dataset	Dataset
Attribute	Attribute
Link	Soft Link or External Link

Note: Using HDF5, NWB links are stored as HDF5 Soft Links or External Links. Hard Links are not used in NWB because the primary location and, hence, primary ownership and link path for secondary locations, cannot be determined for Hard Links.

2.2 Key Mapping

Here we describe the mapping of keys from the specification language to HDF5 storage objects:

2.2.1 Groups

Table 2.2: Mapping of groups

NWB Key	HDF5
name	Name of the Group in HDF5
doc	HDF5 attribute doc on the HDF5 group
groups	HDF5 groups within the HDF5 group
datasets	HDF5 datasets within the HDF5 group
attributes	HDF5 attributes on the HDF5 group
links	HDF5 SoftLinks within the HDF5 group
linkable	Not mapped; Stored in schema only
quantity	Not mapped; Number of appearances of the dataset.
neurodata_type	Attribute `neurodata_type`
namespace ID	Attribute `namespace`
object ID	Attribute `object_id`

2.2.2 Datasets

Table 2.3: Mapping of datasets

NWB Key	HDF5
name	Name of the dataset in HDF5
doc	HDF5 attribute doc on the HDF5 dataset
dtype	Data type of the HDF5 dataset (see *dtype mappings* table)
shape	Shape of the HDF5 dataset if the shape is fixed, otherwise shape defines the maxshape
dims	Not mapped
attributes	HDF5 attributes on the HDF5 group
linkable	Not mapped; Stored in schema only
quantity	Not mapped; Number of appearances of the dataset.
neurodata_type	Attribute `neurodata_type`
namespace ID	Attribute `namespace`
object ID	Attribute `object_id`

Note:

- TODO Update mapping of dims

2.2.3 Attributes

Table 2.4: Mapping of attributes

NWB Key	HDF5
name	Name of the attribute in HDF5
doc	Not mapped; Stored in schema only
dtype	Data type of the HDF5 attribute
shape	Shape of the HDF5 dataset if the shape is fixed, otherwise shape defines the maxshape
dims	Not mapped; Reflected by the shape of the attribute data
required	Not mapped; Stored in schema only
value	Data value of the attribute

2.2.4 Links

Table 2.5: Mapping of links

NWB Key	HDF5
name	Name of the HDF5 Soft Link
doc	Not mapped; Stored in schema only
target_type	Not mapped. The target type is determined by the type of the target of the HDF5 link

2.2.5 dtype mappings

The mappings of data types is as follows

`dtype` **spec value**	**storage type**	**size**
• "float" • "float32"	single precision floating point	32 bit
• "double" • "float64"	double precision floating point	64 bit
• "long" • "int64"	signed 64 bit integer	64 bit
• "int" • "int32"	signed 32 bit integer	32 bit
• "int16"	signed 16 bit integer	16 bit
• "int8"	signed 8 bit integer	8 bit
• "uint32"	unsigned 32 bit integer	32 bit
• "uint16"	unsigned 16 bit integer	16 bit
• "uint8"	unsigned 8 bit integer	8 bit
• "bool"	boolean	8 bit
• "text" • "utf" • "utf8" • "utf-8"	unicode	variable
• "ascii" • "str"	ascii	variable
• "ref" • "reference" • "object"	Reference to another group or dataset	
• region	Reference to a region of another dataset	
• compound dtype	HDF5 compound data type	
• "isodatetime"	ASCII ISO8061 date-time string. For example `2018-09-28T14:43:54.123+02:00`	variable

2.3 Caching format specifications

In practice it is useful to cache the specification a file was created with (including extensions) directly in the HDF5 file. Caching the specification in the file ensures that users can access the specification directly if necessary without requiring external resources. However, the mechanisms for caching format specifications is likely different for different storage backends and is not part of the NWB:N format specification itself. For the HDF5 backend, caching of the schema is implemented as follows.

The HDF5 backend adds the reserved top-level group `/specifications` in which all format specifications (including extensions) are cached. The `/specifications` group contains for each specification namespace a subgroup `/specifications/<namespace-name>/<version>` in which the specification for a particular version of a namespace are stored (e.g., `/specifications/core/2.0.1` in the case of the NWB:N core namespace at version 2.0.1). The actual specification data is then stored as a JSON string in scalar datasets with a binary, variable-length string data type (e.g., `dtype=special_dtype(vlen=binary_type)` in Python). The specification of the namespace is stored in `/specifications/<namespace-name>/<version>/namespace` while additional source files are stored in `/specifications/<namespace-name>/<version>/<source-filename>`. Here `<source-filename>` refers to the main name of the source-file without file extension (e.g., the core namespace defines `nwb.ephys.yaml` as source which would be stored in `/specifications/core/2.0.1/nwb.ecephys`).

CHAPTER 3

Release Notes

3.1 NWB:N - v2.1.0

Added documentation for new NWB key 'object_id' (see also format release notes for NWB 2.1.0: https://nwb-schema.readthedocs.io/en/latest/format_release_notes.html#september-2019).

3.2 NWB:N - v2.0.1

Added missing documentation on how format specification are cached in HDF5.

3.3 NWB:N - v2.0.0

Created separate reStructuredText documentation (i.e., this document) discuss and govern storage-related concerns. In particular this documents describes how primitives and keys described via the specification language are mapped to storage, in particular HDF5.

3.4 NWB:N - v1.0.x and earlier

For version 1.0.x and earlier, there was no official separate document governing NWB:N storage concerns as HDF5 was the only supported storage backend with implicit mapping between HDF5 types and NWB:N language primitives.

CHAPTER 4

Credits

4.1 Authors

4.1.1 NWB:N: Version 2.0.0 and later

Documentation for storage of Version 2 of the NWB:N format and later have been created by Oliver Ruebel and Andrew Tritt et al. in collaboration with the NWB:N community.

4.2 Acknowledgments

For details on the partners, funders, and supporters of NWB:N please the http://www.nwb.org/ project website. For specific contributions to the format specification and this document see the change logs of the Git repository at https://github.com/NeurodataWithoutBorders/nwb-schema .

CHAPTER 5

Legal

Contents

5.1 Copyright

5.2 Licence

CHAPTER 6

Indices and tables

- genindex
- modindex
- search