NWB:N: Advances towards an ecosystem for standardizing neurophysiology

O. Rübel[1*], A. Tritt[1*], N.H. Cain[6], B. Dichter[8], J.-C. Fillion Robin[7], D. Ozturk[7], L.M. Frank[4], E.F. Chang[3], F.T. Sommer[5], K. Svoboda[9], M.Grauer[7], W. Schroeder[7], L. Ng[6], K. Bouchard[2]

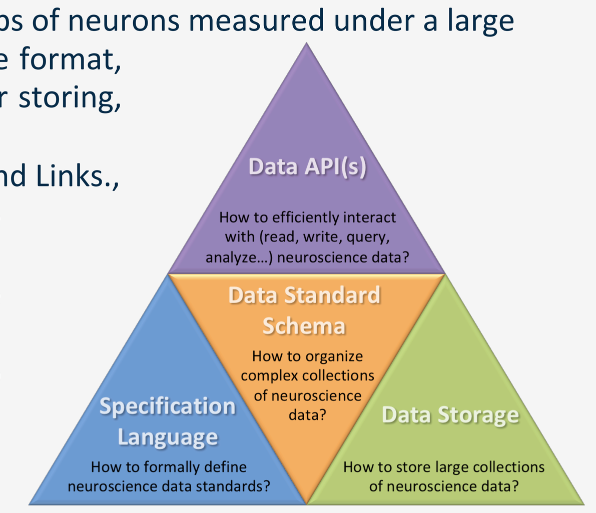
[1] Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley CA
[2] Biological Systems and Engineering Division, Lawrence Berkeley National Laboratory, Berkeley CA
[3] Department of Neurological Surgery, UC San Francisco, San Francisco, CA
[4] Department of Physiology, UC San Francisco, San Francisco, CA
[5] Redwood Center for Theoretical Neuroscience, UC Berkeley, Berkeley, CA
[6] Allen Institute for Brain Science, Seattle, WA
[7] Kitware Inc., Clifton Park, NY
[8] Stanford University, Stanford, CA
[9] HHMI / Janelia Farm Research Campus, Ashburn, VA
[*] These authors have contributed equally to this work

NWB:N – Overview

NWB:N: An Ecosystem for Neuroscience Data Standardization

Neurodata Without Borders: Neurophysiology (NWB:N) is an emerging unified data standard for neurophysiology data, focused primarily on the dynamics of groups of neurons measured under a large range of experimental conditions. NWB:N is more than just a file format, but it defines an ecosystem of tools, methods, and standards for storing, sharing, and analyzing complex neurophysiology data.

Using easy-to-use primitives, e.g.: Groups, Datasets, Attributes, and Links., the NWB:N **specification language** supports formal and verifiable specification of scientific file formats. Using the specification language, the **NWB:N standard schema** formally defines the organization of neuroscience data via the NWB:N data format. Using HDF5, the primary function of the **data storage** is to map NWB:N primitives to storage. Finally, **PyNWB** (Python) and **MatNWB** (Matlab) provide advanced **APIs** to create, access, and interact with NWB:N data files.



Use Cases: NWB: in Practice

The data type specifications in the NWB:N core data standard cover a broad range of common neurophysiology use cases:

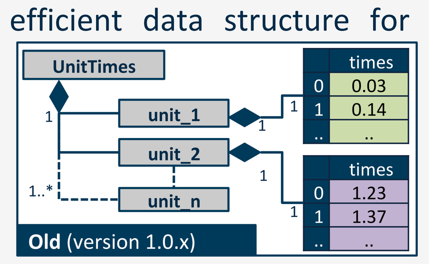
- **Electrophysiology:** The Frank Lab (UCSF) and Buzsaki Lab (NYU) use NWB:N to store local field potential and spiking data from recordings across multiple sites, as well as behavioral data from freely moving rats and experimental epoch data. The Bouchard Lab (LBNL), and Chang Lab (UCSF) among others are also using NWB:N for electrophysiology.
- **Calcium Imaging:** The Allen Institute for Brain Science (AIBS) uses NWB:N for calcium imaging data from their visual coding experiments available via the Allen Brain Observatory. This includes video data from visual stimulation, two photon recordings dF/F fluorescence data, and animal running speed. The Svoboda Lab (HHMI) and Losonczy Lab (Columbia) and others are also using NWB:N for calcium imaging.
- **Patch Clamp Recording:** AIBS uses NWB:N for storing data from Cell Types experiments available in the Allen Brain Atlas. This includes current injection and voltage responses from voltage clamp experiments.
- **Other:** Several labs are also exploring the use of NWB:N for other use cases, e.g., neural simulations (Soltesz Lab (Stanford) and AIBS), as well as the extension to new use cases.

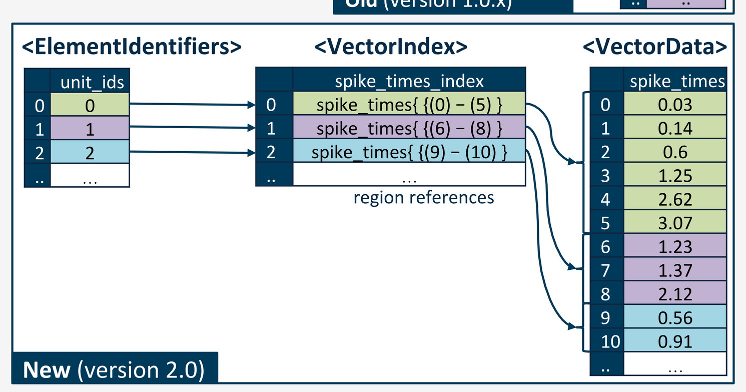
Improved Data Organization

Making the NBW:N Standard more Efficient and Easy-to-Use

Optimized Vector Data Storage

- **Use Case:** Store large collection of variable-length data vectors, e.g., to record event times of observed units.
- **Solution:** Provide efficient data structure for large vector data.
- **Advantages:** Avoid the need for many groups/datasets and improved I/O.





Data Tables

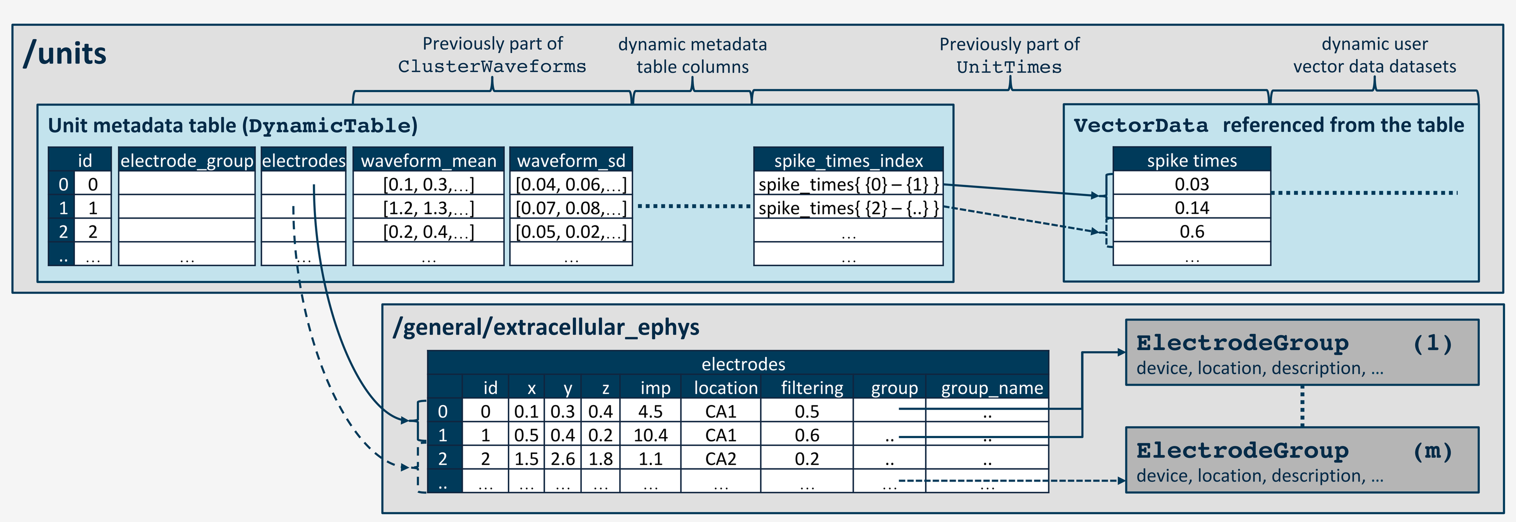
Data tables provide critical new functionality in NWB:N for improved organization of metadata:

- **Row-based tables:** New in NWB:N 2.0 is the ability to define flat compound data types. A row-based table can be defined as a 1D array with a compound type of all the table columns.
- **Column-based tables:** NWB:N 2.0 introduces `DynamicTable` as a means to store tables as a collection of `TableColumn` datasets, each describing a full column of the table.

	Row-based	Column-based
#Datasets	1	n
Add columns	Via extensions only	Dynamically or via extensions
Optimized element access	Rows	Columns
Supports vector data	Indirectly via references	Yes , via `VectorIndex` and `VectorData` types
Row-based referencing	Yes, via region references	Yes, via special type `DynamicTableRegion`
Column-based referencing	No (only indirectly)	Yes

Supporting Unit Data

- **Problem:** Improve efficiency and usability of spiking unit data and support dynamic unit metadata
- **Solution:** NWB:N 2.0 integrates `UnitTimes`, `ClusterWaveforms`, and `Clustering` into a new column-based table and uses the optimized vector data storage to efficiently store spike times.



Supporting Trial Data

- **Problem:** Users at the 5th NBW:N hackathon (LBNL, April 2018) identified improved support for trial data as a central need.
- **Solution:** NWB:N 2.0 supports trial-based metadata via dynamic tables.

Explicit Linking of Data

- Use explicit object/region references and links to improve data use and interpretation.

Select Other Data Types

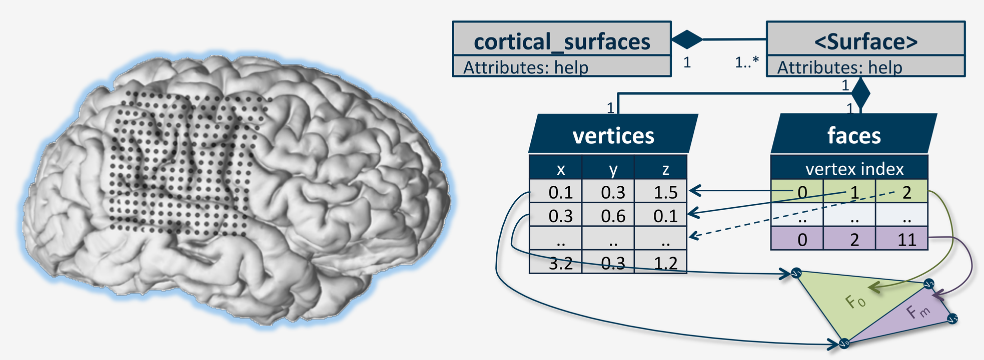
- **ROIs:** To improve storage of ROIs from plane segmentations, NWB:N 2.0 uses vector data storage to store many ROIs efficiently, dynamic tables to organize all metadata, and references for explicit linking of the data.
- **Epochs, Electrodes:** Using dynamic tables and explicit references improves the organization of metadata and supports dynamic metadata.
- **Sweeps:** NWB:N 2.0 adds support for sweeps.

Release Notes :
- **NWB:N Schema:** https://nwb-schema.readthedocs.io/en/latest/format_release_notes.html
- **NWB:N Schema Language:** https://schema-language.readthedocs.io/en/latest/specification_language_release_notes.html

Reports:
- **5th NWB:N Hackathon:** https://github.com/NeurodataWithoutBorders/nwb_hackathons/raw/master/HCK05_2018_Berkeley/report/report-5th-nwbn-hackathon-at-lbnl.pdf

Extending NWB:N: Integrating New Data Types

Use Case: Information about the shape of the cortical surface is critical in ECoG to be able to discover the locations of electrodes, i.e., which gyri and sucli the electrodes are recording from. To address this challenge we model the cortical surface as a triangular mesh.



Creating the Extension Schema

```
from pynwb.spec import NWBDatasetSpec, NWBNamespaceBuilder,
                       NWBGroupSpec, NWBAttributeSpec
surface = NWBGroupSpec(neurodata_type_def='Surface',
            neurodata_type_inc='NWBDataInterface' ,
            quantity='+', doc='brain cortical surface')
surface.add_dataset(NWBDatasetSpec(
            doc='..' , shape=(None, 3),
            name='faces', dtype='uint' , dims=...)
surface.add_dataset(NWBDatasetSpec(
            doc='..', shape=(None, 3) ,
            name='vertices', dtype='float', dims=...)
surface.add_attribute(...)

ns_builder = NWBNamespaceBuilder(doc=..., name='ecog',
               version='1.0', author='Ben Dichter', ...)
ns_builder.add_spec('ecog.extensions.yaml', surface)
ns_builder.export('ecog.namespace.yaml')
```

- **Result:** The result is a set of NWB:N schema language YAML files describing the extension.
- **Optional:** To define advanced API functionality for our extension we can create our own PyNWB API Container classes to interact with the data.

Writing/Reading with the Extension

PyNWB (write)

```
from pynwb import load_namespaces, get_class,
                  NWBHDF5IO, NWBFile ...
nwbfile = NWBFile(...)
load_namespaces('ecog.namespace.yaml')
Surface = get_class('Surface', 'ecog')
surf = Surface(faces=... , vertices=..., name='Surface_1'...)
nwbfile.add_acquisition(surf)
with NWBHDF5IO('surface_example.nwb', 'w') as io:
      io.write(nwbfile)
```

PyNWB (read)

```
load_namespaces('ecog.namespace.yaml' )
io = NWBHDF5IO('surface_example.nwb', 'r')
nwbfile = io.read()
nwbfile.get_acquisition('Surface 1').vertices
```

MatNWB (write)

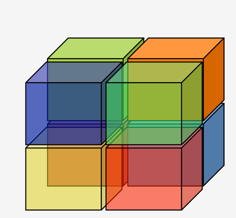
```
generateExtension('ecog.namespace.yaml');
file = nwbfile(...)
surf = types.ecog.Surface(...
              'faces', faces, 'vertices', vertices);
file.acquisition.set('Surfaces_1', surf);
nwbExport(file, 'ecephys_tutorial.nwb');
```

PyNWB Extensions: https://pynwb.readthedocs.io/en/latest/tutorials/general/extensions.html https://pynwb.readthedocs.io/en/latest/extensions.html

Optimizing Data Layout and I/O

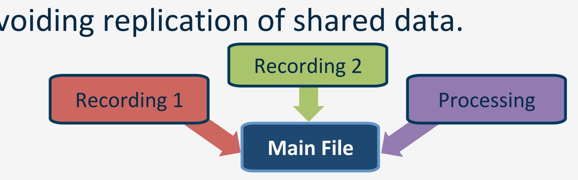
H5DataIO: Chunking and Compression

H5DataIO in PyNWB allows us to wrap arrays to control HDF5 data I/O options on a per-dataset basis. Using chunking enables optimization of data layouts for read/write, creation of resizable arrays, and use of I/O filters, e.g., compression.



Linking Data

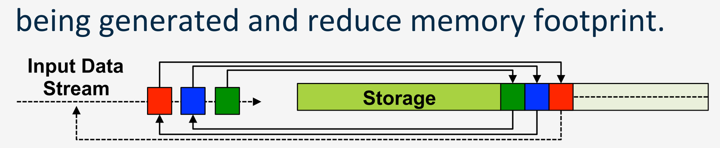
NWB:N supports linking of data within and across files, enabling modular data storage and avoiding replication of shared data.



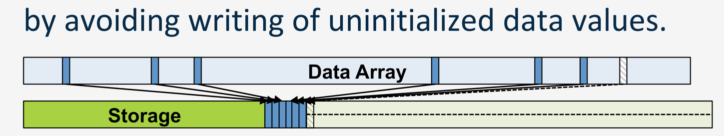
DataChunkIterator: Iterative Data Write

PyNWB supports iterative write by wrapping/generating arrays via *DataChunkIterator* with applications to:

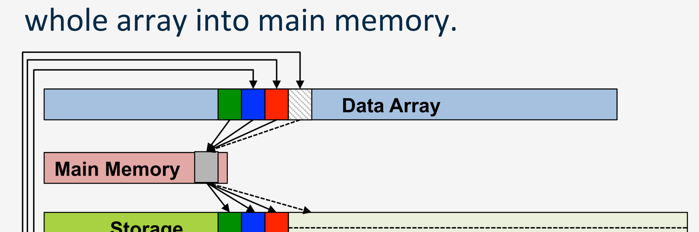
- **Data streaming/generators:** Write data as it is being generated and reduce memory footprint.



- **Sparse data arrays:** Reduce I/O (and storage) cost by avoiding writing of uninitialized data values.



- **Writing large data arrays:** Avoid loading the whole array into main memory.



Advanced HDF5 I/O: https://pynwb.readthedocs.io/en/latest/tutorials/general/advanced_hdf5_io.html
Linking Data: https://pynwb.readthedocs.io/en/latest/tutorials/general/linking_data.html
Iterative Write: https://pynwb.readthedocs.io/en/latest/tutorials/general/iterative_write.html

Community Outreach and Engagement

Working with the neuroscience community towards better science solutions

Science Engagement

- 65+ scientists from 20 major institutions attended the NWB:N development and user hackathons at AIBS and LBNL in April, 2018
- Many groups are already adopting NWB:N, e.g., FrankLab, ChangLab (UCSF), BouchardLab (LBNL), SvobodaLab (HHMI), MeisterLab (CalTech), Redwood (UCB), Allen Institute for Brain Science, among others.
- We are working actively with NIH BRAIN Initiative U19 projects:
  - Cross-U19 NWB:N interest subgroup
  - Soltesz U19 (sharp-wave ripple) -- Soltesz, Schnitzer (Stanford), Buzsaki (NYU), Losonczy (Columbia)
- KAVLI seed grants

Community Resources

Open

- Source: GitHub https://github.com/NeurodataWithoutBorders
- APIs: PyNWB: https://pynwb.readthedocs.io ; MatNWB: https://github.com/NeurodataWithoutBorders/matnwb
- Docs: Read the Docs, GitHub Pages https://neurodatawithoutborders.github.io/
- Channels: slack https://nwb-users.slack.com/ ; Google groups https://groups.google.com/forum/#!forum/neurodatawithoutborders

Industry Engagement

- Kitware (Visualization/CI)
- Vidrio Tech. (N. Clack, L. Niu) (MatNWB)
- MathWorks (Matlab)
- DataJoint (Data Management)

Governance

- NWB Executive Board
- NWB Technical Advisory Board
- KAVLI foundation

User Engagement

- User- and developer hackathons
- Outreach at neuroscience conferences
- Community discussion and surveys

Timeline: History and Future Developments

- 2014, Nov: 1st NWB Hackathon at HHMI
- 2015, May: 2nd NWB Hackathon at HHMI
- Aug: NWB:N 1.0 Released
- Nov: Neuron Paper
- 2016, Nov: NWB:N 1.0.5g Released
- 2017, Jan: KAVLI funds NWB4HPC Project led by LBNL
- Jul: 3rd NWB Hackathon at HHMI
- Aug: NWB Executive Board Founded
- Nov: NWB:N 2.0beta, PyNWB, and MatNWB released
- 2018, Apr: 4th NWB:N Developer Hackathon at AIBS; 5th NWB:N User Hackathon at LBNL
- Oct: NIH funds NWB:N
- Nov: Welcome to SfN
- Dec: NWB:N 2.0, PyNWB, MatNWB, release
- 2019, Mar: Cosyne 2019 (tentative)
- Apr: NWB:N hackathon (tentative)

NWB:N 1.0 — NWB:N 2.0 — Future Events (dates subject to change)

Future directions: Broadening applications, use, and adoption of NWB:N

Methods for integration of controlled vocabulary and provenance with NWB:N

Problem: To enable efficient discovery of data it is critical that metadata is specified using interpretable and consistent terms.

Approach: Integrate standardized metadata models and controlled vocabularies and ontologies with NWB:N, starting with selected key areas, e.g., spatial coordinates, genotype, and behavior. To facilitate interpretation and reuse we will also explore methods for integration of data provenance with NWB:N.

Easy-to-use, high-performance, parallel data query and analysis

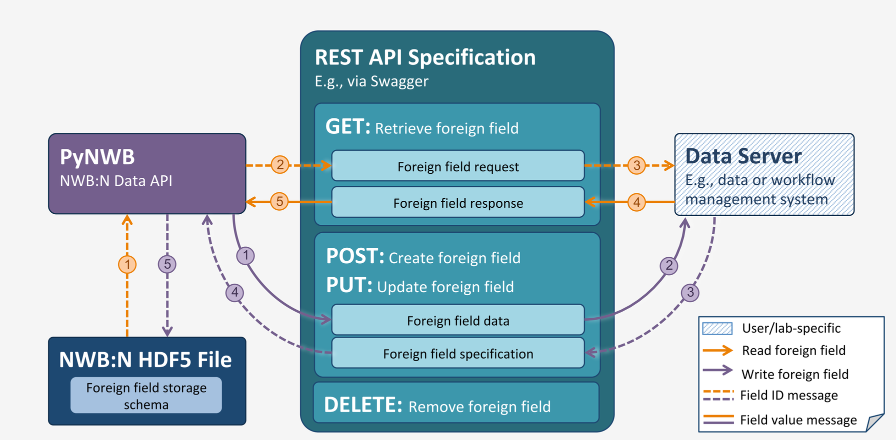
Problem: An often neglected but critical need in data-intensive analyses is the ability to efficiently identify, select, and process relevant data subsets.

Approach: Develop tools to make it easy to query and analyze NWB:N data in parallel.

Methods to facilitate integration of NWB:N with data management systems

Problem: Facilitate the integration of NWB:N with data management systems and provenance data and avoid duplicate storage of data across systems.

Approach: Support foreign fields for referencing of web-accessible data from NWB:N. Foreign fields are retrieved from and written to an external, web-accessible data service using a REST API standard and stored in NWB:N files via standardized JSON documents.



Methods and tools for facilitating community adoption, extension, and curation of NWB:N and integration of new use cases

Problem: Enable the neuroscience community to adopt and curate the NWB:N standard and facilitate the integration of new use-cases.

Approach: Develop the NWB:N-Hub, including easy-to-use mechanisms for extension and tools and processes for sharing, evaluation, acceptance, and use of NWB:N extensions.