NWB file format specification

Version 1.0.6, April 8, 2017

(File 'nwb_core.py', namespace 'core')

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. Data publishers can use the specification language to extend the format in order to store types of

data not managed by the base format.

Naming conventions

In this document (and in the specification language used to define the format) an identifier enclosed in angle brackets (e.g. "*<ElectricalSeries>*") denotes a group or dataset with a "variable" name. That is, the name within the HDF5 file is set by the application creating the file and multiple instances may be created within the same group (each having a unique name). Identifiers that are not enclosed in angle brackets (e.g. "*CompassDirection*") are the actual name of the group or dataset within the HDF5 file. There can only be one instance within a given group since the name is fixed.

Link types

In some instances, the specification refers to HDF5 links. When links are made within the file, HDF5 soft-links (and not hard-links) should be used. This is because soft-links distinguish between the link and the target of the link, whereas hard-links cause multiple names (paths) to be created for the target, and there is no way to determine which of these names are preferable in a given situation. If the target of a soft link is removed (or moved to another location in the HDF5 file)—both of which can be done using the HDF5 API—then the soft link will "dangle," that is point to a target that no longer exists. For this reason, moving or removing targets of soft links should be avoided unless the links are updated to point to the new location.

Automatically created components

In the format, the value of some datasets and attributes can usually be determined automatically from other parts of the HDF5 file. For example, a dataset that has as value the target of a link can be determined automatically from a list of links in the HDF5 file. When possible, the NWB API will automatically create such components and required groups. The components (datasets, attributes and required groups) that are automatically created by the API are indicated by the phrase *(Automatically created)* in the description or comment. The creation of these components is specified by the "autogen" option in the format specification language. This is not a part of the format (different API's may create the data files in different ways). The information is included for the convenience of those using the NWB API and also for developers of other APIs who may wish to also auto-generate these components.

Top level groups

Id	Description	Comment	Required
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. *(Automatically created)*	yes
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. *(Automatically created)*	yes
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. *(Automatically created)*	yes
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. *(Automatically created)*	yes
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. *(Automatically created)*	yes

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. *(Automatically created)*	yes

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* 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.

Id	Type	Description	Comment	Required
file_create_date	text array; dims: ['*unlimited*']	Time file was created, UTC, and subsequent modifications to file.	Date + time, Use ISO format (eg, ISO 8601) or a format that is easy to read and unambiguous. File can be created after the experiment was run, so this may differ from experiment start time. Each modifictation to file adds new entry to array.	yes
identifier	text	A unique text identifier for the file.	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.	yes
nwb_version	text	File version string.	Eg, NWB-1.0.0. This will be the name of the format with trailing major, minor and patch numbers.	yes
session_description	text	One or two sentences describing the experiment and data in the file.		yes
session_start_time	text	Time of experiment/session start, UTC.	Date + time, Use ISO format (eg, ISO 8601) or an easy-to-read and unambiguous format. All times stored in the file use this time as reference (ie, time zero)	yes

All times are stored in seconds using double precision (64 bit) floating point values. A smaller floating point value, e.g. 32 bit, is **not** permitted for storing times. This is because significant errors for time can result from using smaller data sizes. Throughout this document, sizes (number of bits) are provided for many datatypes (e.g. float32). If the size is followed by "!" then the size is the minimum size, otherwise it is the recommended size. For fields with a recommended size, larger or smaller sizes can be used (and for integer types both signed and unsigned), so long as the selected size encompasses the full range of data, and for floats, without loss of significant precision. Fields that have a minimum size can use larger, but not smaller sizes.

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 or changes each of the components (eg, groups and datasets) of its parent and may have 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.

<TimeSeries>

General purpose time series.

Id	Type	Description	Comment	Required
<TimeSeries>	group	Top level group for <TimeSeries>. Name should be descriptive.		yes
. ancestry (attr)	text	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 [2]=PatchClampSeries. The hierarchical order should be preserved in the array -- i.e., the parent object of subclassed element N in the array should be element N-1	yes
. comments (attr)	text	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.		recommended
. data_link (attr)	text array; dims: ['num_dlinks']	A sorted list of the paths of all TimeSeries that share a link to the same data field. Example element of list: "/stimulus/presentation/Sweep_0"`	Attribute is only present if links are present. List should include the path to this TimeSeries also. *(Automatically created)*	yes
. description (attr)	text	Description of TimeSeries		recommended
. extern_fields (attr)	text array; dims: ['num_extern_fields']	List of fields that are HDF5 external links.	Only present if one or more datasets is set to an HDF5 external link. *(Automatically created)*	recommended
. help (attr)	text	Short description indicating what this type of TimeSeries stores.		no
. missing_fields (attr)	text array; dims: ['num_missing_fields']	List of fields that are not optional (i.e. either required or recommended parts of the TimeSeries) that are missing. The list can also include additional identifiers, as long as they are not present in the group, whether or not they optional.	Only present if one or more required or recommended fields are missing. Note that a missing required field (such as data or timestamps) should generate an error by the API *(Automatically created)*	recommended
. neurodata_type (attr)	text	Value is the string "TimeSeries". (const)		yes
. source (attr)	text	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		yes
. timestamp_link (attr)	text array; dims: ['num_tslinks']	A sorted list of the paths of all TimeSeries that share a link to the same timestamps field. Example element of list: "/acquisition/timeseries/lick_trace"	Attribute is only present if links are present. List should include the path to this TimeSeries also. *(Automatically created)*	yes
. control	uint8 array; dims: ['num_times']	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[]. **If either control or control_description are present, then both must be present.**	no
			Array length should be as long as the highest number in control minus one, generating	

. control_description	text array; dims: ['num_control_values']	Description of each control value.	an zero-based indexed array for control values. **If either control or control_description are present, then both must be present.**	no
. data	any array; dims: [['num_times'], ['num_times', 'num_d2'], ['num_times', 'num_d2', 'num_d3']]	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.	yes
. . conversion (attr)	float32!	Scalar to multiply each element in data to convert it to the specified unit		yes
. . 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. If unknown, use NaN	yes
. . 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.	yes
. 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 *(Automatically created)* **Optional under '/stimulus/templates'.**	yes
. 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. **Either starting_time or timestamps must be present, but not both. Must not be present under '/stimulus/templates'.**	no
. . rate (attr)	float32!	Sampling rate, in Hz	Rate information is stored in Hz	yes
. . 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)	yes
. sync	group	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.	no
. timestamps	float64! array; dims: ['num_times']	Timestamps for 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. **Either starting_time or timestamps must be present, but not both. Must not be present under '/stimulus/templates'.**	yes
. . interval (attr)	int32	The number of samples between each timestamp.	Presently this value is restricted to 1 (ie, a timestamp for each sample)	yes

. . 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)	yes

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* 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 b e recognized by analysis tools/scripts existing outside the lab. Extensions are described in section Extending the format).

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" (or extended) 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 either preserved as specified in the parent class or replaced by a new definition (changed). In the tables that follow, identifiers in the "Id" column that change the definition in the parent class are underlined. An initial set of subclasses are described here. Users are free to define subclasses for their particular requirements. This can be done by creating an extension to the format defining a new *TimeSeries* subclass (see Extending the format).
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). 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.
<AbstractFeatureSeries> includes all elements of *<TimeSeries>* with the the following additions or changes:

Id	Type	Description	Comment	Required
<AbstractFeatureSeries>	group	Top level group for <AbstractFeatureSeries>. Name should be descriptive.		yes
. ancestry (attr)	text array; dims: ['2']	Value is ['TimeSeries', 'AbstractFeatureSeries'] (const)		yes
. help (attr)	text	Value is the string "Features of an applied stimulus. This is useful when storing the raw stimulus is impractical". (const)		no
. data	float32 array; dims: ['num_times', 'num_features']	Values of each feature at each time.		yes

. . conversion (attr)	float32!	Scalar to multiply each element in data to convert it to the specified unit		yes
. . 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. If unknown, use NaN	yes
. . unit (attr)	text	Value is the string "see 'feature_units'".		yes
. feature_units	text array; dims: ['num_features']	Units of each feature.		recommended
. features	text array; dims: ['num_features']	Description of the features represented in TimeSeries::data.		yes

<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.

<AnnotationSeries> includes all elements of *<TimeSeries>* with the the following additions or changes:

Id	Type	Description	Required
<AnnotationSeries>	group	Top level group for <AnnotationSeries>. Name should be descriptive.	yes
. ancestry (attr)	text array; dims: ['2']	Value is ['TimeSeries', 'AnnotationSeries'] (const)	yes
. help (attr)	text	Value is the string "Time-stamped annotations about an experiment". (const)	no
. data	text array; dims: ['num_times']	Annotations made during an experiment.	yes
. . conversion (attr)	float32!	Value is float('NaN') since this does not apply.	yes
. . resolution (attr)	float32!	Value is float('NaN') since this does not apply.	yes
. . unit (attr)	text	Value is "n/a" to indicate that this does not apply.	yes

<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. TimeSeries::data array structure: [num times] [num channels] (or [num_times] for single electrode).

<ElectricalSeries> includes all elements of *<TimeSeries>* with the the following additions or changes:

Id	Type	Description	Comment	Required
<ElectricalSeries>	group	Top level group for <ElectricalSeries>. Name should be descriptive.		yes
. ancestry (attr)	text array; dims: ['2']	Value is ['TimeSeries', 'ElectricalSeries'] (const)		yes
. help (attr)	text	Value is the string "Stores acquired voltage data from extracellular recordings". (const)		no
. data	number array; dims: [['num_times'], ['num_times', 'num_channels']]	Recorded voltage data.		yes
. . conversion (attr)	float32!	Scalar to multiply each element in data to convert it to the specified unit		yes
. . 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. If unknown, use NaN	yes
. . unit (attr)	text	Value is the string "volt".		yes
. electrode_idx	int32 array; dims: ['num_channels']	Indicies (zero-based) to electrodes in general/extracellular_ephys/electrode_map.		yes

<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).
<*SpikeEventSeries*> includes all elements of <*ElectricalSeries*> with the the following additions or changes:

Id	Type	Description	Comment	Required
<SpikeEventSeries>	group	Top level group for <SpikeEventSeries>. Name should be descriptive.		yes
. ancestry (attr)	text array; dims: ['3']	Value is ['TimeSeries', 'ElectricalSeries', 'SpikeEventSeries'] (const)		yes
. help (attr)	text	Value is the string "Snapshots of spike events from data.". (const)		no
. data	float32 array; dims: [['num_events', 'num_samples'], ['num_events', 'num_channels', 'num_samples']]	Spike waveforms.		yes
. . conversion (attr)	float32!	Scalar to multiply each element in data to convert it to the specified unit		yes
. . 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. If unknown, use NaN	yes
. . unit (attr)	text	Value is the string "volt".		yes

<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].
<*ImageSeries*> includes all elements of <*TimeSeries*> with the the following additions or changes:

Id	Type	Description	Comment	Required
<ImageSeries>	group	Top level group for <ImageSeries>. Name should be descriptive.		yes
. ancestry (attr)	text array; dims: ['2']	Value is ['TimeSeries', 'ImageSeries'] (const)		yes
. help (attr)	text	Value is the string "Storage object for time-series 2-D image data". (const)		no
. bits_per_pixel	int32	Number of bit per image pixel.		recommended
. data	number array; dims: [['x', 'y'], ['frame', 'y', 'x'], ['frame', 'z', 'y', 'x']]	Either binary data containing image or empty. **Either 'external_file' or 'data' must be specified, but not both**		yes
. . conversion (attr)	float32!	Scalar to multiply each element in data to convert it to the specified unit		yes
. . 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. If unknown, use NaN	yes
. . 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.	yes
. dimension	int32 array; dims: ['rank']	Number of pixels on x, y, (and z) axes.		recommended
. external_file	text array; dims: ['num_files']	Path or URL to one or more external file(s). Field only present if format=external. NOTE: this is only relevant if the image is stored in the file system as one or more image file(s). This field should NOT be used if the image is stored in another HDF5 file and that file is HDF5 linked to this file. **Either 'external_file' or 'data' must be specified, but not both**		no
		Each entry is the frame number (within the full ImageSeries) of the first frame in		

. . *starting_frame (attr)*	int array; dims: ['num_files']	the corresponding external_file entry. This serves as an index to what frames each file contains, allowing random access.Zero-based indexing is used. (The first element will always be zero).	yes
. 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).	recommended

<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.

<ImageMaskSeries> includes all elements of *<ImageSeries>* with the the following additions or changes:

Id	Type	Description	Required
<ImageMaskSeries>	group	Top level group for <ImageMaskSeries>. Name should be descriptive.	yes
. ancestry (attr)	text array; dims: ['3']	Value is ['TimeSeries', 'ImageSeries', 'ImageMaskSeries'] (const)	yes
. help (attr)	text	Value is the string "An alpha mask that is applied to a presented visual stimulus". (const)	no
. masked_imageseries	link; target type=<ImageSeries> (or subtype)	Link to ImageSeries that mask is applied to.	yes
. masked_imageseries_path	text	Path to linked ImageSeries *(Automatically created)*	yes

<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.

<OpticalSeries> includes all elements of *<ImageSeries>* with the the following additions or changes:

Id	Type	Description	Required
<OpticalSeries>	group	Top level group for <OpticalSeries>. Name should be descriptive.	yes
. ancestry (attr)	text array; dims: ['3']	Value is ['TimeSeries', 'ImageSeries', 'OpticalSeries'] (const)	yes
. help (attr)	text	Value is the string "Time-series image stack for optical recording or stimulus". (const)	no
. distance	float32	Distance from camera/monitor to target/eye.	recommended
. field_of_view	float32 array; dims: ['fov']	Width, height and depth of image, or imaged area (meters).	recommended
. orientation	text	Description of image relative to some reference frame (e.g., which way is up). Must also specify frame of reference.	recommended

Structured dimension(s):

Dimension	Components [name (unit)]		
fov (option 1)	width (meter)	height (meter)	
fov (option 2)	width (meter)	height (meter)	depth (meter)

<TwoPhotonSeries> extends <ImageSeries>

A special case of optical imaging.

<TwoPhotonSeries> includes all elements of *<ImageSeries>* with the the following additions or changes:

Id	Type	Description	Required
<TwoPhotonSeries>	group	Top level group for <TwoPhotonSeries>. Name should be descriptive.	yes
. ancestry (attr)	text array;	Value is ['TimeSeries', 'ImageSeries', 'TwoPhotonSeries'] (const)	yes

	dims: ['3']		
. help (attr)	text	Value is the string "Image stack recorded from 2-photon microscope". (const)	no
. field_of_view	float32 array; dims: ['whd']	Width, height and depth of image, or imaged area (meters).	recommended
. imaging_plane	text	Name of imaging plane description in /general/optophysiology.	yes
. pmt_gain	float32	Photomultiplier gain	recommended
. scan_line_rate	float32	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.	recommended

Structured dimension(s):

Dimension	**Components** [name (unit)]		
whd	width (meter)	height (meter)	depth (meter)

<IndexSeries> extends <TimeSeries>

Stores indices to (typically image) frames stored in another TimeSeries. Its purpose 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 TimeSeries, and the timestamps array indicates when that frame was displayed. Can also be used for non-image data.

<IndexSeries> includes all elements of *<TimeSeries>* with the the following additions or changes:

Id	Type	Description	Comment	Required
<IndexSeries>	group	Top level group for <IndexSeries>. Name should be descriptive.		yes
. ancestry (attr)	text array; dims: ['2']	Value is ['TimeSeries', 'IndexSeries'] (const)		yes
. help (attr)	text	Value is the string "A sequence that is generated from an existing image stack. Frames can be presented in an arbitrary order. The data[] field stores frame number in reference stack". (const)		no
. data	int array; dims: ['num_times']	Index of the frame in the referenced ImageSeries.		yes
. . conversion (attr)	float32!	Scalar to multiply each element in data to convert it to the specified unit		yes
. . 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. If unknown, use NaN	yes
. . 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.	yes
. indexed_timeseries	link; target type=<TimeSeries> (or subtype)	HDF5 link to TimeSeries containing images that are indexed.		yes
. indexed_timeseries_path	text	Path to linked TimeSeries *(Automatically created)*		yes

<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.

<IntervalSeries> includes all elements of *<TimeSeries>* with the the following additions or changes:

Id	Type	Description	Required

<IntervalSeries>	group	Top level group for <IntervalSeries>. Name should be descriptive.	yes
. ancestry (attr)	text array; dims: ['2']	Value is ['TimeSeries', 'IntervalSeries'] (const)	yes
. help (attr)	text	Value is the string "Stores the start and stop times for events". (const)	no
. data	int8 array; dims: ['num_times']	>0 if interval started, <0 if interval ended.	yes
. . conversion (attr)	float32!	Value is the number "nan"	yes
. . resolution (attr)	float32!	Value is the number "nan"	yes
. . unit (attr)	text	Value is the string "n/a".	yes

<OptogeneticSeries> extends <TimeSeries>

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

<OptogeneticSeries> includes all elements of *<TimeSeries>* with the the following additions or changes:

Id	Type	Description	Comment	Required
<OptogeneticSeries>	group	Top level group for <OptogeneticSeries>. Name should be descriptive.		yes
. ancestry (attr)	text array; dims: ['2']	Value is ['TimeSeries', 'OptogeneticSeries'] (const)		yes
. help (attr)	text	Value is the string "Optogenetic stimulus". (const)		no
. data	float32 array; dims: ['num_times']	Applied power for optogenetic stimulus.		yes
. . conversion (attr)	float32!	Scalar to multiply each element in data to convert it to the specified unit		yes
. . 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. If unknown, use NaN	yes
. . unit (attr)	text	Value is the string "watt".		yes
. site	text	Name of site description in general/optogentics.		yes

<PatchClampSeries> extends <TimeSeries>

Stores stimulus or response current or voltage. Superclass definition for patch-clamp data (this class should not be instantiated directly).

<PatchClampSeries> includes all elements of *<TimeSeries>* with the the following additions or changes:

Id	Type	Description	Comment	Required
<PatchClampSeries>	group	Top level group for <PatchClampSeries>. Name should be descriptive.		yes
. ancestry (attr)	text array; dims: ['2']	Value is ['TimeSeries', 'PatchClampSeries'] (const)		yes
. help (attr)	text	Value is the string "Superclass definition for patch-clamp data". (const)		no
. data	number array; dims: ['num_times']	Recorded voltage or current.		yes
. . conversion (attr)	float32!	Scalar to multiply each element in data to convert it to the specified unit		yes
. . 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. If unknown, use NaN	yes
. . 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.	yes
. electrode_name	text	Name of electrode entry in /general/intracellular_ephys.		yes
. gain	float	Units: Volt/Amp (v-clamp) or Volt/Volt (c-clamp)		recommended

<CurrentClampSeries> extends <PatchClampSeries>

Stores voltage data recorded from intracellular current-clamp recordings. A corresponding CurrentClampStimulusSeries (stored separately as a stimulus) is used to store the current injected.

<CurrentClampSeries> includes all elements of *<PatchClampSeries>* with the the following

additions or changes:

Id	Type	Description	Required
<CurrentClampSeries>	group	Top level group for <CurrentClampSeries>. Name should be descriptive.	yes
. ancestry (attr)	text array; dims: ['3']	Value is ['TimeSeries', 'PatchClampSeries', 'CurrentClampSeries'] (const)	yes
. help (attr)	text	Value is the string "Voltage recorded from cell during current-clamp recording". (const)	no
. bias_current	float32	Unit: Amp	recommended
. bridge_balance	float32	Unit: Ohm	recommended
. capacitance_compensation	float32	Unit: Farad	recommended

<IZeroClampSeries> extends <CurrentClampSeries>

Stores recorded voltage data from intracellular recordings when all current and amplifier settings are off (i.e., CurrentClampSeries fields will be zero). There is no CurrentClampStimulusSeries associated with an IZero series because the amplifier is disconnected and no stimulus can reach the cell.

<IZeroClampSeries> includes all elements of *<CurrentClampSeries>* with the the following additions or changes:

Id	Type	Description	Required
<IZeroClampSeries>	group	Top level group for <IZeroClampSeries>. Name should be descriptive.	yes
. ancestry (attr)	text array; dims: ['4']	Value is ['TimeSeries', 'PatchClampSeries', 'CurrentClampSeries', 'IZeroClampSeries'] (const)	yes
. help (attr)	text	Value is the string "Voltage from intracellular recordings when all current and amplifier settings are off". (const)	no

<CurrentClampStimulusSeries> extends <PatchClampSeries>

Aliases to standard PatchClampSeries. Its functionality is to better tag PatchClampSeries for machine (and human) readability of the file.

<CurrentClampStimulusSeries> includes all elements of *<PatchClampSeries>* with the the following additions or changes:

Id	Type	Description	Required
<CurrentClampStimulusSeries>	group	Top level group for <CurrentClampStimulusSeries>. Name should be descriptive.	yes
. ancestry (attr)	text array; dims: ['3']	Value is ['TimeSeries', 'PatchClampSeries', 'CurrentClampStimulusSeries'] (const)	yes
. help (attr)	text	Value is the string "Stimulus current applied during current clamp recording". (const)	no

<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.

<VoltageClampSeries> includes all elements of *<PatchClampSeries>* with the the following additions or changes:

Id	Type	Description	Required
<VoltageClampSeries>	group	Top level group for <VoltageClampSeries>. Name should be descriptive.	yes
. ancestry (attr)	text array; dims: ['3']	Value is ['TimeSeries', 'PatchClampSeries', 'VoltageClampSeries'] (const)	yes
. help (attr)	text	Value is the string "Current recorded from cell during voltage-clamp recording". (const)	no
. capacitance_fast	float32	Unit: Farad	recommended
. . unit (attr)	text	Value is the string "Farad".	recommended
. capacitance_slow	float32	Unit: Farad	recommended
. . unit (attr)	text	Value is the string "Farad".	recommended
. resistance_comp_bandwidth	float32	Unit: Hz	recommended
. . unit (attr)	text	Value is the string "Hz".	recommended

. resistance_comp_correction	float32	Unit: %	recommended
. . *unit (attr)*	text	Value is the string "pecent".	recommended
. resistance_comp_prediction	float32	Unit: %	recommended
. . *unit (attr)*	text	Value is the string "pecent".	recommended
. whole_cell_capacitance_comp	float32	Unit: Farad	recommended
. . *unit (attr)*	text	Value is the string "Farad".	recommended
. whole_cell_series_resistance_comp	float32	Unit: Ohm	recommended
. . *unit (attr)*	text	Value is the string "Ohm".	recommended

<VoltageClampStimulusSeries> extends <PatchClampSeries>

Aliases to standard PatchClampSeries. Its functionality is to better tag PatchClampSeries for machine (and human) readability of the file.

<VoltageClampStimulusSeries> includes all elements of *<PatchClampSeries>* with the the following additions or changes:

Id	Type	Description	Required
<VoltageClampStimulusSeries>	group	Top level group for <VoltageClampStimulusSeries>. Name should be descriptive.	yes
. ancestry (attr)	text array; dims: ['3']	Value is ['TimeSeries', 'PatchClampSeries', 'VoltageClampStimulusSeries'] (const)	yes
. help (attr)	text	Value is the string "Stimulus voltage applied during voltage clamp recording". (const)	no

<RoiResponseSeries> extends <TimeSeries>

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

<RoiResponseSeries> includes all elements of *<TimeSeries>* with the the following additions or changes:

Id	Type	Description	Comment	Required
<RoiResponseSeries>	group	Top level group for <RoiResponseSeries>. Name should be descriptive.		yes
. ancestry (attr)	text array; dims: ['2']	Value is ['TimeSeries', 'RoiResponseSeries'] (const)		yes
. help (attr)	text	Value is the string "ROI responses over an imaging plane. Each row in data[] should correspond to the signal from one ROI". (const)		no
. data	float32 array; dims: ['num_times', 'num_ROIs']	Signals from ROIs		yes
. . *conversion (attr)*	float32!	Scalar to multiply each element in data to convert it to the specified unit		yes
. . *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. If unknown, use NaN	yes
. . *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.	yes
. roi_names	text array; dims: ['num_ROIs']	List of ROIs represented, one name for each row of data[].		yes
. segmentation_interface	link; target type=ImageSegmentation (or subtype)	HDF5 link to image segmentation module defining ROIs.		yes
. segmentation_interface_path	text	Path to segmentation module. *(Automatically created)*		yes

<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.
<*SpatialSeries*> includes all elements of <*TimeSeries*> with the the following additions or changes:

Id	Type	Description	Comment	Required
<SpatialSeries>	group	Top level group for <SpatialSeries>. Name should be descriptive.		yes
. ancestry (attr)	text array; dims: ['2']	Value is ['TimeSeries', 'SpatialSeries'] (const)		yes
. help (attr)	text	Value is the string "Stores points in space over time. The data[] array structure is [num samples][num spatial dimensions]". (const)		no
. data	number array; dims: ['num_times', 'num_features']	2-D array storing position or direction relative to some reference frame.		yes
. . conversion (attr)	float32!	Scalar to multiply each element in data to convert it to the specified unit		yes
. . 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. If unknown, use NaN	yes
. . unit (attr)	text	Value is the string "meter".		yes
. reference_frame	text	Description defining what exactly 'straight-ahead' means.		recommended

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 directly under group /processing. The name of each module is chosen by the data provider (i.e. modules have a "variable" name). The particular data within each module is specified by one or more *interfaces*, which are groups residing directly within a module. Each interface extends (contains the attributes in) group <*Interface*> and has a fixed name (e.g. *ImageSegmentation*) that suggests the type of data it contains. The names of the interfaces within a given module are listed in the "interfaces" attribute for the module. The different types of Interfaces are described below.

<Module>

Module. Name should be descriptive. Stores a collection of related data organized by contained interfaces. Each interface is a contract specifying content related to a particular type of data.

Id	Type	Description	Comment	Required
<Module>	group	Top level group for <Module>. Name should be descriptive.		yes
. description (attr)	text	Description of Module		no
. interfaces (attr)	text array; dims: ['num_interfaces']	Names of the data interfaces offered by this module.	E.g., [0]="EventDetection", [1]="Clustering", [2]="FeatureExtraction" *(Automatically created)*	yes
. neurodata_type (attr)	text	The string "Module"		yes
. <Interface> subtype	group	Any subtype of <Interface>. These include: BehavioralEpochs, BehavioralEvents, BehavioralTimeSeries, ClusterWaveforms, Clustering, CompassDirection, DfOverF, EventDetection, EventWaveform, EyeTracking, FeatureExtraction, FilteredEphys, Fluorescence, ImageSegmentation, ImagingRetinotopy, LFP, MotionCorrection, Position, PupilTracking, UnitTimes		no

<Interface>

The attributes specified here are included in all interfaces.

Id	Type	Description	Required
<Interface>	group	Top level group for <Interface>. This is an abstract group which can only be used by subclassing.	yes
. help (attr)	text	Short description of what this type of Interface contains.	no
. neurodata_type (attr)	text	Value is the string "Interface".	yes
. source (attr)	text	Path to the origin of the data represented in this interface.	yes

BehavioralEpochs

TimeSeries for storing behavoioral epochs. The objective of this and the other two Behavioral interfaces (e.g. BehavioralEvents and BehavioralTimeSeries) 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.

BehavioralEpochs includes all elements of *<Interface>* with the the following additions or changes:

Id	Type	Description	Required
BehavioralEpochs	group	Top level group for BehavioralEpochs.	yes
. help (attr)	text	Value is the string "General container for storing behavorial epochs". (const)	no
. <IntervalSeries> or subtype	group	<IntervalSeries> or any subtype.	yes

BehavioralEvents

TimeSeries for storing behavioral events. See description of BehavioralEpochs for more details.

BehavioralEvents includes all elements of *<Interface>* with the the following additions or changes:

Id	Type	Description	Required
BehavioralEvents	group	Top level group for BehavioralEvents.	yes
. help (attr)	text	Value is the string "General container for storing event series". (const)	no
. <TimeSeries> or subtype	group	<TimeSeries> or any subtype. Subtypes include: <AbstractFeatureSeries>, <AnnotationSeries>, <CurrentClampSeries>, <CurrentClampStimulusSeries>, <ElectricalSeries>, <IZeroClampSeries>, <ImageMaskSeries>, <ImageSeries>, <IndexSeries>, <IntervalSeries>, <OpticalSeries>, <OptogeneticSeries>, <PatchClampSeries>, <RoiResponseSeries>, <SpatialSeries>, <SpikeEventSeries>, <TwoPhotonSeries>, <VoltageClampSeries>, <VoltageClampStimulusSeries>	yes

BehavioralTimeSeries

TimeSeries for storing Behavoioral time series data.See description of BehavioralEpochs for more details.

BehavioralTimeSeries includes all elements of *<Interface>* with the the following additions or changes:

Id	Type	Description	Required
BehavioralTimeSeries	group	Top level group for BehavioralTimeSeries.	yes
. <TimeSeries> or subtype	group	<TimeSeries> or any subtype. Subtypes include: <AbstractFeatureSeries>, <AnnotationSeries>, <CurrentClampSeries>, <CurrentClampStimulusSeries>, <ElectricalSeries>, <IZeroClampSeries>, <ImageMaskSeries>, <ImageSeries>, <IndexSeries>, <IntervalSeries>, <OpticalSeries>, <OptogeneticSeries>, <PatchClampSeries>, <RoiResponseSeries>, <SpatialSeries>, <SpikeEventSeries>, <TwoPhotonSeries>, <VoltageClampSeries>, <VoltageClampStimulusSeries>	yes

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.
ClusterWaveforms includes all elements of *<Interface>* with the the following additions or changes:

Id	Type	Description	Required
ClusterWaveforms	group	Top level group for ClusterWaveforms.	yes
. help (attr)	text	Value is the string "Mean waveform shape of clusters. Waveforms should be high-pass filtered (ie, not the same bandpass filter used waveform analysis and clustering)". (const)	no
. clustering_interface	link; target type=Clustering (or subtype)	HDF5 link to Clustering interface that was the source of the clustered data	yes
. clustering_interface_path	text	Path to linked clustering interface *(Automatically created)*	yes
. waveform_filtering	text	Filtering applied to data before generating mean/sd	yes
. waveform_mean	float32 array; dims: ['num_clusters', 'num_samples']	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)	yes
. waveform_sd	float32 array; dims: ['num_clusters', 'num_samples']	Stdev of waveforms for each cluster, using the same indices as in mean	yes

Clustering

Clustered spike data, whether from automatic clustering tools (e.g., klustakwik) or as a result of manual sorting.
Clustering includes all elements of *<Interface>* with the the following additions orchanges:

Id	Type	Description	Required
Clustering	group	Top level group for Clustering.	yes
. help (attr)	text	Value is the string "Clustered spike data, whether from automatic clustering tools (eg, klustakwik) or as a result of manual sorting". (const)	no
. cluster_nums	int32 array; dims: ['num_clusters']	List of cluster number that are a part of this set (cluster numbers can be non-continuous) *(Automatically created)*	yes
. description	text	Description of clusters or clustering, (e.g. cluster 0 is noise, clusters curated using Klusters, etc)	yes
. num	int32 array; dims: ['num_events']	Cluster number of each event	yes
. peak_over_rms	float32 array; dims: ['num_clusters']	Maximum ratio of waveform peak to RMS on any channel in the cluster (provides a basic clustering metric).	yes
. times	float64! array; dims: ['num_events']	Times of clustered events, in seconds. This may be a link to times field in associated FeatureExtraction module.	yes

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.
CompassDirection includes all elements of *<Interface>* with the the following additions or changes:

Id	Type	Description	Required
CompassDirection	group	Top level group for CompassDirection.	yes
. help (attr)	text	Value is the string "Direction as measured radially. Spatial series reference frame should indicate which direction corresponds to zero and what is the direction of positive rotation". (const)	no
. <SpatialSeries> or subtype	group	<SpatialSeries> or any subtype.	yes

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).
DfOverF includes all elements of *<Interface>* with the the following additions orchanges:

Id	Type	Description	Required
DfOverF	group	Top level group for DfOverF.	yes
. help (attr)	text	Value is the string "Df/f over time of one or more ROIs. TimeSeries names should correspond to imaging plane names". (const)	no
. <RoiResponseSeries> or subtype	group	<RoiResponseSeries> or any subtype.	yes

EventDetection

Detected spike events from voltage trace(s).
EventDetection includes all elements of *<Interface>* with the the following additions orchanges:

Id	Type	Description	Required
EventDetection	group	Top level group for EventDetection.	yes
. detection_method	text	Description of how events were detected, such as voltage threshold, or dV/dT threshold, as well as relevant values.	yes
. source_electricalseries	link; target type=<ElectricalSeries> (or subtype)	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	yes
. source_electricalseries_path	text	Path to linked ElectricalSeries. *(Automatically created)*	yes
. source_idx	int32 array; dims: ['num_events']	Indices (zero-based) 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	yes
. times	float64! array; dims: ['num_events']	Timestamps of events, in Seconds	yes

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.
EventWaveform includes all elements of *<Interface>* with the the following additions orchanges:

Id	Type	Description	Required
EventWaveform	group	Top level group for EventWaveform.	yes
. help (attr)	text	Value is the string "Waveform of detected extracellularly recorded spike events". (const)	no
. <SpikeEventSeries> or subtype	group	<SpikeEventSeries> or any subtype.	yes

EyeTracking

Eye-tracking data, representing direction of gaze.
EyeTracking includes all elements of *<Interface>* with the the following additions orchanges:

Id	Type	Description	Required
EyeTracking	group	Top level group for EyeTracking.	yes
. help (attr)	text	Value is the string "Eye-tracking data, representing direction of gaze". (const)	no
. <SpatialSeries> or subtype	group	<SpatialSeries> or any subtype.	yes

FeatureExtraction

Features, such as PC1 and PC2, that are extracted from signals stored in a SpikeEvent TimeSeries or other source.
FeatureExtraction includes all elements of *<Interface>* with the the following additions or changes:

Id	Type	Description	Required
FeatureExtraction	group	Top level group for FeatureExtraction.	yes
. help (attr)	text	Value is the string "Container for salient features of detected events". (const)	no
. description	text array; dims: ['num_features']	Description of features (eg, "PC1") for each of the extracted	yes

		features.	
. electrode_idx	int32 array; dims: ['num_channels']	Indices (zero-based) to electrodes described in the experiment's electrode map array (under /general/extracellular_ephys).	yes
. features	float32 array; dims: ['num_events', 'num_channels', 'num_features']	Multi-dimensional array of features extracted from each event.	yes
. times	float64! array; dims: ['num_events']	Times of events that features correspond to (can be a link).	yes

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.

FilteredEphys includes all elements of *<Interface>* with the the following additions orchanges:

Id	Type	Description	Required
FilteredEphys	group	Top level group for FilteredEphys.	yes
. help (attr)	text	Value is the string "Ephys data from one or more channels that is subjected to filtering, such as for gamma or theta oscillations (LFP has its own interface). Filter properties should be noted in the ElectricalSeries". (const)	no
. <ElectricalSeries> or subtype	group	<ElectricalSeries> or any subtype. Subtypes include:<SpikeEventSeries>	yes

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).

Fluorescence includes all elements of *<Interface>* with the the following additions orchanges:

Id	Type	Description	Required
Fluorescence	group	Top level group for Fluorescence.	yes
. help (attr)	text	Value is the string "Fluorescence over time of one or more ROIs. TimeSeries names should correspond to imaging plane names". (const)	no
. <RoiResponseSeries> or subtype	group	<RoiResponseSeries> or any subtype.	yes

ImageSegmentation

Stores pixels in an image that represent different regions of interest (ROIs) or masks. All segmentation for a given imaging plane is stored together, with storage for multiple imaging planes (masks) supported. Each ROI is stored in its own subgroup, with the ROI group containing both a 2D mask and a list of pixels that make up this mask. Segments can also be used for masking neuropil. If segmentation is allowed to change with time, a new imaging plane (or module) is required and ROI names should remain consistent between them.

ImageSegmentation includes all elements of *<Interface>* with the the following additions or changes:

Id	Type	Description	Required
ImageSegmentation	group	Top level group for ImageSegmentation.	yes
. help (attr)	text	Value is the string "Stores groups of pixels that define regions of interest from one or more imaging planes". (const)	no
. <image_plane>	group	Group name is human-readable description of imaging plane	no
. . <roi_name>	group	Name of ROI	no
. . . img_mask	float32 array; dims: ['num_y', 'num_x']	ROI mask, represented in 2D ([y][x]) intensity image	yes
. . . pix_mask	uint16 array; dims: ['num_pixels', '2']	List of pixels (x,y) that compose the mask	yes
. . . pix_mask_weight	float32 array; dims: ['num_pixels']	Weight of each pixel listed in pix_mask	yes

. . . roi_description	text	Description of this ROI.	yes
. . description	text	Description of image plane, recording wavelength, depth, etc	recommended
. . imaging_plane_name	text	Name of imaging plane under general/optophysiology	yes
. . reference_images	group	Stores image stacks segmentation mask apply to.	yes
. . . <image_name>	<ImageSeries> (or subtype) (group)	One or more image stacks that the masks apply to (can be one-element stack)	yes
. . roi_list	text array; dims: ['num_rois']	List of ROIs in this imaging plane *(Automatically created)*	yes

ImagingRetinotopy

Intrinsic signal optical imaging or widefield imaging for measuring retinotopy. Stores orthogonal maps (e.g., altitude/azimuth; radius/theta) of responses to specific stimuli and a combined polarity map from which to identify visual areas.

Note: for data consistency, all images and arrays are stored in the format [row][column] and [row, col], which equates to [y][x]. Field of view and dimension arrays may appear backward (i.e., y before x).

ImagingRetinotopy includes all elements of *<Interface>* with the the following additions or changes:

Id	Type	Description	Required
ImagingRetinotopy	group	Top level group for ImagingRetinotopy.	yes
. help (attr)	text	Value is the string "Intrinsic signal optical imaging or Widefield imaging for measuring retinotopy". (const)	no
. axis_1_phase_map	float32 array; dims: ['num_rows', 'num_cols']	Phase response to stimulus on the first measured axis	yes
. . dimension (attr)	int32 array; dims: ['row_col']	Number of rows and columns in the image. NOTE: row, column representation is equivalent to height,width.	yes
. . field_of_view (attr)	float array; dims: ['row_col']	Size of viewing area, in meters	yes
. . unit (attr)	text	Unit that axis data is stored in (e.g., degrees)	yes
. axis_1_power_map	float32 array; dims: ['num_rows', 'num_cols']	Power response on the first measured axis. Response is scaled so 0.0 is no power in the response and 1.0 is maximum relative power.	recommended
. . dimension (attr)	int32 array; dims: ['row_col']	Number of rows and columns in the image. NOTE: row, column representation is equivalent to height,width.	yes
. . field_of_view (attr)	float array; dims: ['row_col']	Size of viewing area, in meters	recommended
. . unit (attr)	text	Unit that axis data is stored in (e.g., degrees)	yes
. axis_2_phase_map	float32 array; dims: ['num_rows', 'num_cols']	Phase response to stimulus on the second measured axis	yes
. . dimension (attr)	int32 array; dims: ['row_col']	Number of rows and columns in the image. NOTE: row, column representation is equivalent to height,width.	yes
. . field_of_view (attr)	float array; dims: ['row_col']	Size of viewing area, in meters	yes
. . unit (attr)	text	Unit that axis data is stored in (e.g., degrees)	yes
. axis_2_power_map	float32 array; dims: ['num_rows', 'num_cols']	Power response on the second measured axis. Response is scaled so 0.0 is no power in the response and 1.0 is maximum relative power.	recommended
. . dimension (attr)	int32 array; dims: ['row_col']	Number of rows and columns in the image. NOTE: row, column representation is equivalent to height,width.	yes
. . field_of_view (attr)	float array; dims: ['row_col']	Size of viewing area, in meters	recommended
. . unit (attr)	text	Unit that axis data is stored in (e.g., degrees)	yes
. axis_descriptions	text array; dims: ['2']	Two-element array describing the contents of the two response axis fields. Description should be something like ['altitude', 'azimuth'] or '['radius', 'theta']	yes
. focal_depth_image	uint16 array; dims: ['num_rows', 'num_cols']	Gray-scale image taken with same settings/parameters (e.g., focal depth, wavelength) as data collection. Array format: [rows][columns]	yes
. . bits_per_pixel (attr)	int32	Number of bits used to represent each value. This is necessary to determine maximum (white) pixel value	yes
. . dimension (attr)	int32 array; dims:	Number of rows and columns in the image. NOTE: row, column	yes

	['row_col']	representation is equivalent to height,width.	
. . field_of_view (attr)	float array; dims: ['row_col']	Size of viewing area, in meters	recommended
. . focal_depth (attr)	float	Focal depth offset, in meters	recommended
. . format (attr)	text	Format of image. Right now only 'raw' supported	yes
. sign_map	float32 array; dims: ['num_rows', 'num_cols']	Sine of the angle between the direction of the gradient in axis_1 and axis_2	yes
. . dimension (attr)	int32 array; dims: ['row_col']	Number of rows and columns in the image. NOTE: row, column representation is equivalent to height,width.	yes
. . field_of_view (attr)	float array; dims: ['row_col']	Size of viewing area, in meters.	recommended
. vasculature_image	uint16 array; dims: ['num_rows', 'num_cols']	Gray-scale anatomical image of cortical surface. Array structure: [rows][columns]	yes
. . bits_per_pixel (attr)	int32	Number of bits used to represent each value. This is necessary to determine maximum (white) pixel value	yes
. . dimension (attr)	int32 array; dims: ['row_col']	Number of rows and columns in the image. NOTE: row, column representation is equivalent to height,width.	yes
. . field_of_view (attr)	float array; dims: ['row_col']	Size of viewing area, in meters	recommended
. . format (attr)	text	Format of image. Right now only 'raw' supported	yes

Structured dimension(s):

Dimension	**Components** [name (unit)]	
row_col	row (meter)	column (meter)

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.

LFP includes all elements of *<Interface>* with the the following additions orchanges:

Id	Type	Description	Required
LFP	group	Top level group for LFP.	yes
. help (attr)	text	Value is the string "LFP data from one or more channels. Filter properties should be noted in the ElectricalSeries". (const)	no
. <ElectricalSeries> or subtype	group	<ElectricalSeries> or any subtype. Subtypes include:<SpikeEventSeries>	yes

MotionCorrection

An image stack where all frames are shifted (registered) to a common coordinate system, to account for movement and drift between frames. Note: each frame at each point in time is assumed to be 2-D (has only x & y dimensions).

MotionCorrection includes all elements of *<Interface>* with the the following additions or changes:

Id	Type	Description	Comment	Required
MotionCorrection	group	Top level group for MotionCorrection.		yes
. help (attr)	text	Value is the string "Image stacks whose frames have been shifted (registered) to account for motion". (const)		no
. <image stack name>	group	One of possibly many. Name should be informative.		yes
. . corrected	<ImageSeries> (or subtype) (group)	Image stack with frames shifted to the common coordinates.		yes
. . original	link; target type=<ImageSeries> (or subtype)	HDF5 Link to image series that is being registered.		yes
. . original_path	text	Path to linked original timeseries *(Automatically created)*		yes
. . xy_translation	<TimeSeries> (group)	Stores the x,y delta necessary to align each frame to the common coordinates, for example, to align each frame to a reference image.		yes
. . . data	float array; dims: ['num_times', 'xy']	TimeSeries for storing x,y offset for each image frame.		yes

. . . . conversion (attr)	float32!	Scalar to multiply each element in data to convert it to the specified unit		yes
. . . . 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. If unknown, use NaN	yes
. . . . 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.	yes

Structured dimension(s):

Dimension	Components [name (unit)]	
xy	x (pixels)	y (pixels)

Position

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

Position includes all elements of *<Interface>* with the the following additions orchanges:

Id	Type	Description	Required
Position	group	Top level group for Position.	yes
. help (attr)	text	Value is the string "Position data, whether along the x, xy or xyz axis". (const)	no
. <SpatialSeries> or subtype	group	<SpatialSeries> or any subtype.	yes

PupilTracking

Eye-tracking data, representing pupil size.

PupilTracking includes all elements of *<Interface>* with the the following additions orchanges:

Id	Type	Description	Required
PupilTracking	group	Top level group for PupilTracking.	yes
. help (attr)	text	Value is the string "Eye-tracking data, representing pupil size". (const)	no
. <TimeSeries>	group	General purpose time series.	yes

UnitTimes

Event times of observed units (e.g. 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).

UnitTimes includes all elements of *<Interface>* with the the following additions orchanges:

Id	Type	Description	Required
UnitTimes	group	Top level group for UnitTimes.	yes
. help (attr)	text	Value is the string "Estimated spike times from a single unit". (const)	no
. <unit_N>	group	Group storing times for <unit_N>.	yes
. . 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	recommended
. . times	float64! array; dims: ['num_events']	Spike time for the units (exact or estimated)	yes
. . unit_description	text	Description of the unit (eg, cell type).	yes
. unit_list	text array; dims: ['num_units']	List of units present. *(Automatically created)*	yes

File organization

Group: /acquisition

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

Id	Type	Description	Comment	Required

. images	group	Acquired images *(Automatically created)*		yes
. . <image_X>	binary array; dims: [['1d_size'], ['num_rows', 'num_cols']]	Photograph of experiment or experimental setup (video also OK).	Name is arbitrary. Data is stored as a single binary object (HDF5 opaque type).	no
. . . *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	recommended
. . . *format (attr)*	text	Format of the image.	eg, jpg, png, mpeg	yes
. 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. *(Automatically created)*	yes
. . <TimeSeries> or subtype	group	<TimeSeries> or any subtype. Subtypes include:<AbstractFeatureSeries>, <AnnotationSeries>, <CurrentClampSeries>, <CurrentClampStimulusSeries>, <ElectricalSeries>, <IZeroClampSeries>, <ImageMaskSeries>, <ImageSeries>, <IndexSeries>, <IntervalSeries>, <OpticalSeries>, <OptogeneticSeries>, <PatchClampSeries>, <RoiResponseSeries>, <SpatialSeries>, <SpikeEventSeries>, <TwoPhotonSeries>, <VoltageClampSeries>, <VoltageClampStimulusSeries>		yes

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.

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
No members or attributes specified for this group

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.

Id	Type	Description	Comment	Required
. *tags (attr)*	text array; dims: ['num_tags']	A sorted list of the different tags used by epochs.	This is a sorted list of all tags that are in any of the <epoch_X>/tags datasets`. *(Automatically created)*	yes
. <epoch_X>	group	One of possibly many different experimental epoch	Name is arbitrary but must be unique within the experiment.	no
. . *links (attr)*	text array; dims: ['num_links']	A sorted list mapping TimeSeries entries in the epoch to the path of the TimeSeries within the file. Each entry in the list has the following format: "'<*TimeSeries_X*>' **is** '*path_to_TimeSeries*'", where <*TimeSeries_X*> is the name assigned to group <TimeSeries_X> (below). Note that the name and path are both enclosed in single quotes and the word "is" (with a single space before and after) separate them. **Example list element:** "'auditory_cue' is '/stimulus/presentation/auditory_cue'". *(Automatically created)*		yes
. . *neurodata_type (attr)*	text	The string "Epoch"		yes
. . <timeseries_X>	group	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.	no
. . . count	int32	Number of data samples available in this time series, during this epoch.		yes
. . . idx_start	int32	Epoch's start index in TimeSeries data[] field.	This can be used to calculate location in TimeSeries timestamp[] field	yes

. . . timeseries	link; target type=<TimeSeries> (or subtype)	Link to TimeSeries. An HDF5 soft-link should be used.		yes
. . description	text	Description of this epoch (<epoch_X>).		no
. . start_time	float64!	Start time of epoch, in seconds		yes
. . stop_time	float64!	Stop time of epoch, in seconds		yes
. . tags	text array; dims: ['num_tags']	User-defined tags used throughout the epochs. Tags are to help identify or categorize epochs.	E.g., can describe stimulus (if template) or behavioral characteristic (e.g., "lick left")	no

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.

All entries in the below table are to be included when data is present. Unused groups (e.g., intracellular_ephys in an optophysiology experiment) should not be created unless there is data to store within them.

Id	Type	Description	Comment	Required
. __custom	int	Indicates that this group (general/) is the default location for custom nodes. This dataset in the format specification is just a flag. There is no actual data stored in the HDF5 file for this dataset.		no
. data_collection	text	Notes about data collection and analysis.	Can be from Methods	no
. devices	group	Description of hardware devices used during experiment.	Eg, monitors, ADC boards, microscopes, etc	no
. . <device_X>	text	One of possibly many. Information about device and device description.	Name should be informative. Contents can be from Methods.	no
. experiment_description	text	General description of the experiment.	Can be from Methods	recommended
. experimenter	text	Name of person who performed the experiment.	More than one person OK. Can specify roles of different people involved.	recommended
. extracellular_ephys	group	Metadata related to extracellular electrophysiology.		no
. institution	text	Institution(s) where experiment was performed		recommended
. intracellular_ephys	group	Metadata related to intracellular electrophysiology		no
. lab	text	Lab where experiment was performed		recommended
. notes	text	Notes about the experiment.	Things particular to this experiment	no
. optogenetics	group	Metadata describing optogenetic stimuluation		no
. optophysiology	group	Metadata related to optophysiology.		no
. pharmacology	text	Description of drugs used, including how and when they were administered.	Anesthesia(s), painkiller(s), etc., plus dosage, concentration, etc.	no
. protocol	text	Experimetnal protocol, if applicable.	E.g., include IACUC protocol	no
. related_publications	text	Publication information.	PMID, DOI, URL, etc. If multiple, concatenate together and describe which is which. such as PMID, DOI, URL, etc	no
. session_id	text	Lab-specific ID for the session.	Only 1 session_id per file, with all time aligned to experiment start time.	recommended
. slices	text	Description of slices, including information about preparation thickness, orientation, temperature and bath solution		no
. source_script	text	Script file used to create this NWB file.		no
. . *file_name (attr)*	text	Name of script file		no

. specifications	group	Group for storing format specification files.		no
. . <specification_file>	text	Dataset for storing contents of a specification file for either the core format or an extension. Name should match name of file.`		no
. . . *help (attr)*	text	Value is the string "Contents of format specification file.".		no
. . . *namespaces (attr)*	text array; dims: ['num_namespaces']	Namespaces defined in the file		yes
. stimulus	text	Notes about stimuli, such as how and where presented.	Can be from Methods	no
. subject	group	Information about the animal or person from which the data was measured.		no
. . age	text	Age of subject		no
. . description	text	Description of subject and where subject came from (e.g., breeder, if animal)		no
. . genotype	text	Genetic strain	If absent, assume Wild Type (WT)	no
. . sex	text	Gender of subject		no
. . species	text	Species of subject		no
. . subject_id	text	ID of animal/person used/participating in experiment (lab convention)		no
. . weight	text	Weight at time of experiment, at time of surgery and at other important times		no
. surgery	text	Narrative description about surgery/surgeries, including date(s) and who performed surgery.	Much can be copied from Methods	no
. virus	text	Information about virus(es) used in experiments, including virus ID, source, date made, injection location, volume, etc		no

Group: /general/extracellular_ephys

Metadata related to extracellular electrophysiology.

Id	Type	Description	Comment	Required
. <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_X and subgroups for electrode_group_X.	Name is arbitrary but should be meaningful.	yes
. . description	text	Description of probe or shank		yes
. . device	text	Name of device(s) in /general/devices		yes
. . 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	yes
. electrode_group	text array; dims: ['num_electrodes']	Identification string for probe, shank or tetrode each electrode resides on. Name should correspond to one of electrode_group_X 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.	yes
. electrode_map	number array; dims: ['num_electrodes', 'xyz']	Physical location of electrode, (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.	yes
. 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. If this changes between TimeSeries, filter description should be stored as	yes

			a text attribute for each TimeSeries.	
. impedance	text array; dims: ['num_electrodes']	Impedence of electrodes listed in electrode_map.	Text, in the event that impedance is stored as range and not a fixed value	yes

Structured dimension(s):

Dimension	Components [name (unit)]		
xyz	x (meter)	y (meter)	z (meter)

Group: /general/intracellular_ephys

Metadata related to intracellular electrophysiology

Id	Type	Description	Comment	Required
. <electrode_X>	group	One of possibly many.	Name should be informative.	yes
. . description	text	Recording description, description of electrode (e.g., whole-cell, sharp, etc)	Free-form text (can be from Methods)	yes
. . device	text	Name(s) of devices in general/devices		no
. . filtering	text	Electrode specific filtering.		no
. . initial_access_resistance	text	Initial access resistance		no
. . location	text	Area, layer, comments on estimation, stereotaxis coordinates (if in vivo, etc)		no
. . resistance	text	Electrode resistance	unit: Ohm	no
. . seal	text	Information about seal used for recording		no
. . slice	text	Information about slice used for recording		no
. 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.	no

Group: /general/optogenetics

Metadata describing optogenetic stimuluation

Id	Type	Description	Comment	Required
. <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	no
. . description	text	Description of site		yes
. . device	text	Name of device in /general/devices		yes
. . excitation_lambda	text	Excitation wavelength		yes
. . location	text	Location of stimulation site		yes

Group: /general/optophysiology

Metadata related to optophysiology.

Id	Type	Description	Comment	Required
. <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	no
. . <channel_X>	group	One of possibly many groups storing channel-specific data	Name is arbitrary but should be meaningful	yes
. . . description	text	Any notes or comments about the channel		yes
. . . emission_lambda	text	Emission lambda for channel		yes
. . description	text	Description of <image_plane_X>		no
. . device	text	Name of device in /general/devices		yes
. . excitation_lambda	text	Excitation wavelength		yes
. . imaging_rate	text	Rate images are acquired, in Hz.		yes
. . indicator	text	Calcium indicator		yes
. . location	text	Location of image plane		yes
. . manifold	float32 array; dims: ['height',	Physical position of each pixel.	"xyz" represents the position of the pixel relative to the defined coordinate space	recommended

	'weight', 'xyz']			
. . . *conversion (attr)*	float	Multiplier to get from stored values to specified unit (e.g., 1000 for millimeters)		yes
. . . *unit (attr)*	text	Base unit that coordinates are stored in (e.g., Meters)		yes
. . 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	recommended

Structured dimension(s):

Dimension	**Components** [name (unit)]		
xyz	x (Meter)	y (Meter)	z (Meter)

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.

Id	Type	Description	Required
. <Module>	group	Module. Name should be descriptive. Stores a collection of related data organized by contained interfaces. Each interface is a contract specifying content related to a particular type of data.	no

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.

Id	Type	Description	Comment	Required
. presentation	group	Stimuli presented during the experiment. *(Automatically created)*		yes
. . <TimeSeries> or subtype	group	<TimeSeries> or any subtype. Subtypes include:<AbstractFeatureSeries>, <AnnotationSeries>, <CurrentClampSeries>, <CurrentClampStimulusSeries>, <ElectricalSeries>, <IZeroClampSeries>, <ImageMaskSeries>, <ImageSeries>, <IndexSeries>, <IntervalSeries>, <OpticalSeries>, <OptogeneticSeries>, <PatchClampSeries>, <RoiResponseSeries>, <SpatialSeries>, <SpikeEventSeries>, <TwoPhotonSeries>, <VoltageClampSeries>, <VoltageClampStimulusSeries>		yes
. 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. *(Automatically created)*	yes
. . <TimeSeries> or subtype	group	<TimeSeries> or any subtype. Subtypes include:<AbstractFeatureSeries>, <AnnotationSeries>, <CurrentClampSeries>, <CurrentClampStimulusSeries>, <ElectricalSeries>, <IZeroClampSeries>, <ImageMaskSeries>, <ImageSeries>, <IndexSeries>, <IntervalSeries>, <OpticalSeries>, <OptogeneticSeries>, <PatchClampSeries>, <RoiResponseSeries>, <SpatialSeries>, <SpikeEventSeries>, <TwoPhotonSeries>, <VoltageClampSeries>, <VoltageClampStimulusSeries>		yes

Extending the format

The data organization presented in this document constitutes the *core* NWB format. Extensibility is handled by allowing users to store additional data as necessary using new datasets, attributes or groups. There are two ways to document these additions. The first is to add an attribute "neurodata_type" with value the string "Custom" to the additional groups or datasets, and provide documentation to describe the extra data if it is not clear from the context what the data represent. This method is simple but does not include a consistant way to describe the additions. The second method is to write an *extension* to the format. With this method, the additions are describe by the extension and attribute "schema_id" is set to the schema_id associated with the extension. Extensions to the format are written using the same specification language that is used to define the core format. Creating an extension allows adding the new data to the file through the API, validating files containing extra data, and also generating documentation for the

additions. Popular extensions can be proposed and added to the official format specification. Writing and using extensions are described in the API documentation. Both methods allow 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 Buzsaki Lab at New York University School of Medicine, and the University of California, Berkeley. Ovation.io is our founding development partner. Ken Harris at University College London provided invaluable input and advice.

Change history

1.0.6, April 8, 2017
Minor fixes:

- Modify <IntervalSeries>/ documentation to use html entities for < and >.
- Fix indentation of unit attribute data_type, and conversion attribute description in /general/optophysiology/<imaging_plane_X>/manifold.
- Fix typos in <AnnotationSeries>/ conversion, resolution and unit attributes.
- Update documentation for IndexSeries to reflect more general usage.
- Change to all numerical version number to remove warning message when installing using setuptools.

1.0.5i_beta, Dec 6, 2016
Remove some comments. Modify author string in info section.

1.0.5h_beta, Nov 30, 2016
Add dimensions to /acquisition/images/<image_X>

1.0.5g_beta, Oct 7, 2016
Replace group options: autogen: {"type": "create"} and "_closed": True with "_properties": {"create": True} and "_properties": {"closed": True}. This done to make the specification language more consistent by having these group properties specified in one place ("_properties" dictionary).

1.0.5f_beta, Oct 3, 2016
Minor fixes to allow validation of schema using json-schema specification in file "meta-schema.py" using utility "check_schema.py".

1.0.5e_beta, Sept 22, 2016
Moved definition of <Module>/ out of /processing group to allow creating subclasses of Module. This is useful for making custom Module types that specified required interfaces. Example of this is in python-api/examples/create_scripts/module-e.py and the extension it uses (extensions/e-module.py).
Fixed malformed html in nwb_core.py documentation.
Changed html generated by doc_tools.py to html5 and fixed so passes validation at https://validator.w3.org.

1.0.5d_beta, Sept 6, 2016
Changed ImageSeries img_mask dimensions to:
"dimensions": ["num_y","num_x"]
to match description.

1.0.5c_beta, Aug 17, 2016

Change IndexSeries to allow linking to any form of TimeSeries, not just an ImageSeries

1.0.5b_beta, Aug 16, 2016

- Make 'manifold' and 'reference_frame' (under /general/optophysiology) recommended rather than required.
- In all cases, allow subclasses of a TimeSeries to fulfill validation requirements when an instance of TimeSeries is required.
- Change unit attributes in VoltageClampSeries series datasets from required to recommended.
- Remove 'const'=True from TimeSeries attributes in AnnotationSeries and IntervalSeries.
- Allow the base TimeSeries class to store multi-dimensional arrays in 'data'. A user is expected to describe the contents of 'data' in the comments and/or description fields.

1.0.5a_beta, Aug 10, 2016
Expand class of Ids allowed in TimeSeries missing_fields attribute to allow custom uses.

1.0.5_beta Aug 2016
Allow subclasses to be used for merges instead of base class (specified by 'merge+' in format specification file).
Use 'neurodata_type=Custom' to flag additions that are not describe by a schema.
Exclude TimeSeries timestamps and starting time from under /stimulus/templates

1.0.4_beta June 2016
Generate documentation directly from format specification file."
Change ImageSeries external_file to an array.
Made TimeSeries description and comments recommended.

1.0.3 April, 2016
Renamed "ISI_Retinotopy" to "ISIRetinotopy"
Change ImageSeries external_file to an array. Added attribute starting_frame.
Added IZeroClampSeries.

1.0.2 February, 2016
Fixed documentation error, updating 'neurodata_version' to 'nwb_version'
Created ISI_Retinotopy interface
In ImageSegmentation module, moved pix_mask::weight attribute to be its own dataset, named pix_mask_weight. Attribute proved inadequate for storing sufficiently large array data for some segments
Moved 'gain' field from Current/VoltageClampSeries to parent PatchClampSeries, due need of stimuli to sometimes store gain
Added Ken Harris to the Acknowledgements section

1.0.1 October 7th, 2015
Added 'required' field to tables in the documentation, to indicate if group/dataset/attribute is required, standard or optional
Obsoleted 'file_create_date' attribute 'modification_time' and made file_create_date a text array
Removed 'resistance_compensation' from CurrentClampSeries due being duplicate of another field
Upgraded TwoPhotonSeries::imaging_plane to be a required value
Removed 'tags' attribute to group 'epochs' as it was fully redundant with the 'epoch/tags' dataset
Added text to the documentation stating that specified sizes for integer values are recommended sizes, while sizes for floats are minimum sizes
Added text to the documentation stating that, if the TimeSeries::data::resolution attribute value is unknown then store a NaN
Declaring the following groups as required (this was implicit before)

acquisition/
_ images/
_ timeseries/
analysis/
epochs/
general/
processing/
stimulus/
_ presentation/
_ templates/

This is to ensure consistency between .nwb files, to provide a minimum expected structure, and to avoid confusion by having someone expect time series to be in places they're not. I.e., if 'acquisition/timeseries' is not present, someone might reasonably expect that acquisition time series might reside in 'acquisition/'. It is also a subtle reminder about what the file is designed to store, a sort of built-in documentation. Subfolders in 'general/' are only to be included as needed. Scanning 'general/' should provide the user a quick idea what the experiment is about, so only domain-relevant subfolders should be present (e.g., 'optogenetics' and 'optophysiology'). There should always be a 'general/devices', but it doesn't seem worth making it mandatory without making all subfolders mandatory here.

1.0.0 September 28th, 2015
Convert document to .html
TwoPhotonSeries::imaging_plane was upgraded to mandatory to help enforce inclusion of important metadata in the file.

Design notes

The listed size of integers is the suggested size. What's important for integers is simply that the integer is large enough to store the required data, and preferably not larger. For floating point, double is required for timestamps, while floating point is largely sufficient for other uses. This is why doubles (float64) are stated in some places. Because floating point sizes are provided, integer sizes are provided as well.

Why do timestamps_link and data_link record linking between datasets, but links between epochs and timeseries are not recorded?
Epochs have a hardlink to entire timeseries (ie, the HDF5 group). If 100 epochs link to a time series, there is only one time series. The data and timestamps within it are not shared anywhere (at least from the epoch linking). An epoch is an entity that is put in for convenience and annotation so there isn't necessarily an important association between what epochs link to what time series (all epochs could link to all time series).
The timestamps_link and data_link fields refer to links made between time series, such as if timeseries A and timeseries B, each having different data (or time) share time (or data). This is much more important information as it shows structural associations in the data.