NWB File Format Specification

Version 1.0.0

Table of Contents

Introduction

Neurodata Without Borders: Neurophysiology is a project to develop a unified data format for cellular-based neurophysiology data, focused on the dynamics of groups of neurons measured under a large range of experimental conditions. Participating labs provided use cases and critical feedback to the effort. The design goals for the NWB format included:

Compatibility

- Cross-platform
- Support for tool makers

Usability

- Quickly develop a basic understanding of an experiment and its data
- Review an experiment's details without programming knowledge

Flexibility

- Accommodate an experiment's raw and processed data
- Encapsulate all of an experiment's data, or link to external data source when necessary

Extensibility

- Accommodate future experimental paradigms without sacrificing backwards compatibility.
- Support custom extensions when the standard is lacking

Longevity

- Data published in the format should be accessible for decades

Hierarchical Data Format (HDF) 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, 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.

The NWB format standard is codified in a schema file written in a specification language created for this project. The specification language describes the schema, including data types and associations. A new schema file will be published for each revision of the NWB format standard. Tool developers are expected to take advantage of this explicit description of the format. However, data publishers are not expected to do so. They can add custom datasets to existing TimeSeries, Epochs, Modules and Interfaces.

Top Level Groups

Each NWB file is organized into several sections. These are stored as groups in the root of the HDF5 file. Those groups are as follows:

Group	Description	Comment
/**general**	Experimental metadata, including protocol, notes and description of hardware device(s)	The metadata stored in this section should be used to describe the experiment. Metadata necessary for interpreting the data is stored with the data.
/**acquisition**	Data streams recorded from the system, including ephys, ophys, tracking, etc.	This group is read-only after the experiment is completed and timestamps are corrected to a common timebase. The data stored here may be links to raw data stored in external HDF5 files. This will allow keeping bulky raw data out of the file while preserving the option of keeping some/all in the file.
/**stimulus**	Data pushed into the system (eg, video stimulus, sound, voltage, etc) and secondary representations of that data (eg, measurements of something used as a stimulus)	This group is read-only after experiment complete and timestamps are corrected to common timebase. Stores both presented stimuli and stimulus templates, the latter in case the same stimulus is presented multiple times, or is pulled from an external stimulus library.
/**epochs**	Experimental intervals, whether that be logically distinct sub-experiments having a particular scientific goal, trials during an experiment, or epochs deriving from analysis of data.	Epochs provide pointers to time series that are relevant to the epoch, and windows into the data in those time series (i.e., the start and end indices of *TimeSeries::data[]* that overlap with the epoch). This allows easy access to a range of data in specific experimental intervals.
/**processing**	The home for processing *Modules*. These modules perform intermediate analysis of data that is necessary to perform before scientific analysis. Examples include spike clustering, extracting position from tracking data, stitching together image slices.	*Modules* are defined below. They can be large and express many data sets from relatively complex analysis (e.g., spike detection and clustering) or small, representing extraction of position information from tracking video, or even binary “lick/no-lick” decisions. Common software tools (e.g., klustakwik, MClust) are expected to read/write data here.
/**analysis**	Lab-specific and custom scientific analysis of data. There is no defined format for the content of this group – the format is up to the individual user/lab.	To facilitate sharing analysis data between labs, the contents here should be stored in standard types (eg, INCF types) and appropriately documented.

The content of these organizational groups is more fully described in the section titled File Organization. The NWB format is based on *TimeSeries* and *Modules* objects and these are defined first.

NWB stores general optical and electrical physiology data in a way that should be understandable to a naive user after a few minutes using looking at the file in an HDF5 browser, such as HDFView. The format is designed to be friendly to and usable by software tools and analysis scripts, and to impose few a priori assumptions about data representation and analysis. Metadata required to understand the data itself (core metadata) is generally stored with the data. Information required to interpret the experiment (general metadata) is stored in the group 'general'. Most general metadata is stored in free-form text fields. Machine-readable metadata is stored as attributes on these free-form text fields.

The only API assumed necessary to read a NWB file is an HDF5 library (e.g., h5py in python, libhdf5 in C, JHI5 in Java).

Top-level datasets

Top-level datasets are for file identification and version information.

Dataset	Type	Description	Comments
/neurodata_version	text	File version string	Eg, “NWB-1.0.0”. This is the identifying string (e.g., “NWB”) with trailing major, minor and patch numbers.
/identifier	text	Unique file ID	Eg, concatenated lab name, file creation date/time and experimentalist, or a hash of these and/or other values. The goal is that the string should be unique to all other files.
/file_create_date	text	Time file was created, UTC	Date + time, Use ISO format (eg, ISO 8601). File can be created after the experiment was run, so this may differ from experiment start time.
modification_time (attr)	text array	Times that file was modified	This field is only present if an existing file is modified. Each modification time is stored (ISO 8601)
/session_start_time	text	Time of experiment/session start, UTC	Date + time, Use ISO format (eg, ISO 8601). All times stored in the file use this time as reference (ie, time zero)
/session_description	text	One or two sentences describing the experiment and data in the file	

TimeSeries

The file format is designed around a data structure called a *TimeSeries* which stores time-varying data. A *TimeSeries* is a superset of several INCF types, including signal events, image stacks and experimental events. To account for different storage requirements and different modalities, a *TimeSeries* is defined in a minimal form and it can be extended, or subclassed, to account for different modalities and data storage requirements. When a *TimeSeries* is extended, it means that the 'subclassed' instance maintains all of the data interface (eg, groups and datasets) as its parent and it has new groups and/or datasets of its own. The *TimeSeries* makes this process of defining such pairs more hierarchical.

Each *TimeSeries* has its own HDF5 group, and all datasets belonging to a *TimeSeries* are in that group. The group contains time and data components and users are free to add additional fields as necessary. There are two time objects represented. The first, *timestamps*, stores time information that is corrected to the experiment's time base (i.e., aligned to a master clock, with time-zero aligned to the starting time of the experiment). This field is used for data processing and subsequent scientific analysis. The second, *sync*, is an optional group that can be used to store the sample times as reported by the acquisition/stimulus hardware, before samples are converted to a common timebase and corrected relative to the master clock. This approach allows the NWB format to support streaming of data directly from hardware sources.

Dataset	Type	Description	Comment
data	polymorphic	Data values. Can also store binary data (eg, image frames)	This field may be a link to data stored in an external file, especially in the case of raw data.
unit (attr)	text	The base unit of measure used to store data. This should be in the SI unit.	This is the SI unit (when appropriate) of the stored data, such as Volts. If the actual data is stored in millivolts, the field 'conversion' below describes how to convert the data to the specified SI unit
conversion (attr)	float32	Scalar to multiply each element in data to convert it to the specified *unit*	
resolution (attr)	float32	Smallest meaningful difference between values in data, stored in the specified by *unit*	E.g., the change in value of the least significant bit, or a larger number if signal noise is known to be present.
timestamps	float64 array	Timestamps for the samples stored in *data*.	Timestamps here have all been corrected to the common experiment master-clock. Time is stored as seconds and all timestamps are relative to experiment start time. **Only one of *timestamps* and *starting_time* should be present**
interval (attr)	int32	The number of samples between each timestamp	Presently this value is restricted to 1 (ie, a timestamp for each sample)
unit (attr)	text	The string “Seconds”	All timestamps in the file are stored in seconds. Specifically, this is the number of seconds since the start of the experiment (i.e., since *session_start_time*)
starting_time	float64	The timestamp of the first sample	When timestamps are uniformly spaced, the timestamp of the first sample can be specified and all subsequent ones calculated from the sampling rate. **Only one of *timestamps* and *starting_time* should be present**
rate (attr)	float32	Sampling rate, in Hz	Rate information is stored in Hz
unit (attr)	text	The string “Seconds”	All timestamps in the file are stored in seconds. Specifically, this is he number of seconds since the start of the experiment (i.e., since *session_start_time*)
num_samples	int32	Number of samples in data, or number of image frames.	This is important if the length of *timestamp* and *data* are different, such as for externally stored stimulus image stacks
control	uint8 array	Numerical labels that apply to each element in data[]	Optional field. If present, the control array should have the same number of elements as data[]
control_description	text array	Description of each control value	Array length should be as long as the highest number in control, generating an indexed array for control values
sync/	group (optional)	Lab specific time and sync information as provided directly from hardware devices and that is necessary for aligning all acquired time information to a common timebase. The *timestamp* array stores time in the common timebase.	This group will usually only be populated in *TimeSeries* that are stored external to the NWB file, in files storing raw data. Once *timestamp* data is calculated, the contents of 'sync' are mostly for archival purposes.

When data is streamed from experiment hardware it should be stored in an HDF5 dataset having the same attributes as *data*, with time information stored as necessary. This allows the raw data files to be separate file-system objects that can be set as read-only once the experiment is complete. *TimeSeries* objects in /acquisition will link to the *data* field in the raw time series. Hardware-recorded time data must be corrected to a common time base (e.g., timestamps from all hardware sources aligned) before it can be included in *timestamps*. The

uncorrected time can be stored in the *sync* group.

The group holding the *TimeSeries* has the following attributes:

Attribute	Type	Description	Comment
description	text	Description of *TimeSeries*	
source	text array	Name of *TimeSeries* or *Modules* that serve as the source for the data contained here. It can also be the name of a device, for stimulus or acquisition data	
comments		Human-readable comments about the *TimeSeries*. This second descriptive field can be used to store additional information, or descriptive information if the primary description field is populated with a computer-readable string.	
ancestry	text array	The class-hierarchy of this *TimeSeries*, with one entry in the array for each ancestor. An alternative and equivalent description is that this *TimeSeries* object contains the datasets defined for all of the *TimeSeries* classes listed. The class hierarchy is described more fully below	For example: [0]=“TimeSeries”, [1]=“ElectricalSeries” [1]=“PatchClampSeries”
neurodata_type	text	The string “TimeSeries”	
data_link	text array	List of the paths of all *TimeSeries* that share a hard link to the same *data* field	Field is only present if links are present. List should include the path to this *TimeSeries* also
timestamp_link		List of the paths of all *TimeSeries* that share a hard link to the same *timestamps* field	Field is only present if links are present. List should include the path to this *TimeSeries* also
missing_fields	text array	(Optional) This is a list of fields defined as 'required' parts of the *TimeSeries* that are missing	Only present if one or more required fields are missing. Note that a missing required field (such as *data* or *timestamps*) should generate an error by the API
help	text	(Optional) Displays the 'description' field of the time series stored in the specification file/language	

The group holding the *TimeSeries* can be used to store additional information (HDF5 datasets) beyond what is required by the specification. I.e., an end user is free to add additional key/value pairs as necessary for their needs. It should be noted that such lab-specific extensions may not be recognized by analysis tools/scripts existing outside the lab. All datasets that are added and that are not defined by the schema should have an attribute 'neurodata_type' storing the string 'Custom'.

The *data* element in the *TimeSeries* will typically be an array of any valid HDF5 data type (e.g., a multi-dimentsional floating point array). The data stored can be in any unit. The attributes of the data field must indicate the SI unit that the data relates to (or appropriate counterpart, such as color-space) and the multiplier necessary to convert stored values to the specified SI unit.

TimeSeries Class Hierarchy

The *TimeSeries* is a data structure/object. It can be “subclassed” to represent more narrowly focused modalities (e.g., electrical versus optical physiology) as well as new modalities (eg, video tracking of whisker positions). When it a *TimeSeries* is subclassed, new datasets can be added while all datasets of parent classes are preserved. An initial set of subclasses are described here. Users are free to define subclasses for their particular requirements. This can be done by adding a new *TimeSeries* entry to the specification language file.

All datasets that are defined to be part of TimeSeries have the text attribute 'unit' that stores the unit specified in the documentation.

AbstractFeatureSeries extends TimeSeries

Abstract features, such as quantitative descriptions of sensory stimuli. The *TimeSeries::data* field is a 2D array, storing those features (e.g., for visual grating stimulus this might be orientation, spatial frequency and contrast). Array structure: [num frames] [num features]. Null stimuli (eg, uniform gray) can be marked as being an independent feature (eg, 1.0 for gray, 0.0 for actual stimulus) or by storing NaNs for feature values, or through use of the *TimeSeries::control* fields. A set of features is considered to persist until the next set of features is defined. The final set of features stored should be the null set.

Dataset	Type	Description
feature	text array	Description of the features represented in *TimeSeries::data*
feature_units	text array	Array of features' units

AnnotationSeries extends TimeSeries

Stores, eg, user annotations made during an experiment. The TimeSeries::data[] field stores a text array, and timestamps are stored for each annotation (ie, interval=1). This is largely an alias to a standard TimeSeries storing a text array but that is identifiable as storing annotations in a machine-readable way.

IndexSeries extends TimeSeries

Stores indices to image frames stored in an ImageSeries. The purpose of the ImageIndexSeries is to allow a static image stack to be stored somewhere, and the images in the stack to be referenced out-of-order. This can be for the display of individual images, or of movie segments (as a movie is simply a series of images). The *data* field stores the index of the frame in the referenced ImageSeries, and the *timestamps* array indicates when that image was displayed. The ImageIndexSeries containes all datasets of *TimeSeries* plus the following:

Dataset	Type	Description
indexed_timeseries	link: *TimeSeries*	HDF5 link to *TimeSeries* containing images that are indexed
indexed_timeseries_path	text	Path to linked *TimeSeries*

IntervalSeries extends TimeSeries

Stores intervals of data. The *timestamps* field stores the beginning and end of intervals. The *data* field stores whether the interval just started (>0 value) or ended (<0 value). Different interval types can be represented in the same series by using multiple key values (eg, 1 for feature A, 2 for feature B, 3 for feature C, etc). The field *data* stores an 8-bit integer. This is largely an alias of a standard TimeSeries but that is identifiable as representing time intervals in a machine-readable way.

OptogeneticSeries extends TimeSeries

Optogenetic stimulus. The data[] field is in unit of watts.

Dataset	Type	Description
site	text	Name of site description in general/optogentics

RoiResponseSeries extends TimeSeries
ROI responses over an imaging plane. Each row in data[] should correspond to the signal from one ROI.

Dataset	Type	Description
segmentation_interface	link: Segmentation interface	HDF5 link to image segmentation module defining ROIs
segmentation_interface_path	text	Path to segmentation module
roi_names	text array	List of ROIs represented, one name for each row of *data[]*

SpatialSeries extends TimeSeries
Direction, e.g., of gaze or travel, or position. The *TimeSeries::data* field is a 2D array storing position or direction relative to some reference frame. Array structure: [num measurements] [num dimensions]. Each SpatialSeries has a text dataset *reference_frame* that indicates the zero-position, or the zero-axes for direction. For example, if representing gaze direction, “straight-ahead” might be a specific pixel on the monitor, or some other point in space. For position data, the 0,0 point might be the top-left corner of an enclosure, as viewed from the tracking camera. The unit of *data* will indicate how to interpret *SpatialSeries* values. A SpatialSeries has all the datasets of a *TimeSeries* plus the following:

Dataset	Type	Description
reference_frame	text	Description defining what exactly “straight-ahead” means

ElectricalSeries extends TimeSeries
Stores acquired voltage data from extracellular recordings. The *data* field of an ElectricalSeries is an int or float array storing data in Volts. Array structure: [num time samples] [num channels]. It contains all of the datasets of the basic *TimeSeries* as well as the following:

Dataset	Type	Description
electrode_idx	int array	Indices to electrodes described in the experiment's electrode map array (under /general/extracellular_ephys)

SpikeEventSeries extends ElectricalSeries
Stores “snapshots” of spike events (i.e., threshold crossings) in *data*. This may also be raw data, as reported by ephys hardware. If so, the *TimeSeries::description* field should describing how events were detected. All *SpikeEventSeries* should reside in a module (under EventWaveform interface) even if the spikes were reported and stored by hardware. All events span the same recording channels and store snapshots of equal duration. *TimeSeries::data* array structure: [num events] [num channels] [num samples] (or [num events][num samples] for single electrode).

PatchClampSeries extends TimeSeries
Stores stimulus or response current or voltage. These are regular TimeSeries except for the addition of the following field:

Dataset	Type	Description
electrode_name	text	Name of electrode entry in /general/intracellular_ephys

VoltageClampStimulusSeries extends PatchClampSeries
CurrentClampStimulusSeries extends PatchClampSeries
These are aliases to standard PatchClampSeries. Their functionality is to better tag PatchClampSeries for machine (and human) readability of the file.

VoltageClampSeries extends PatchClampSeries
Stores current data recorded from intracellular voltage-clamp recordings. A corresponding *VoltageClampStimulusSeries* (stored separately as a stimulus) is used to store the voltage injected. The *VoltageClampSeries* has all of the datasets of an *PatchClampSeries* as well as the following:

Dataset	Type	Description
capacitance_fast	float	Unit: Farads
capacitance_slow	float	Unit: Farads
resistance_comp_bandwidth	float	Unit: Hz
resistance_comp_correction	float	Unit: %
resistance_comp_prediction	float	Unit: %
whole_cell_capacitance_comp	float	Unit: Farads
whole_cell_series_resistance_comp	float	Unit: Ohms
gain	float	Unit: Volt/Amp

CurrentClampSeries extends PatchClampSeries
Stores voltage data recorded from intracellular current-clamp recordings. A corresponding Current*ClampStimulusSeries* (stored separately as a stimulus) is used to store the current injected. The *CurrentClampSeries* has all of the datasets of an *PatchClampSeries* as well as the following:

Dataset	Type	Description
bias_current	float	Unit: Amps
bridge_balance	float	Unit: Ohms
capacitance_compensation	float	Unit: Farads
resistance_compensation	float	Unit: Ohms
gain	float	Unit: Volt/Volt

ImageSeries extends TimeSeries
General image data that is common between acquisition and stimulus time series. Sometimes the image data is stored in the HDF5 file in a raw format while other times it will be stored as an external image file in the host file system. The *data* field will either be binary data or empty. *TimeSeries::data* array structure: [frame][y][x] or [frame][z][y][x]. The *ImageSeries* contains all of the datasets of the *TimeSeries* as well as the following:

Dataset	Type	Description
format	text	Format of image. If this is “external” then the field external_file contains the path or URL information to that file. For tiff, png, jpg, etc, the binary representation of the image is stored in data. If the format is “raw” then the fields bit_per_pixel and dimension are used. For raw images, only a single channel is stored (eg, red)
external_file	text	Path or URL to external file, if format = “external”
bits_per_pixel	int	Number of bits per image pixel
dimension	int array	Number of pixels on x, y and z axes

ImageMaskSeries extends ImageSeries
An alpha mask that is applied to a presented visual stimulus. The data[] array contains an array of mask values that are applied to the displayed image. Mask values are stored as RGBA. Mask can vary with time. The timestamps array indicates the starting time of a mask, and that mask pattern continues until it's explicitly changed.

Dataset	Type	Description
masked_imageseries	link: *ImageSeries*	Link to *ImageSeries* that mask is applied to
masked_imageseries_path	text	Path to linked *ImageSeries*

TwoPhotonSeries extends ImageSeries
A special case of optical imaging. The *TwoPhotonSeries* has all the datasets of the *ImageSeries* as well as the following:

Dataset	Type	Description
pmt_gain	float	Photomultiplier gain
field_of_view	float array	Width, height and depth of image, or imaged area (meters)
imaging_plane	text	Name of imaging plane description in /general/optophysiology
scan_line_rate	float	Lines imaged per second. This is also stored in /general/optophysiology but is kept here as it is useful information for analysis, and so good to be stored w/ the actual data

OpticalSeries extends ImageSeries
Image data that is presented or recorded. A stimulus template movie will be stored only as an image. When the image is presented as stimulus, additional data is required, such as field of view (eg, how much of the visual field the image covers, or how what is the area of the target being imaged). If the *OpticalSeries* represents acquired imaging data, orientation is also important. The *OpticalSeries* has all datasets of the *ImageSeries* as well as the following

Dataset	Type	Description
field_of_view	float array	Width, height and depth of image, or imaged area (meters)
distance	float	Distance of camera/monitor from target/eye
orientation	text	Description of image relative to some reference frame (e.g., which way is 'up'). Must also specify frame of reference

Modules

NWB uses modules to store data for and represent the results of common data processing steps, such as spike sorting and image segmentation, that occur before scientific analysis of the data. Modules store the data used by software tools to calculate these intermediate results. Each module provides a list of the data it makes available, and it is free to provide whatever additional data that the module generates. Additional documentation is required for data that goes beyond standard definitions. All modules are stored in the /processing group.

Like *TimeSeries*, *Modules* are each have their own group where they store their data. Each module has the following attributes:

Attribute	Type	Description	Comments
interfaces	text array	Names of the data interfaces offered by this module. Interface descriptions are below	E.g., [0]="EventDetection", [1]= "Clusters", [2]= "FeatureExtraction"
neurodata_type	text	The string "Module"	

Modules have subgroups for each interface that is a part of the module, as well as any additional custom datasets and subgroups that are stored in the module. The *interface* list provides a machine-readable way to identify what information a module provides. Interfaces are not required to publish data and included interfaces can extend beyond what is listed here, so long as the data made available through the new interface is documented.

The *source* field provides information that allows moving backward through the processing stream to find the source of data. Storage of additional provenance information (e.g., parameters, user decisions, etc.) is left to the module's developer, as the types of provenance information can vary greatly between modules.

Module Interfaces

Several module interfaces are listed. When a module provides an interface name in its *interfaces* list, it is required to have the data fields listed with that interface. The data published by each interface should reside in a group in the module having the same name as the interface.

All interfaces have the following attributes:

Attribute	Type	Description
source	text	Path or description to the origin of the data represented in this module
neurodata_type	text	The string, "interface"

BehavioralEvents
BehavioralEpochs
BehavioralTimeSeries
The objective of these interfaces is to provide generic hooks for software tools/scripts. This allows a tool/script to take the output one specific interface (e.g., UnitTimes) and plot that data relative to another data modality (e.g., behavioral events) without having to define all possible modalities in advance. Declaring one of these interfaces means that one or more *TimeSeries* of the specified type is published. These *TimeSeries* should reside in a group having the same name as the interface. For example, if a BehavioralTimeSeries interface is declared, the module will have one or more *TimeSeries* defined in the module sub-group "BehavioralTimeSeries". BehavioralEpochs should use *IntervalSeries*. BehavioralEvents is used for irregular events. BehavioralTimeSeries is for continuous data.

Clustering

Clustered spike data, whether from automatic clustering tools (e.g., klustakwik) or as a result of manual sorting. A Clustering module publishes the following datasets:

Dataset/group	Type	Description
Clustering/	group	
times	double array	Times of clustered events, in seconds. This may be a link to *times* field in associated FeatureExtraction module. Array structure: [num events]
num	int array	Cluster number for each event Array structure: [num events]
description	text	Description of clusters or clustering (e.g., cluster 0 is electrical noise, clusters curated using Klusters, etc)
cluster_nums	int array	List of cluster numbers that are a part of this set (cluster numbers can be non-continuous)
peak_over_rms	float array	Maximum ratio of waveform peak to RMS on any channel in the cluster (provides a basic clustering metric). Array structure: [num clusters]

ClusterWaveforms

The mean waveform shape, including standard deviation, of the different clusters. Ideally, the waveform analysis should be performed on data that is only high-pass filtered. This is a separate module because it is expected to require updating. For example, IMEC probes may require different storage requirements to store/display mean waveforms, requiring a new interface or an extension of this one. A ClusterWaveform module publishes the following datasets:

Dataset/group	Type	Description
ClusterWaveforms/	group	
waveform_mean	2D float array	The mean waveform for each cluster, using the same indices for each wave as cluster numbers in the associated Clustering module (i.e, cluster 3 is in array slot [3]). Waveforms corresponding to gaps in cluster sequence should be empty (e.g., zero-filled)
waveform_sd	2D float array	Stdev of waveforms for each cluster, using the same indices as in mean
waveform_filtering	text	Filtering applied to data before generating mean/sd
clustering_interface	link: *Clustering* interface	HDF5 link to Clustering interface that was the source of the clustered data
clustering_interface_path	text	Path to linked clustering interface

CompassDirection

With a CompassDirection interface, a module publishes a *SpatialSeries* object representing a floating point value for theta. The *SpatialSeries::reference_frame* field should indicate what direction corresponds to “0” and which is the direction of rotation (this should be “clockwise”). The *si_unit* for the *SpatialSeries* should be “radians” or “degrees”.

Dataset/group	Type	Description
CompassDirection/	group	
<*SpatialSeries*>	*SpatialSeries*	One of possibly many *SpatialSeries* storing direction. Name should be informative

DfOverF

dF/F information about a region of interest (ROI). Storage hierarchy of dF/F should be the same as for segmentation (ie, same names for ROIs and for image planes).

Dataset/group	Type	Description
DfOverF/	group	
<RoiResponseSeries>	*RoiResponseSeries*	One of possibly many RoiResponseSeries, one for each imaging plane. Name should match entry in /general/optophysiology

EventDetection

Detected spike events from voltage trace(s).

Dataset/group	Type	Description
EventDetection/	group	
times	double array	
detection_method	text	Description of how events were detected, such as voltage threshold, or dV/dT threshold, as well as relevant values.
source_idx	int array	Indices into source *ElectricalSeries::data* array corresponding to time of event. Module description should define what is meant by time of event (e.g., .25msec before action potential peak, zero-crossing time, etc). The index points to each event from the raw data
source_electricalseries	link:*ElectricalSeries*	HDF5 link to *ElectricalSeries* that this data was calculated from. Metadata about electrodes and their position can be read from that *ElectricalSeries* so it's not necessary to mandate that information be stored here
source_electricalseries_path	text	Path to linked *ElectricalSeries*

EventWaveform

Represents either the waveforms of detected events, as extracted from a raw data trace in /acquisition, or the event waveforms that were stored during experiment acquisition.

Dataset/group	Type	Description
EventWaveform/	group	
<SpikeEventSeries>	*SpikeEventSeries*	One of possibly many. Name should be informative

EyeTracking

Eye-tracking data, representing direction of gaze.

Dataset/group	Type	Description
EyeTracking/	group	
<SpatialSeries>	*SpatialSeries*	One of possibly many SpatialSeries storing direction of gaze

FeatureExtraction
Features, such as PC1 and PC2, that are extracted from signals stored in a SpikeEvent *TimeSeries* or other source.

Dataset/group	Type	Description
FeatureExtraction/	group	
features	float array	Multi-dimensional array of features extracted for each event. Array structure: [num events] [num channels] [num features]
times	double array	Times of events that features correspond to (can be a link). Array structure: [num events]
description	text array	Description of features (eg, “PC1”) for each of the extracted features. Array structure: [num features]
electrode_idx	int array	Indices to electrodes described in the experiment's electrode map array (under /general/extracellular_ephys). Array structure: [num channels]

FilteredEphys
Ephys data from one or more channels that has been subjected to filtering. Examples of filtered data include Theta and Gamma (LFP has its own interface). FilteredEphys modules publish an *ElectricalSeries* for each filtered channel or set of channels. The name of each *ElectricalSeries* is arbitrary but should be informative. The source of the filtered data, whether this is from analysis of another time series or as acquired by hardware, should be noted in each's *TimeSeries::description* field. There is no assumed 1::1 correspondence between filtered ephys signals and electrodes, as a single signal can apply to many nearby electrodes, and one electrode may have different filtered (e.g., theta and/or gamma) signals represented.

Dataset/group	Type	Description
FilteredEphys/	group	
<ElectricalSeries>	*ElectricalSeries*	One of possibly many ElectricalSeries storing DSP filtered data

Fluorescence
Fluorescence information about a region of interest (ROI). Storage hierarchy of fluorescence should be the same as for segmentation (ie, same names for ROIs and for image planes).

Dataset/group	Type	Description
Fluorescense/	group	
<RoiResponseSeries>	*RoiResponseSeries*	One of possibly many RoiResponseSeries, one for each imaging plane. Name should match entry in /general/optophysiology

ImageSegmentation

Stores pixels in an image that represent different regions of interest (ROIs). Pixels are stored in both lists and 2D maps representing image intensity. All segmentation data is stored in a "segmentation" subgroup. Each ROI is stored in its own subgroup within *ImageSegmentation*, with the ROI group containing both a 2D mask and a list of pixels that make up this mask. Also for masking neuropil. If segmentation is allowed to change with time, a new interface is required (e.g., use the former version of this one, with img_mask_0 and start_time_0).

Dataset/group	Type	Description
ImageSegmentation/	group	
<imaging_plane>	group	Group name is human-readable description of imaging plane
description	text	Description of image plane, recording wavelength, depth, etc
imaging_plane_name	text	Name of imaging plane under general/optophysiology
roi_list	text array	List of ROIs in this imaging plane
<roi_name>	group	Name of ROI
img_mask	2D float array	Mask, represented in 2D ([y][x]) intensity image
pix_mask	2D int array	List of pixels (x,y) that compose the mask
attr: weight	float array	Weight of each pixel
roi_description	text	Description of the ROI
reference_images	group	
<image_name>	*ImageSeries*	One or more image stacks that the masks apply to (can be one-element stack)

LFP

LFP data from one or more channels. The electrode map in each published *ElectricalSeries* will identify which channels are providing LFP data. Filter properties should be noted in the *ElectricalSeries* description or comments field.

Dataset/group	Type	Description
LFP/	group	
<*ElectricalSeries*>	*ElectricalSeries*	One of possibly many ElectricalSeries storing LFP data

MotionCorrection

Publishes an image stack where all frames are shifted (registered) to a common coordinate system, to account for movement and drift between frames.

Dataset/group	Type	Description
MotionCorrection/	group	
<image stack name>	group	One of possibly many. Name should be informative
original	link: *ImageSeries*	Link to image series that is being registered
xy_translation	*TimeSeries*	Stores the x,y delta necessary to align each frame to the common coordinates
corrected	*ImageSeries*	Image stack with frames shifted to the common coordinates

Position
Position data, whether along the x, x/y or x/y/z axis.

Dataset/group	Type	Description
Position/	group	
<*SpatialSeries*>	*SpatialSeries*	One of possibly many SpatialSeries storing position

PupilTracking
Eye-tracking data, representing pupil size.

Dataset/group	Type	Description
PupilTracking/	group	
<*TimeSeries*>	*TimeSeries*	One of possibly many TimeSeries storing pupil size

UnitTimes
Event times in the observed units (eg, cell, synapse, etc). The UnitTimes group contains a group for each unit. The name of the group should match the value in the source module, if that is possible/relevant (e.g., name of ROIs from Segmentation module).

Dataset/group	Type	Description
UnitTimes/	group	
unit_list	text array	List of units present
<unit_N>/	group	
times	double array	Spike times for the unit (exact or estimated)
source	text	Name, path or description of where unit times originated. This is necessary only if the info here differs from or is more fine-grained than the interface's source field
unit_description	text	Description of the unit (eg, cell type)

File Organization

Top-level group: general
General experimental metadata, including animal strain, experimental protocols, experimenter, devices, etc, are stored under 'general'. Core metadata (e.g., that required to interpret data fields) is stored with the data itself, and implicitly defined by the file specification (eg, time is in seconds). The strategy used here for storing non-core metadata is to use free-form text fields, such as would appear in sentences or paragraphs from a Methods section. Metadata fields are text to enable them to be more general, for example to represent ranges instead of numerical values. Machine-readable metadata is stored as attributes to these free-form datasets.

387

Datasets & subgroups	Type	Description	Comments
session_id	text	Lab-specific ID for	Only 1 session_id per file, with all time aligned to experiment start time.
experimenter	text	Who performed the experiment	More than one person OK. Can specify roles of different people involved.
institution	text	Institution where experiment was performed	
lab	text	Lab where experiment was performed	
related_publications	text	Publication information.	PMID, DOI, URL, etc. If multiple, concatenate together and describe which is which.
notes	text	Notes about the experiment	Things particular to this experiment
experiment_description	text	General description of the experiment	Can be from Methods
data_collection	text	Notes about data collection and analysis	Can be from Methods
stimulus	text	Notes about stimuli, such as how and where presented	Can be from Methods
pharmacology	text	Description of drugs used, including how and when they were administered	Anesthesia(s), painkiller(s), etc., plus dosage, concentration, etc.
surgery	text	Narrative description of surgery/surgeries, including date(s) and who performed surgery	Much can be copied from Methods
protocol	text	Experimental protocol, if applicable	E.g., include IACUC protocol
subject	group		
subject_id	text	ID of animal/person used/participating in experiment (lab convention)	
description	text	Description of subject and where subject came from (eg, breeder, if animal)	
species	text	Species	
genotype	text	Genetic strain	If absent, assume WT
sex	text		
age	text	Age of person, animal, embryo	
weight	text	Weight at time of experiment, at time of surgery, and at other important times	
virus	text	Information about virus(es) used in experiment, including virus ID, source, date made, injection location and volume, etc.	Content can be from Methods section of paper.
slices	text	Description of slice(s) in free-form text, including information about preparation, thickness, orientation, temperature and bath solution.	Content can be from Methods section of paper.

388

continued from previous page

Datasets & subgroups	Type	Description	Comments
intracellular_ephys	group		
filtering	text	Description of filtering used	Includes filtering type and parameters, frequency fall-off, etc. If this changes between *TimeSeries*, filter description should be stored as a text attribute for each *TimeSeries*.
electrode_X	group	One of possibly many.	Name should be informative. This can optionally be a free-form text field that includes relevant description and metadata.
description (attr)	text	Recording description, description of electrode (eg, whole-cell, sharp)	Free-form text (can be from Methods)
location	text	Area, layer, comments on estimation, stereotaxic coords (if in vivo), etc.	
device	text	Name(s) of device(s) in /general/devices	
slice	text	Name of entry in /general/slices, if relevant	
initial_access_resistance	text		
seal	text		
resistance	text	Electrode resistance	unit: Ohm
extracellular_ephys	group		
electrode_map	float array	Physical location of electrodes (x,y,z, in meters)	Location of electrodes relative to one another. This records the points in space. If an electrode is moved, it needs a new entry in the electrode map for its new location. Otherwise format doesn't support using the same electrode in a new location, or processing spikes pre/post drift.
electrode_group	text array	Identification string for probe, shank or tetrode each electrode resides on. Name should correspond to one of electrode_group_N groups below.	There's one entry here for each element in electrode_map. All elements in an electrode group should have a functional association, for example all being on the same planar electrode array, or on the same shank.
impedance	text array	Impedence of electrodes listed in electrode_map	Text, in the event that impedance is stored as range and not a fixed value
filtering	text	Description of filtering used	Includes filtering type and parameters, frequency fall-off, etc. If this changes between *TimeSeries*, filter description should be stored as a text attribute for each *TimeSeries*.

continued from previous page

Datasets & subgroups	Type	Description	Comments
electrode_group_X	group	One of possibly many groups, one for each electrode group. If the groups have a hierarchy, such as multiple probes each having multiple shanks, that hierarchy can be mirrored here, using groups for electrode_probe_N and subgroups for electrode_group_N.	Name is arbitrary but should be meaningful.
description	text	Description of probe/shank	
location	text	Description of probe location	E.g., stereotaxic coordinates and other data, e.g., drive placement, angle and orientation and tetrode location in drive and tetrode depth
device	link	Name(s) of device(s) in /general/devices	
optophysiology	group		
imaging_plane_X	group	One of possibly many groups describing an imaging plane	Name is arbitrary but should be meaningful. It is referenced by TwoPhotonSeries and also ImageSegmentation and DfOverF interfaces
manifold	3D float array	Physical position of each pixel. Array structure: [height][width][xyz]	“xyz” is a vector that represents the position of the pixel relative to the defined coordinate space or reference frame
unit (attr)	text	Base unit that coordinates are stored in (e.g., Meters)	
conversion (attr)	float	Multiplier to get from stored values to specified unit (e.g., 1e-3 for millimeters)	
reference_frame	text	Describes position and reference frame of manifold based on position of first element in manifold. For example, text description of anotomical location or vectors needed to rotate to common anotomical axis (eg, AP/DV/ML)	This field is necessary to interpret manifold. If manifold is not present then this field is not required
channel_X	group	One of possibly many groups storing channel-specific data	Name is arbitrary but should be meaningful
emission_lambda	text	Emission lambda for channel	
description	text	Any notes or comments about the channel	
indicator	text	Calcium indicator	
excitation_lambda	text	Excitation wavelength	
imaging_rate	text		
location	text	Location of image plane	
device	text	Name of device in /general/devices	

continued from previous page

Datasets & subgroups	Type	Description	Comments
optogenetics	group		
site_X	group	One of possibly many groups describing an optogenetic stimuluation site	Name is arbitrary but should be meaningful. Name is referenced by OptogeneticSeries
description	text		
device	text	Name of device in /general/devices	
excitation_lambda	text	Excitation wavelength	
location	text	Location of stimulation site	
devices	group	Description of hardware devices used during experiment.	Eg, monitors, ADC boards, microscopes, etc
device_X	text	One of possibly many. Information about device and device description.	Name should be informative. Contents can be from Methods.

Top-level group: acquisition

Acquired data includes tracking and experimental data streams (ie, everything measured from the system)

Datasets & subgroups	Type	Description	Comments
timeseries	group	Acquired TimeSeries	When importing acquisition data to an NWB file, all acquisition/tracking/stimulus data must already be aligned to a common time frame. It is assumed that this task has already been performed.
timeseries_X	generic TimeSeries	One of possibly many acquisition TimeSeries objects.	Name is arbitrary. The TimeSeries::data[] dataset may be a link to raw acquired data in an external file.
images	group		
image_X	binary	Photograph of experiment or experimental setup (video also OK)	Name is arbitrary
format (attr)	text	format of image	eg, jpg, png, mpeg
description (attr)	text	human description of image	If image is of slice data, include slice thickness and orientation, and reference to appropriate entry in /general/slices

If bulky data is stored in the /acquisition group, the data can exist in a separate HDF5 file that is linked to by the file being used for processing and analysis.

When converting data from another format into NWB, there will be times that some data, particularly the raw data in *acquisition* and *stimulus*, is not included as part of the conversion. In such cases, a *TimeSeries* should be created that represents the missing data, even if the contents of that *TimeSeries* are empty. This helps to interpret the data in the file.

Top-level group: stimulus

Stimuli are here defined as any signal that is pushed into the system as part of the experiment (eg, sound, video, voltage, etc). Many different experiments can use the same stimuli, and stimuli can be re-used during an experiment. The stimulus group is organized so that one version of template stimuli can be stored and these be used multiple times. These templates can exist in the present file or can be HDF5-linked to a remote library file.

Datasets and subgroups	Type	Description	Comments
templates	group	Template stimuli	Time stamps in templates are based on stimulus design and are relative to the beginning of the stimulus. When templates are used, the stimulus instances must convert presentation times to the experiment's time reference frame.
timeseries_X	generic TimeSeries	One of possibly many templates, stored as a time series	Name is arbitrary.
presentation	group	Stimuli presented during the experiment	
timeseries_X	generic TimeSeries	One of possibly many stimuli, stored as a time series	Name is arbitrary. The TimeSeries::data[] field for the stimulus may be a link to data a stimulus template.

See above, in *acquisition*, about missing *TimeSeries*.

Top-level group: epochs

An experiment can be separated into one or many logical intervals, with the order and duration of these intervals often definable before the experiment starts. In this document, and in the context of NWB, these intervals are called 'epochs'. Epochs have acquisition and stimulus data associated with them, and different epochs can overlap. Examples of epochs are the time when a rat runs around an enclosure or maze as well as intervening sleep sessions; the presentation of a set of visual stimuli to a mouse running on a wheel; or the uninterrupted presentation of current to a patch-clamped cell. Epochs can be limited to the interval of a particular stimulus, or they can span multiple stimuli. Different windows into the same time series can be achieved by including multiple instances of that time series, each with different start/stop times.

Datasets and subgroups	Type	Description	Comments
epoch_X		One of possibly many different experimental epoch	Name is arbitrary but must be unique within the experiment.
neurodata_type (attr)	text	The string “Epoch”	
description	text		
start_time	double		
stop_time	double		
tags	text array	User-defined tags to help identify or categorize epoch.	Can describe stimulus (if template) or behavioral characteristic (e.g., “lick left”)
timeseries_X		One of possibly many input or output streams recorded during epoch	Name is arbitrary and does not have to match the TimeSeries that it refers to
idx_start	int	Epoch's start index in TimeSeries data[] field	This can be used to calculate location in TimeSeries timestamp[] field
count	int	Number of data samples available in this time series, during this epoch	Changed from stop_idx to lessen ambiguity
timeseries		link to TimeSeries	

The top-level group *epochs* has the following attributes:

Attribute	Type	Description	Comments
links	text	A human-readable list mapping *TimeSeries* entries in the epoch to the path of the *TimeSeries* within the file	
tags	text array	A list of the different tags used by epochs	

Top-level group: processing

'Processing' refers to intermediate analysis of the acquired data to make it more amenable to scientific analysis. These are performed using *Modules*, as defined above. All modules reside in the processing group.

Top-level group: analysis

The file can store lab-specific and custom data analysis without restriction on its form or schema, reducing data formatting restrictions on end users. Such data should be placed in the analysis group. The analysis data should be documented so that it is sharable with other labs.

Extending the format

Extensibility is handled by allowing users to add new datasets to existing TimeSeries, Epochs, Modules and Interfaces. What is described in this document is the minimum data that these objects store. The user is free to store additional data as necessary. Documentation should be provided to describe extra data if it is not clear from the context what the data represent.

More extensive modifications, such as defining new TimeSeries or Interfaces, can be achieved by extending the Specification Language. Generally speaking, language-level extensions will only be necessary for changes that need to be recognized by new software tools. Popular extensions can be proposed and added to the official format specification. Documentation on how to extend the specification itself will be forthcoming.

This approach allows extensibility without breaking backward compatibility.

Acknowledgements

The Neurodata Without Borders: Neurophysiology Initiative is funded by GE, the Allen Institute for Brain Science, the Howard Hughes Medical Institute (HHMI), The Kavli Foundation and the International Neuroinformatics Coordinating Facility. Our founding scientific partners are the Allen Institute, the Svoboda Lab at the Janelia Research Campus of HHMI, the Meister Lab at the California Institute of Technology, the Buzsáki Lab at New York University School of Medicine, and the University of California, Berkeley. Ovation.io is our founding development partner.