NWB Format Specification

Release 2.8.0-alpha

unknown

Apr 25, 2024

TABLE OF CONTENTS

Part I

Introduction

CHAPTER

ONE

OVERVIEW

The NWB Format is a core component of the Neurodata Without Borders (NWB) project. The NWB format is designed to store general optical and electrical physiology data in a way that is both understandable to humans as well as accessible to programmatic interpretation. 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.

The NWB format uses the following main primitives to hierarchically organize neuroscience data :

- A *Group* is similar to a folder and may contain an arbitrary number of other groups and datasets,
- A *Dataset* describes an n-dimensional array and provides the primary means for storing data,
- An *Attribute** is a small dataset that is attached to a specific group or dataset and is typically used to store metadata specific to the object they are associated with, and
- A *Link* is a reference to another group or dataset.

The NWB format is formally described via formal specification documents using the NWB specification language . HDF5 currently serves as the main format for storing data in the NWB format (see http://nwb-storage.readthedocs.io/en/latest/ for details). The PyNWB API is available to enable users to efficiently interact with NWB format files.

The NWB format uses a modular design in which all main semantic components of the format have a unique *neurodata_type* (similar to a Class in object-oriented design)(Section 1.1). This allows for reuse and extension of types through inclusion and inheritance. All datasets and groups in the format can be uniquely identified by either their name and/or *neurodata_type*.

Two important base types in the NWB format are *NWBContainer* and *TimeSeries*. *NWBContainer* defines a generic container for storing collection of data and is used to define common features and functionality across data containers (see Section 1.2). *TimeSeries* is a central component in the NWB format for storing complex temporal series (see Section 1.3). In the format, these types are then extended to define more specialized types. To organize and define collections of processed data from common data processing steps, the NWB format then defines the concept of *ProcessingModule* where each processing step is represented by a corresponding *NWBDataInterface* (an extension of *NWBContainer*) (see Section 1.4 for details).

At a high level, data is organized into the following main groups:

- *acquisition/* : data streams recorded from the system, including ephys, ophys, tracking, etc.,
- *intervals/* : experimental intervals,
- *stimulus/* : stimulus data,
- *general/* : experimental metadata, including protocol, notes and description of hardware device(s).
- *processing/* : standardized processing modules, often as part of intermediate analysis of data that is necessary to perform before scientific analysis,
- *analysis/* : lab-specific and custom scientific analysis of data.

The high-level data organization within NWB files is described in detail in Section 4.5.2. The top-level datasets and attributes are described in Table 4.22 and the top-level organization of data into groups is described in Table 4.23.

1.1 `neurodata_type` : Assigning types to specifications

The concept of a *neurodata_type* is similar to the concept of a Class in object-oriented programming. In the NWB format, groups or datasets may be given a unique *neurodata_type*. The *neurodata_type* allows the unique identification of the type of objects in the format and also enable the reuse of types through the concept of inheritance. A group or dataset may, hence, define a new *neurodata_type* while extending an existing type. E.g., *AbstractFeatureSeries* defines a new type that inherits from *TimeSeries*.

1.2 `NWBContainer`, `NWBData`, `NWBDataInterface`: Base neurodata_types for containers and datasets

NWBContainer is a specification of a group that defines a generic container for storing collections of data. *NWBContainer* serves as the base type for all main data containers (including *TimeSeries*) of the core NWB data format and allows us to define and integrate new common functionality in a central place and via common mechanisms (see Section 4.1.5).

NWBDataInterface extends *NWBContainer* and serves as base type for primary data (e.g., experimental or analysis data) and is used to distinguish in the schema between non-metadata data containers and metadata containers (see Section 4.1.6).

NWBData is a specification of a Dataset that functions as a common base neurodata_type for datasets with an assigned *neurodata_type* (see Section 4.1.1).

Note: The concept of *NWBContainer* and *NWBData* have been introduced in NWB 2. *NWBDataInterface* (also introduced in NWB 2) replaces `Interface` from NWB 1.x. `Interface` was renamed to *NWBDataInterface* to ease intuition and the concept was generalized via *NWBContainer* to provide a common base for data containers (rather than being specific to *ProcessingModules* as in NWB 1.x).

1.3 `Time Series` : A base neurodata_type for storing time series data

The file format is designed around a data structure called a *TimeSeries* which stores time-varying data. A *TimeSeries* is a superset of several neurodata_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 (i.e., subclassed) to account for different modalities and data storage requirements (see Section 1.5).

Each *TimeSeries* has its own HDF5 group, and all datasets belonging to a *TimeSeries* are in that group. In particular, a *TimeSeries* defines components to store *data* and *time*.

The *data* element in the *TimeSeries* will typically be an array of any valid HDF5 data type (e.g., a multi-dimensional 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 support provides two time objects representations. 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 time-base and corrected relative to the master clock. This approach allows the NWB format to support streaming of data directly from hardware sources.

In addition to data and time, the *TimeSeries* group can be used to store additional information 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 via the concept of extensions. It should be noted that such lab-specific extensions may not be recognized by analysis tools/scripts existing outside the lab. Extensions are described in section (see Section 1.5).

1.4 Data Processing Modules: Organizing processed data

NWB uses *ProcessingModule* 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. Processing modules store the data used by software tools to calculate these intermediate results. All processing modules are stored directly in the group */processing*. The name of each module is chosen by the data provider (i.e. processing modules have a "variable" name). The particular data within each processing module is specified by one or more *NWBDataInterface*, which are groups residing directly within a processing module. Each NWBDataInterface has a unique neurodata_type (e.g., *ImageSegmentation*) that describes and defines the data contained in the NWBDataInterface. For NWBDataInterfaces designed for use with processing modules, a default name (usually the same as the neurodata_type) is commonly specified to further ease identification of the data in a file. However, to support storage of multiple instances of the same subtype in the same processing module, NWB allows users to optionally define custom names as well.

1.5 Extending the format

The data organization presented in this document constitutes the *core* NWB format. Extensibility is handled via the concept of extensions, allowing users to extend (i.e., add to) existing and create new neurodata_types definitions for storing custom data. To avoid collisions between extensions, extensions are defined as part of custom namespaces (which typically import the core NWB namespace). Extensions to the format are written using the Specification Language . To ease development of extensions, the PyNWB (and HDMF used by PyNWB) API provides dedicated data structures that support programmatic creation and use of extensions. An example for extending NWB using PyNWB is available at https://pynwb.readthedocs.io/en/stable/tutorials/general/extensions.html and additional details are also available as part of the PyNWB tutorials https://pynwb.readthedocs.io/en/stable/tutorials/index.html .

Creating extensions allows adding and documenting new data to NWB, interaction with custom data via the API, validation of custom data contents, sharing and collaboration of extensions and data. Popular extensions may be proposed and added to the official format specification after community discussion and review. To propose a new extensions for the NWB core format you may file an issue at https://github.com/NeurodataWithoutBorders/nwb-schema/issues .

1.5.1 Extending TimeSeries and NWBContainer

Like any other neurodata_type, *TimeSeries* can be extended via extensions by defining corresponding derived neurodata_types. This is typically done to to represent more narrowly focused modalities (e.g., electrical versus optical physiology) as well as new modalities (e.g., video tracking of whisker positions). When a neurodata_type inherits from *TimeSeries*, new data objects (i.e., datasets, attributes, groups, and links) can be added while all objects of the parent *TimeSeries* type are inherited and, hence, part of the new neurodata_type. Section Section 4.1.7 includes a list of all *TimeSeries* types.

Extending *NWBContainer* works in the same way, e.g., to create more specific types for data processing.

1.6 Common attributes

All NWB Groups and Datasets with an assigned neurodata_type have three required attributes: *neurodata_type*, *namespace*, and *object_id*.

- `neurodata_type` (variable-length string) is the name of the NWB primitive that this group or dataset maps onto
- `namespace` (variable-length string) is the namespace where `neurodata_type` is defined, e.g. "core" or the namespace of an extension
- `object_id` (variable-length string) is a universally unique identifier for this object within its hierarchy. It should be set to the string representation of a random UUID version 4 value (see RFC 4122) upon first creation. It is **not** a hash of the data. Files that contain the exact same data but were generated in different instances will have different `object_id` values. Currently, modification of an object does not require its `object_id` to be changed.

CHAPTER

TWO

COMMENTS AND DEFINITIONS

2.1 Notation

The description of the format is divided into subsection based on *neurodata_type*. Each *neurodata_type* section includes:

- A basic description of the type
- An optional figure describing the organization of data within the type
- A set of tables describing the datasets, attributes and groups contained in the type.
- An optional set of further subsections describing the content of subgroups contained in the given *neurodata_type*.

In the tables we use the following notation in the **Id** column to uniquely identify datasets, groups, attributes:

- `` `name` `` describes the unique name of an object
- `` `<neurodata_type>` `` describes the `` `neurodata_type` `` of the object in case that the object does not have a unique name
- `` `-` `` prefixes are used to indicate the depth of the object in the hierarchy to allow identification of the parent of the object. E.g., an object with a `` `--` `` prefix will belong to the previous object with a `` `-` `` prefix.

Here a quick example:

Table 2.1: Example illustrating the description of the contents of `` `neurodata_types` ``.

Id	Type	Description
<MyTimeSeries>	group	Top level group for the neurodata_type. The group the neurodata_type *MyTimeSeries* but no fixed name
—myattr	attribute	Attribute with the fixed name myattr defined on <MyTimeSeries>
—mydata	dataset	Required dataset with a unique name contained in <MyTimeSeries>
——unit	attribute	Attribute unit defined on the dataset .mydata
—myotherdata	dataset	Optional dataset with a unique name contained in <MyTimeSeries>
—<ElectricalSeries>	group	Optional set of groups with the neurodata_type ElectricalSeries that are contained in <MyTimeSeries>

2.2 Storing Time Values

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 specified for many datatypes (e.g., float32). For fields with a specified size, larger sizes can be used, so long as the selected size encompasses the full range of data, and for floats, without loss of significant precision.

2.3 Links and data references

Soft Links: In some instances, the specification refers to links. When links are made within a file, NWB uses soft-links and not hard-links. 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), 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.

Object References: Object references are similar to SoftLinks but instead of being stored as elements in the data hierarchy (similar to a Group or Dataset) the object reference defines a data type on a Dataset, i.e., the object references are stored as elements in a Dataset.

Region References: Region references are similar to object references but instead of pointing to another Group/Dataset as a whole a region reference defines an additional selection and as such describe a reference to a subset of a dataset.

2.4 Design notes

Data type sizes

The listed size of integers and floating point values is the minimum size. The size of the data type should be large enough to store the required data, and preferably not larger. 64-bit floating point (double) is required for timestamps, while 32-bit floating point is largely sufficient for other uses.

Extra fields

All parts of an NWB file should be governed by either the core schema or defined in a neurodata extension (NDX). *Extra fields* are any datasets, attributes, groups, links etc. that are included in a file but which are not described by the NWB schema or a neurodata extension (NDX). Extra fields are not considered part of the NWB file and as such, any NWB API may ignore extra fields. For API's this specifically means:

- an NWB file that includes extra fields should be readable by the API as long as the file is otherwise valid,
- an API is permitted to ignore extra fields on read,
- an API is permitted to ignore (including remove) extra fields on write.

In practice, the use of extra fields is highly discouraged and instead neurodata extensions (NDX) should be used to extend NWB to include additional fields if necessary.

Why do timestamps_link and data_link record linking between datasets, but links between epochs and timeseries are not recorded?

Epochs have a link to entire timeseries (i.e., 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.

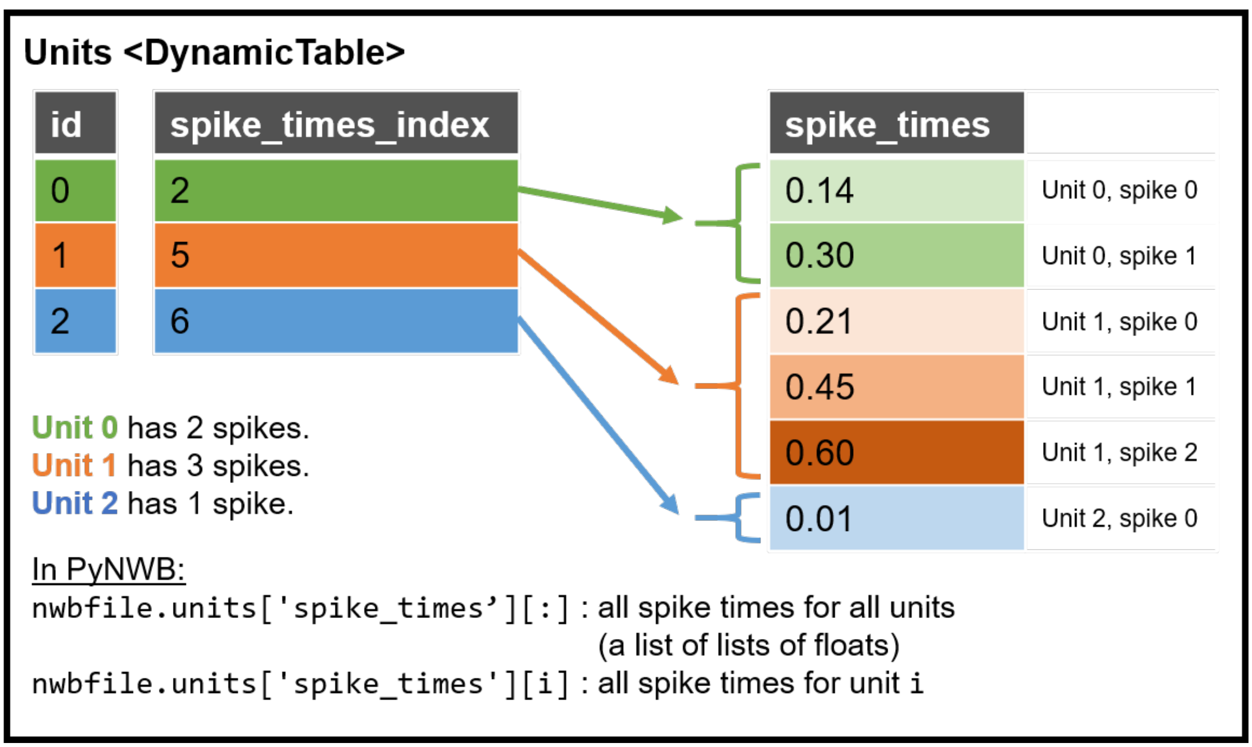
2.5 Tables and ragged arrays

The NWB schema includes several tables, such as for storing data/metadata about trials, epochs, single units and multi-units, electrodes, and ROIs. All of the tables in NWB derive from the base data type, DynamicTable. DynamicTable is a column-based representation of a table that allows users to add custom columns (of type VectorData) that are not pre-defined in the specification. This is useful for handling types of data where every experiment or lab may want to store information unique to that experiment or lab, e.g., metadata related to the trials in a session or spike sorting metrics.

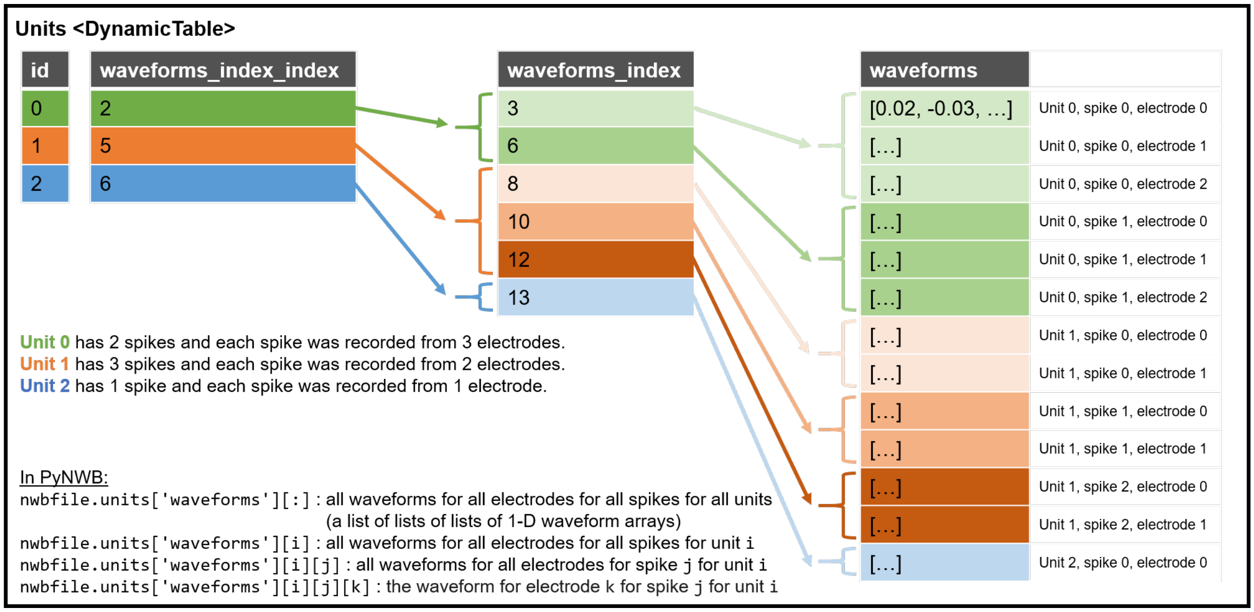
DynamicTable objects typically contain columns that are of equal length, where the i-th element of a column corresponds to the i-th element of all of the other columns. In other words, each row has a single item in each column. However, in some situations, users may wish to store and associate multiple items in a single column for each row. For example, in the Units table, each row represents a single sorted unit and each unit has multiple spike times associated with it, where the number of spike times differs between units (rows). This is sometimes called a ragged array or jagged array.

Ragged array columns can be created by creating a primary VectorData column that contains all of the data values (e.g., spike times) and creating a secondary VectorIndex column that contains a mapping from rows to elements of its target VectorData column. The VectorIndex column has the same number of elements (rows) as the rest of the table.

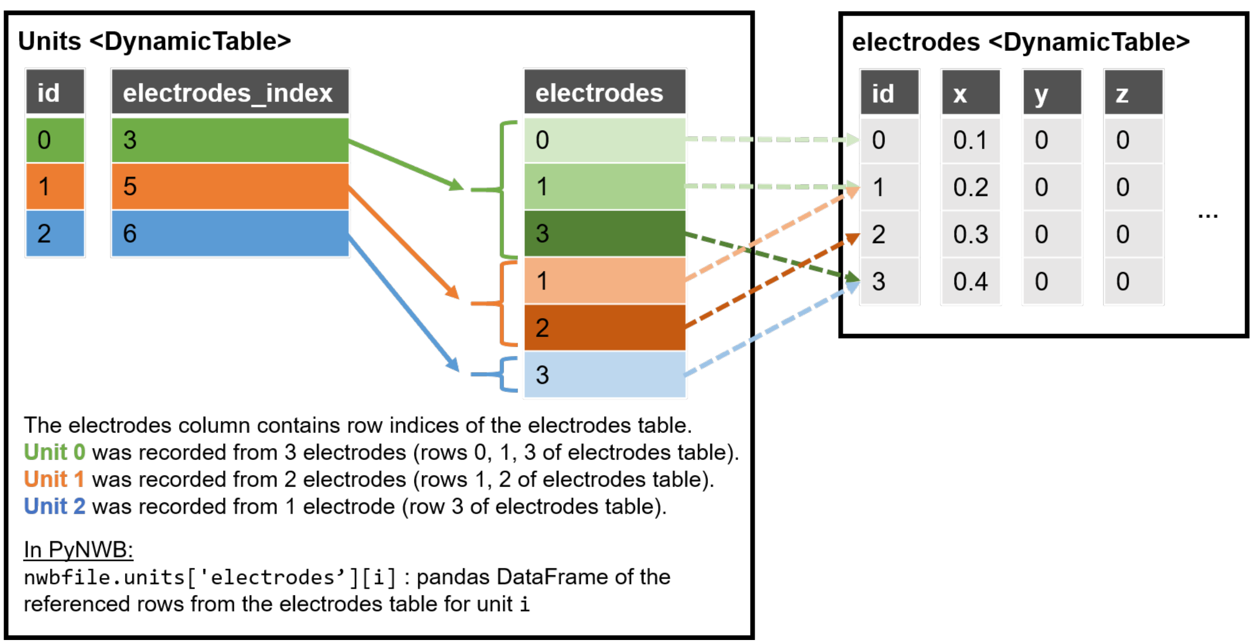
The values of the VectorIndex column follow the mapping such that the data associated with the first row is at VectorData[0:VectorIndex[0]], and the data associated with the second row is at VectorData[VectorIndex[0]:VectorIndex[1]], and so on.



2.6 Doubly ragged arrays



2.7 References to rows of a table



Part II

Format Specification

Version 2.8.0-alpha Apr 25, 2024

CHAPTER

THREE

FORMAT OVERVIEW

3.1 Namespace – NWB core

- **Description:** NWB namespace
- **Name:** core
- **Full Name:** NWB core
- **Version:** 2.6.0-alpha
- **Authors:**
  - Andrew Tritt
  - Oliver Ruebel
  - Ryan Ly
  - Ben Dichter
  - Keith Godfrey
  - Jeff Teeters
- **Contacts:**
  - ajtritt@lbl.gov
  - oruebel@lbl.gov
  - rly@lbl.gov
  - bdichter@lbl.gov
  - keithg@alleninstitute.org
  - jteeters@berkeley.edu
- **Schema:**
  - **namespace:** hdmf-common
  - **doc:** This source module contains base data types used throughout the NWB data format.
  - **source:** nwb.base.yaml
  - **title:** Base data types
  - **doc:** This source module contains neurodata_types for device data.
  - **source:** nwb.device.yaml
  - **title:** Devices

- **doc:** This source module contains neurodata_types for epoch data.
- **source:** nwb.epoch.yaml
- **title:** Epochs
- **doc:** This source module contains neurodata_types for image data.
- **source:** nwb.image.yaml
- **title:** Image data
- **doc:** Main NWB file specification.
- **source:** nwb.file.yaml
- **title:** NWB file
- **doc:** Miscellaneous types.
- **source:** nwb.misc.yaml
- **title:** Miscellaneous neurodata_types.
- **doc:** This source module contains neurodata_types for behavior data.
- **source:** nwb.behavior.yaml
- **title:** Behavior
- **doc:** This source module contains neurodata_types for extracellular electrophysiology data.
- **source:** nwb.ecephys.yaml
- **title:** Extracellular electrophysiology
- **doc:** This source module contains neurodata_types for intracellular electrophysiology data.
- **source:** nwb.icephys.yaml
- **title:** Intracellular electrophysiology
- **doc:** This source module contains neurodata_types for opto-genetics data.
- **source:** nwb.ogen.yaml
- **title:** Optogenetics
- **doc:** This source module contains neurodata_types for optical physiology data.
- **source:** nwb.ophys.yaml
- **title:** Optical physiology
- **doc:** This source module contains neurodata_type for retinotopy data.
- **source:** nwb.retinotopy.yaml
- **title:** Retinotopy

CHAPTER

FOUR

TYPE SPECIFICATIONS

4.1 Base data types

This source module contains base data types used throughout the NWB data format.

4.1.1 NWBData

Overview: An abstract data type for a dataset.

`NWBData` extends `Data` and includes all elements of Data with the following additions or changes.

- **Extends:** Data
- **Primitive Type:** Dataset
- **Inherits from:** Data
- **Subtypes:** *Image*, *RGBAImage*, *ImageReferences*, *ScratchData*, *RGBImage*, *GrayscaleImage*
- **Source filename:** nwb.base.yaml
- **Source Specification:** see Section 5.2.1

4.1.2 TimeSeriesReferenceVectorData

Overview: Column storing references to a TimeSeries (rows). For each TimeSeries this VectorData column stores the start_index and count to indicate the range in time to be selected as well as an object reference to the TimeSeries.

`TimeSeriesReferenceVectorData` extends `VectorData` and includes all elements of VectorData with the following additions or changes.

- **Extends:** VectorData
- **Primitive Type:** Dataset
- **Data Type: Compound data type with the following elements:**
  - **idx_start:** Start index into the TimeSeries 'data' and 'timestamp' datasets of the referenced TimeSeries. The first dimension of those arrays is always time. (*dtype*= int32)
  - **count:** Number of data samples available in this time series, during this epoch (*dtype*= int32)
  - **timeseries:** The TimeSeries that this index applies to (*dtype*= object reference to *TimeSeries*)
- **Default Name:** timeseries
- **Inherits from:** VectorData, Data
- **Source filename:** nwb.base.yaml
- **Source Specification:** see Section 5.2.2

Table 4.1: Datasets, Links, and Attributes contained in <TimeSeriesReferenceVectorData>

Id	Type	Description
<TimeSeriesReferenceVectorData>	Dataset	Top level Dataset for <TimeSeriesReferenceVectorData> • **Neurodata Type:** TimeSeriesReferenceVectorData • **Extends:** VectorData • **Data Type: Compound data type with the following elements:** – **idx_start:** Start index into the TimeSeries 'data' and 'timestamp' datasets of the referenced TimeSeries. The first dimension of those arrays is always time. (*dtype*= int32) – **count:** Number of data samples available in this time series, during this epoch (*dtype*= int32) – **timeseries:** The TimeSeries that this index applies to (*dtype*= object reference to *TimeSeries*) • **Default Name:** timeseries
—description	Attribute	Description of what these vectors represent. • **Data Type:** text • **Name:** description

4.1.3 Image

Overview: An abstract data type for an image. Shape can be 2-D (x, y), or 3-D where the third dimension can have three or four elements, e.g. (x, y, (r, g, b)) or (x, y, (r, g, b, a)).

`Image` extends `NWBData` and includes all elements of *NWBData* with the following additions or changes.

- **Extends:** *NWBData*
- **Primitive Type:** Dataset
- **Data Type:** numeric
- **Dimensions:** [['x', 'y'], ['x', 'y', 'r, g, b'], ['x', 'y', 'r, g, b, a']]
- **Shape:** [[None, None], [None, None, 3], [None, None, 4]]
- **Inherits from:** *NWBData*, Data
- **Subtypes:** *RGBAImage*, *GrayscaleImage*, *RGBImage*
- **Source filename:** nwb.base.yaml
- **Source Specification:** see Section 5.2.3

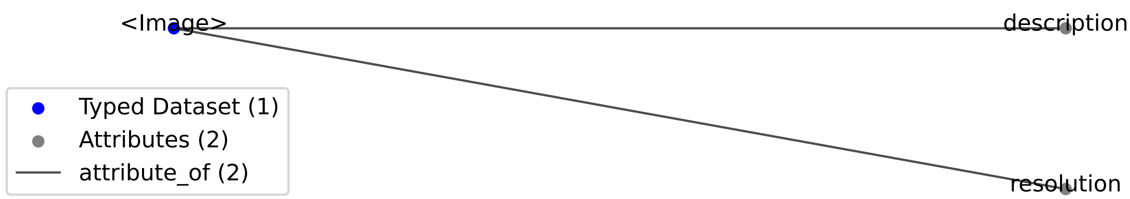


Table 4.2: Datasets, Links, and Attributes contained in <Image>

Id	Type	Description
<Image>	Dataset	Top level Dataset for <Image> • **Neurodata Type:** Image • **Extends:** *NWBData* • **Data Type:** numeric • **Dimensions:** [['x', 'y'], ['x', 'y', 'r, g, b'], ['x', 'y', 'r, g, b, a']] • **Shape:** [[None, None], [None, None, 3], [None, None, 4]]
—resolution	Attribute	Pixel resolution of the image, in pixels per centimeter. • **Data Type:** float32 • **Required:** False • **Name:** resolution
—description	Attribute	Description of the image. • **Data Type:** text • **Required:** False • **Name:** description

4.1.4 ImageReferences

Overview: Ordered dataset of references to Image objects.

`ImageReferences` extends `NWBData` and includes all elements of *NWBData* with the following additions or changes.

- **Extends:** *NWBData*
- **Primitive Type:** Dataset
- **Data Type:** object reference to *Image*
- **Dimensions:** ['num_images']
- **Shape:** [None]
- **Inherits from:** *NWBData*, Data
- **Source filename:** nwb.base.yaml
- **Source Specification:** see Section 5.2.4

4.1.5 NWBContainer

Overview: An abstract data type for a generic container storing collections of data and metadata. Base type for all data and metadata containers.

`NWBContainer` extends `Container` and includes all elements of Container with the following additions or changes.

- **Extends:** Container
- **Primitive Type:** Group
- **Inherits from:** Container
- **Subtypes:** *ElectricalSeries*, *EventDetection*, *ImagingPlane*, *OptogeneticSeries*, *DecompositionSeries*, *AbstractFeatureSeries*, *Position*, *ImageSegmentation*, *LabMetaData*, *ImageSeries*, *OpticalSeries*, *BehavioralEpochs*, *OptogeneticStimulusSite*, *CorrectedImageStack*, *Device*, *IndexSeries*, *MotionCorrection*, *Images*, *ElectrodeGroup*, *BehavioralEvents*, *FeatureExtraction*, *Fluorescence*, *RoiResponseSeries*, *LFP*, *FilteredEphys*, *BehavioralTimeSeries*, *AnnotationSeries*, *CurrentClampSeries*, *EyeTracking*, *ImageMaskSeries*, *SpatialSeries*, *NWBDataInterface*, *CurrentClampStimulusSeries*, *OpticalChannel*, *CompassDirection*, *PatchClampSeries*, *ImagingRetinotopy*, *NWBFile*, *SpikeEventSeries*, *TwoPhotonSeries*, *VoltageClampSeries*, *IntracellularElectrode*, *Clustering*, *TimeSeries*, *VoltageClampStimulusSeries*, *DfOverF*, *IZeroClampSeries*, *IntervalSeries*, *ProcessingModule*, *ClusterWaveforms*, *Subject*, *PupilTracking*, *OnePhotonSeries*, *EventWaveform*
- **Source filename:** nwb.base.yaml
- **Source Specification:** see Section 5.2.5

4.1.6 NWBDataInterface

Overview: An abstract data type for a generic container storing collections of data, as opposed to metadata.

`NWBDataInterface` extends `NWBContainer` and includes all elements of *NWBContainer* with the following additions or changes.

- **Extends:** *NWBContainer*
- **Primitive Type:** Group
- **Inherits from:** *NWBContainer*, Container
- **Subtypes:** *ElectricalSeries*, *EventDetection*, *OptogeneticSeries*, *DecompositionSeries*, *AbstractFeatureSeries*, *Position*, *ImageSegmentation*, *ImageSeries*, *OpticalSeries*, *BehavioralEpochs*, *CorrectedImageStack*, *IndexSeries*, *MotionCorrection*, *Images*, *BehavioralEvents*, *FeatureExtraction*, *Fluorescence*, *RoiResponseSeries*, *LFP*, *FilteredEphys*, *BehavioralTimeSeries*, *AnnotationSeries*, *CurrentClampSeries*, *EyeTracking*, *ImageMaskSeries*, *SpatialSeries*, *CurrentClampStimulusSeries*, *CompassDirection*, *PatchClampSeries*, *ImagingRetinotopy*, *SpikeEventSeries*, *TwoPhotonSeries*, *VoltageClampSeries*, *Clustering*, *TimeSeries*, *VoltageClampStimulusSeries*, *DfOverF*, *IZeroClampSeries*, *IntervalSeries*, *ClusterWaveforms*, *PupilTracking*, *OnePhotonSeries*, *EventWaveform*
- **Source filename:** nwb.base.yaml
- **Source Specification:** see Section 5.2.6

4.1.7 TimeSeries

Overview: General purpose time series.

`TimeSeries` extends `NWBDataInterface` and includes all elements of *NWBDataInterface* with the following additions or changes.

- **Extends:** *NWBDataInterface*
- **Primitive Type:** Group
- **Inherits from:** *NWBDataInterface*, *NWBContainer*, Container
- **Subtypes:** *ElectricalSeries*, *OptogeneticSeries*, *DecompositionSeries*, *AbstractFeatureSeries*, *ImageSeries*, *OpticalSeries*, *IndexSeries*, *RoiResponseSeries*, *AnnotationSeries*, *CurrentClampSeries*, *ImageMaskSeries*, *SpatialSeries*, *CurrentClampStimulusSeries*, *PatchClampSeries*, *SpikeEventSeries*, *TwoPhotonSeries*, *VoltageClampSeries*, *VoltageClampStimulusSeries*, *IZeroClampSeries*, *IntervalSeries*, *OnePhotonSeries*
- **Source filename:** nwb.base.yaml
- **Source Specification:** see Section 5.2.7

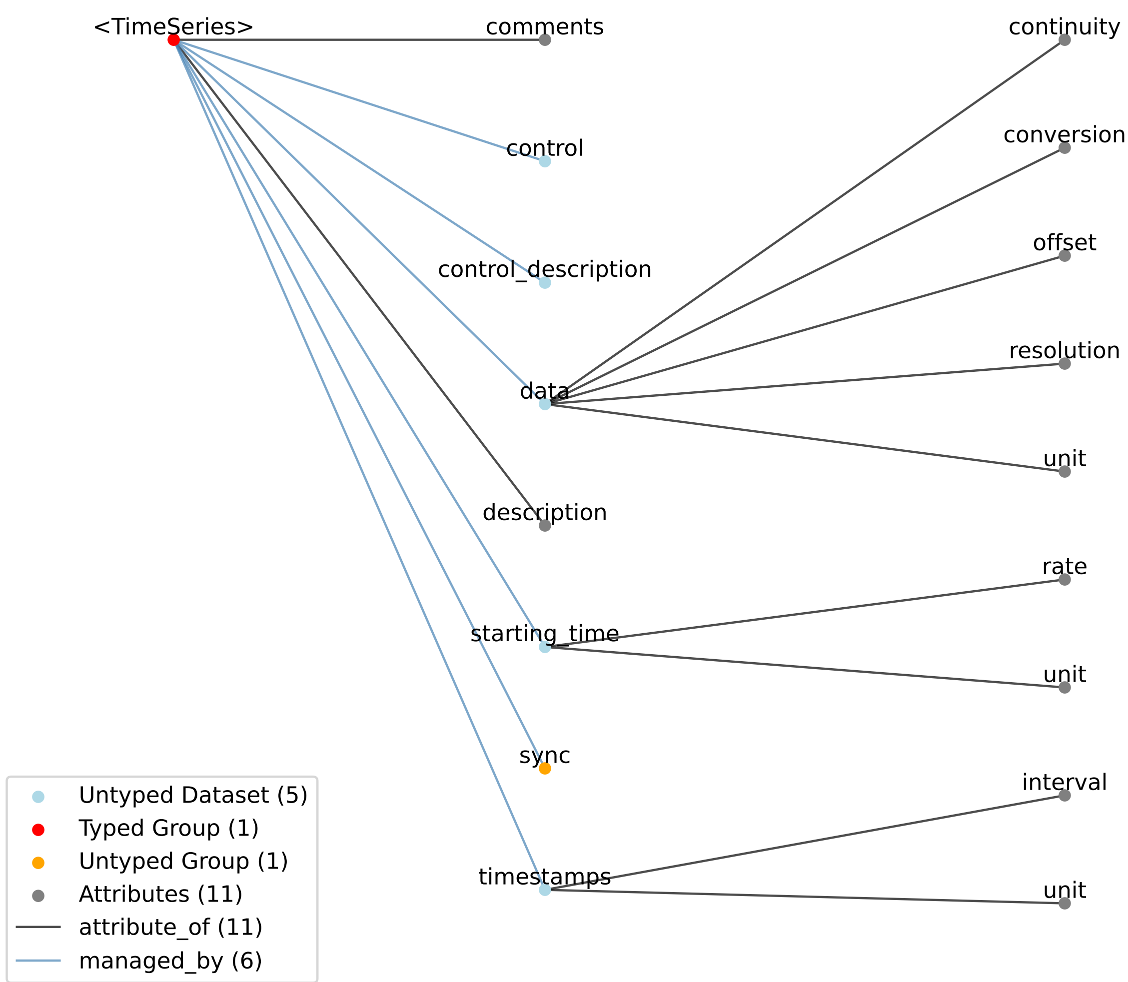


Table 4.3: Datasets, Links, and Attributes contained in <TimeSeries>

Id	Type	Description
<TimeSeries>	Group	Top level Group for <TimeSeries> • **Neurodata Type:** *TimeSeries* • **Extends:** *NWBDataInterface*
—description	Attribute	Description of the time series. • **Data Type:** text • **Required:** False • **Default Value:** no description • **Name:** description
—comments	Attribute	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. • **Data Type:** text • **Required:** False • **Default Value:** no comments • **Name:** comments
—data	Dataset	Data values. Data can be in 1-D, 2-D, 3-D, or 4-D. The first dimension should always represent time. This can also be used to store binary data (e.g., image frames). This can also be a link to data stored in an external file. • **Dimensions:** [['num_times'], ['num_times', 'num_DIM2'], ['num_times', 'num_DIM2', 'num_DIM3'], ['num_times', 'num_DIM2', 'num_DIM3', 'num_DIM4']] • **Shape:** [[None], [None, None], [None, None, None], [None, None, None, None]] • **Name:** data
——conversion	Attribute	Scalar to multiply each element in data to convert it to the specified 'unit'. If the data are stored in acquisition system units or other units that require a conversion to be interpretable, multiply the data by 'conversion' to convert the data to the specified 'unit'. e.g. if the data acquisition system stores values in this object as signed 16-bit integers (int16 range -32,768 to 32,767) that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition values to recorded volts is 2.5/32768/8000 = 9.5367e-9. • **Data Type:** float32 • **Required:** False • **Default Value:** 1.0 • **Name:** conversion

continues on next page

Table 4.3 – continued from previous page

Id	Type	Description
——offset	Attribute	Scalar to add to the data after scaling by 'conversion' to finalize its coercion to the specified 'unit'. Two common examples of this include (a) data stored in an unsigned type that requires a shift after scaling to re-center the data, and (b) specialized recording devices that naturally cause a scalar offset with respect to the true units. • **Data Type:** float32 • **Required:** False • **Default Value:** 0.0 • **Name:** offset
——resolution	Attribute	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 -1.0. • **Data Type:** float32 • **Required:** False • **Default Value:** -1.0 • **Name:** resolution
——unit	Attribute	Base unit of measurement for working with the data. Actual stored values are not necessarily stored in these units. To access the data in these units, multiply 'data' by 'conversion' and add 'offset'. • **Data Type:** text • **Name:** unit
——continuity	Attribute	Optionally describe the continuity of the data. Can be "continuous", "instantaneous", or "step". For example, a voltage trace would be "continuous", because samples are recorded from a continuous process. An array of lick times would be "instantaneous", because the data represents distinct moments in time. Times of image presentations would be "step" because the picture remains the same until the next timepoint. This field is optional, but is useful in providing information about the underlying data. It may inform the way this data is interpreted, the way it is visualized, and what analysis methods are applicable. • **Data Type:** text • **Required:** False • **Name:** continuity
—starting_time	Dataset	Timestamp of the first sample in seconds. When timestamps are uniformly spaced, the timestamp of the first sample can be specified and all subsequent ones calculated from the sampling rate attribute. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Name:** starting_time
——rate	Attribute	Sampling rate, in Hz. • **Data Type:** float32 • **Name:** rate

continues on next page

Table 4.3 – continued from previous page

Id	Type	Description
——unit	Attribute	Unit of measurement for time, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—timestamps	Dataset	Timestamps for samples stored in data, in seconds, relative to the common experiment master-clock stored in NWBFile.timestamps_reference_time. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** timestamps
——interval	Attribute	Value is '1' • **Data Type:** int32 • **Value:** 1 • **Name:** interval
——unit	Attribute	Unit of measurement for timestamps, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—control	Dataset	Numerical labels that apply to each time point in data for the purpose of querying and slicing data by these values. If present, the length of this array should be the same size as the first dimension of data. • **Quantity:** 0 or 1 • **Data Type:** uint8 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** control
—control_description	Dataset	Description of each control value. Must be present if control is present. If present, control_description[0] should describe time points where control == 0. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** ['num_control_values'] • **Shape:** [None] • **Name:** control_description

Table 4.4: Groups contained in <TimeSeries>

Id	Type	Description
<TimeSeries>	Group	Top level Group for <TimeSeries> • **Neurodata Type:** *TimeSeries* • **Extends:** *NWBDataInterface*
—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. • **Quantity:** 0 or 1 • **Name:** sync

4.1.7.1 Groups: sync

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.

- **Quantity:** 0 or 1
- **Name:** sync

4.1.8 ProcessingModule

Overview: A collection of processed data.

`ProcessingModule` extends `NWBContainer` and includes all elements of *NWBContainer* with the following additions or changes.

- **Extends:** *NWBContainer*
- **Primitive Type:** Group
- **Inherits from:** *NWBContainer*, Container
- **Source filename:** nwb.base.yaml
- **Source Specification:** see Section 5.2.8

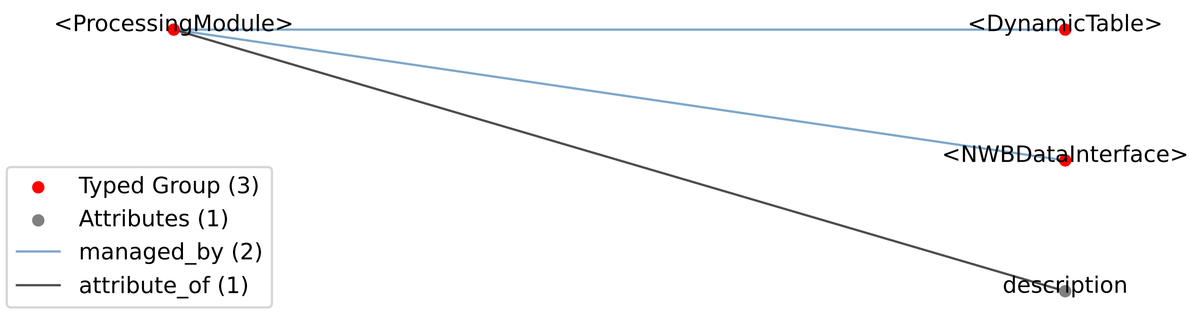


Table 4.5: Datasets, Links, and Attributes contained in <ProcessingModule>

Id	Type	Description
<ProcessingModule>	Group	Top level Group for <ProcessingModule> • **Neurodata Type:** *ProcessingModule* • **Extends:** *NWBContainer*
—description	Attribute	Description of this collection of processed data. • **Data Type:** text • **Name:** description

Table 4.6: Groups contained in <ProcessingModule>

Id	Type	Description
<ProcessingModule>	Group	Top level Group for <ProcessingModule> • **Neurodata Type:** *ProcessingModule* • **Extends:** *NWBContainer*
—<*NWBDataInterface*>	Group	Data objects stored in this collection. • **Extends:** *NWBDataInterface* • **Quantity:** 0 or more

continues on next page

Table 4.6 – continued from previous page

Id	Type	Description
—<DynamicTable>	Group	Tables stored in this collection. • **Extends:** DynamicTable • **Quantity:** 0 or more

4.1.8.1 Groups: <NWBDataInterface>

Data objects stored in this collection.

- **Extends:** *NWBDataInterface*
- **Quantity:** 0 or more

4.1.8.2 Groups: <DynamicTable>

Tables stored in this collection.

- **Extends:** DynamicTable
- **Quantity:** 0 or more

4.1.9 Images

Overview: A collection of images with an optional way to specify the order of the images using the "order_of_images" dataset. An order must be specified if the images are referenced by index, e.g., from an IndexSeries.

`Images` extends `NWBDataInterface` and includes all elements of *NWBDataInterface* with the following additions or changes.

- **Extends:** *NWBDataInterface*
- **Primitive Type:** Group
- **Default Name:** Images
- **Inherits from:** *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.base.yaml
- **Source Specification:** see Section 5.2.9

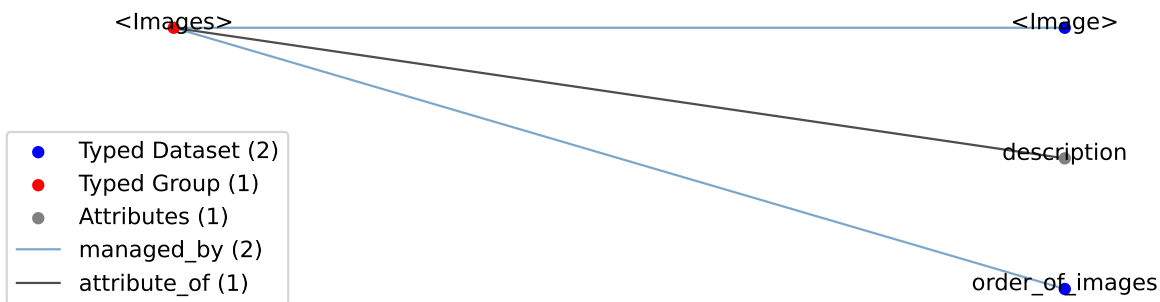


Table 4.7: Datasets, Links, and Attributes contained in <Images>

Id	Type	Description
<Images>	Group	Top level Group for <Images> • **Neurodata Type:** *Images* • **Extends:** *NWBDataInterface* • **Default Name:** Images
—description	Attribute	Description of this collection of images. • **Data Type:** text • **Name:** description
—<*Image*>	Dataset	Images stored in this collection. • **Extends:** *Image* • **Quantity:** 1 or more
—order_of_images	Dataset	Ordered dataset of references to Image objects stored in the parent group. Each Image object in the Images group should be stored once and only once, so the dataset should have the same length as the number of images. • **Extends:** *ImageReferences* • **Quantity:** 0 or 1 • **Name:** order_of_images

4.2 Devices

This source module contains neurodata_types for device data.

4.2.1 Device

Overview: Metadata about a data acquisition device, e.g., recording system, electrode, microscope.

`Device` extends `NWBContainer` and includes all elements of *NWBContainer* with the following additions or changes.

- **Extends:** *NWBContainer*
- **Primitive Type:** Group
- **Inherits from:** *NWBContainer*, Container
- **Source filename:** nwb.device.yaml
- **Source Specification:** see Section 5.3.1

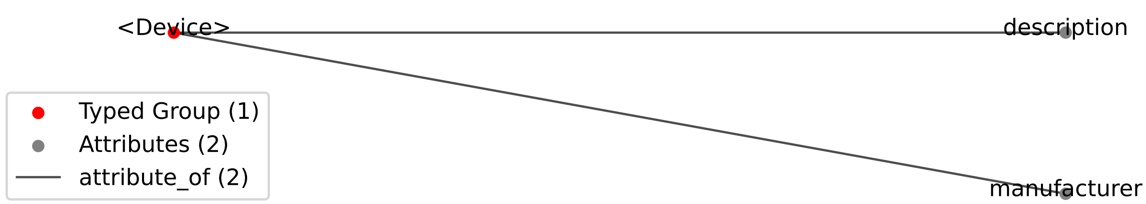


Table 4.8: Datasets, Links, and Attributes contained in <Device>

Id	Type	Description
<Device>	Group	Top level Group for <Device> • **Neurodata Type:** *Device* • **Extends:** *NWBContainer*
—description	Attribute	Description of the device (e.g., model, firmware version, processing software version, etc.) as free-form text. • **Data Type:** text • **Required:** False • **Name:** description
—manufacturer	Attribute	The name of the manufacturer of the device. • **Data Type:** text • **Required:** False • **Name:** manufacturer

4.3 Epochs

This source module contains neurodata_types for epoch data.

4.3.1 TimeIntervals

Overview: A container for aggregating epoch data and the TimeSeries that each epoch applies to.

`TimeIntervals` extends `DynamicTable` and includes all elements of DynamicTable with the following additions or changes.

- **Extends:** DynamicTable
- **Primitive Type:** Group
- **Inherits from:** DynamicTable, Container
- **Source filename:** nwb.epoch.yaml
- **Source Specification:** see Section 5.4.1

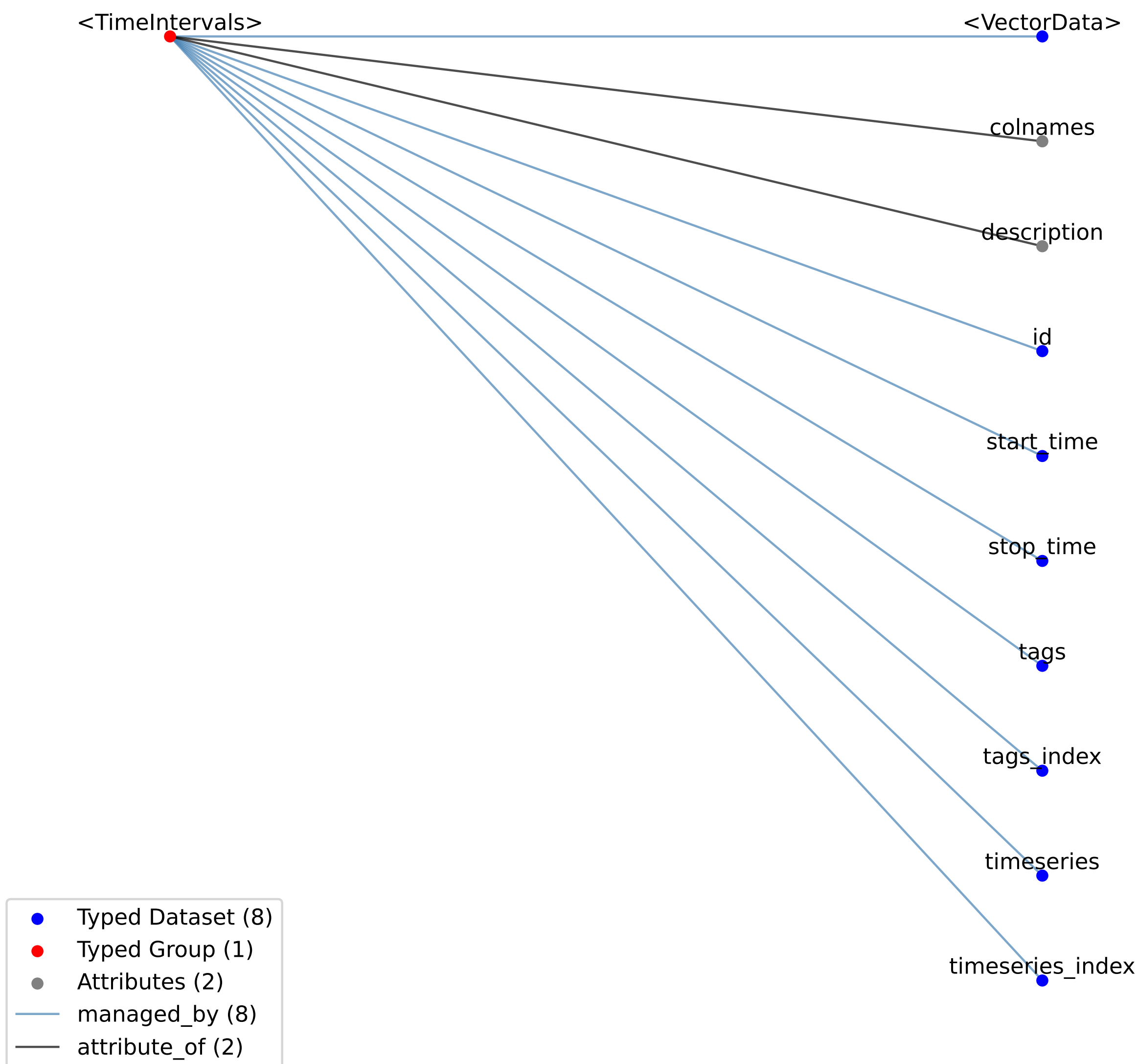
Table 4.9: Datasets, Links, and Attributes contained in <TimeIntervals>

Id	Type	Description
<TimeIntervals>	Group	Top level Group for <TimeIntervals> • **Neurodata Type:** *TimeIntervals* • **Extends:** DynamicTable
—colnames	Attribute	The names of the columns in this table. This should be used to specify an order to the columns. • **Data Type:** text • **Dimensions:** [‘num_columns’] • **Shape:** [None] • **Name:** colnames
—description	Attribute	Description of what is in this dynamic table. • **Data Type:** text • **Name:** description
—start_time	Dataset	Start time of epoch, in seconds. • **Extends:** VectorData • **Data Type:** float32 • **Name:** start_time
—stop_time	Dataset	Stop time of epoch, in seconds. • **Extends:** VectorData • **Data Type:** float32 • **Name:** stop_time

continues on next page

Table 4.9 – continued from previous page

Id	Type	Description
—tags	Dataset	User-defined tags that identify or categorize events. • **Extends:** VectorData • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** tags
—tags_index	Dataset	Index for tags. • **Extends:** VectorIndex • **Quantity:** 0 or 1 • **Name:** tags_index
—timeseries	Dataset	An index into a TimeSeries object. • **Extends:** *TimeSeriesReferenceVectorData* • **Quantity:** 0 or 1 • **Name:** timeseries
—timeseries_index	Dataset	Index for timeseries. • **Extends:** VectorIndex • **Quantity:** 0 or 1 • **Name:** timeseries_index
—id	Dataset	Array of unique identifiers for the rows of this dynamic table. • **Extends:** ElementIdentifiers • **Data Type:** int • **Dimensions:** ['num_rows'] • **Shape:** [None] • **Name:** id
—<VectorData>	Dataset	Vector columns, including index columns, of this dynamic table. • **Extends:** VectorData • **Quantity:** 0 or more


<TimeIntervals>
<VectorData>
colnames
description
id
start_time
stop_time
tags
tags_index
timeseries
timeseries_index
Typed Dataset (8)
Typed Group (1)
Attributes (2)
managed_by (8)
attribute_of (2)

4.4 Image data

This source module contains neurodata_types for image data.

4.4.1 GrayscaleImage

Overview: A grayscale image.

`GrayscaleImage` extends `Image` and includes all elements of *Image* with the following additions or changes.

- **Extends:** *Image*
- **Primitive Type:** Dataset
- **Data Type:** numeric
- **Dimensions:** ['x', 'y']
- **Shape:** [None, None]
- **Inherits from:** *Image*, *NWBData*, Data
- **Source filename:** nwb.image.yaml
- **Source Specification:** see Section 5.5.1

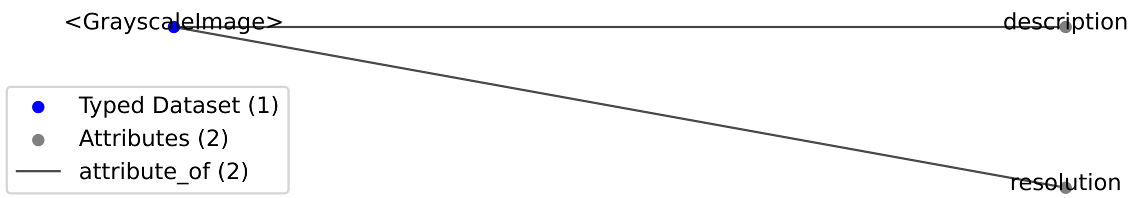


Table 4.10: Datasets, Links, and Attributes contained in <GrayscaleImage>

Id	Type	Description
<GrayscaleImage>	Dataset	Top level Dataset for <GrayscaleImage> • **Neurodata Type:** GrayscaleImage • **Extends:** *Image* • **Data Type:** numeric • **Dimensions:** ['x', 'y'] • **Shape:** [None, None]
—resolution	Attribute	Pixel resolution of the image, in pixels per centimeter. • **Data Type:** float32 • **Required:** False • **Name:** resolution
—description	Attribute	Description of the image. • **Data Type:** text • **Required:** False • **Name:** description

4.4.2 RGBImage

Overview: A color image.

`RGBImage` extends `Image` and includes all elements of *Image* with the following additions or changes.

- **Extends:** *Image*
- **Primitive Type:** Dataset
- **Data Type:** numeric
- **Dimensions:** ['x', 'y', 'r, g, b']
- **Shape:** [None, None, 3]
- **Inherits from:** *Image*, *NWBData*, Data
- **Source filename:** nwb.image.yaml
- **Source Specification:** see Section 5.5.2

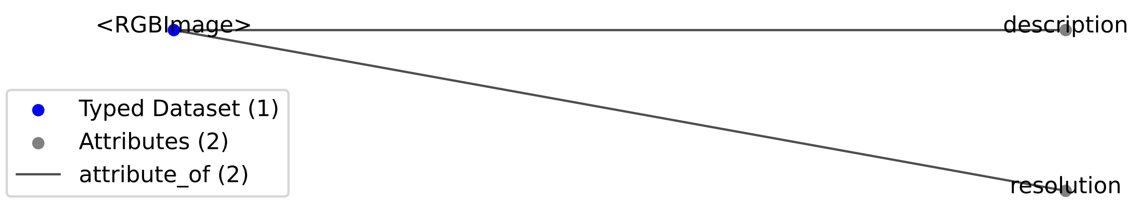


Table 4.11: Datasets, Links, and Attributes contained in <RGBImage>

Id	Type	Description
<RGBImage>	Dataset	Top level Dataset for <RGBImage> • **Neurodata Type:** RGBImage • **Extends:** *Image* • **Data Type:** numeric • **Dimensions:** ['x', 'y', 'r, g, b'] • **Shape:** [None, None, 3]
—resolution	Attribute	Pixel resolution of the image, in pixels per centimeter. • **Data Type:** float32 • **Required:** False • **Name:** resolution
—description	Attribute	Description of the image. • **Data Type:** text • **Required:** False • **Name:** description

4.4.3 RGBAImage

Overview: A color image with transparency.

`RGBAImage` extends `Image` and includes all elements of *Image* with the following additions or changes.

- **Extends:** *Image*
- **Primitive Type:** Dataset
- **Data Type:** numeric
- **Dimensions:** ['x', 'y', 'r, g, b, a']
- **Shape:** [None, None, 4]
- **Inherits from:** *Image*, *NWBData*, Data
- **Source filename:** nwb.image.yaml
- **Source Specification:** see Section 5.5.3

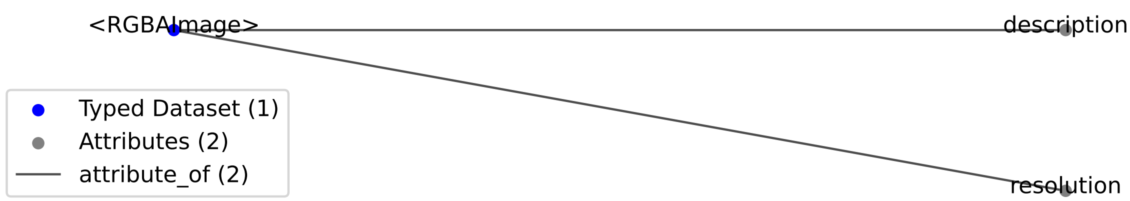


Table 4.12: Datasets, Links, and Attributes contained in <RGBAImage>

Id	Type	Description
<RGBAImage>	Dataset	Top level Dataset for <RGBAImage> • **Neurodata Type:** RGBAImage • **Extends:** *Image* • **Data Type:** numeric • **Dimensions:** ['x', 'y', 'r, g, b, a'] • **Shape:** [None, None, 4]
—resolution	Attribute	Pixel resolution of the image, in pixels per centimeter. • **Data Type:** float32 • **Required:** False • **Name:** resolution
—description	Attribute	Description of the image. • **Data Type:** text • **Required:** False • **Name:** description

4.4.4 ImageSeries

Overview: General image data that is common between acquisition and stimulus time series. Sometimes the image data is stored in the file in a raw format while other times it will be stored as a series of external image files in the host file system. The data field will either be binary data, if the data is stored in the NWB file, or empty, if the data is stored in an external image stack. [frame][x][y] or [frame][x][y][z].

`ImageSeries` extends `TimeSeries` and includes all elements of *TimeSeries* with the following additions or changes.

- **Extends:** *TimeSeries*
- **Primitive Type:** Group
- **Inherits from:** *TimeSeries*, *NWBDataInterface*, *NWBContainer*, Container
- **Subtypes:** *TwoPhotonSeries*, *OnePhotonSeries*, *ImageMaskSeries*, *OpticalSeries*
- **Source filename:** nwb.image.yaml
- **Source Specification:** see Section 5.5.4

Table 4.13: Datasets, Links, and Attributes contained in <ImageSeries>

Id	Type	Description
<ImageSeries>	Group	Top level Group for <ImageSeries> • **Neurodata Type:** *ImageSeries* • **Extends:** *TimeSeries*
—description	Attribute	Description of the time series. • **Data Type:** text • **Required:** False • **Default Value:** no description • **Name:** description
—comments	Attribute	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. • **Data Type:** text • **Required:** False • **Default Value:** no comments • **Name:** comments
—data	Dataset	Binary data representing images across frames. If data are stored in an external file, this should be an empty 3D array. • **Data Type:** numeric • **Dimensions:** [['frame', 'x', 'y'], ['frame', 'x', 'y', 'z']] • **Shape:** [[None, None, None], [None, None, None, None]] • **Name:** data

continues on next page

Table 4.13 – continued from previous page

Id	Type	Description
——conversion	Attribute	Scalar to multiply each element in data to convert it to the specified 'unit'. If the data are stored in acquisition system units or other units that require a conversion to be interpretable, multiply the data by 'conversion' to convert the data to the specified 'unit'. e.g. if the data acquisition system stores values in this object as signed 16-bit integers (int16 range -32,768 to 32,767) that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition values to recorded volts is 2.5/32768/8000 = 9.5367e-9. • **Data Type:** float32 • **Required:** False • **Default Value:** 1.0 • **Name:** conversion
——offset	Attribute	Scalar to add to the data after scaling by 'conversion' to finalize its coercion to the specified 'unit'. Two common examples of this include (a) data stored in an unsigned type that requires a shift after scaling to re-center the data, and (b) specialized recording devices that naturally cause a scalar offset with respect to the true units. • **Data Type:** float32 • **Required:** False • **Default Value:** 0.0 • **Name:** offset
——resolution	Attribute	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 -1.0. • **Data Type:** float32 • **Required:** False • **Default Value:** -1.0 • **Name:** resolution
——unit	Attribute	Base unit of measurement for working with the data. Actual stored values are not necessarily stored in these units. To access the data in these units, multiply 'data' by 'conversion' and add 'offset'. • **Data Type:** text • **Name:** unit

continues on next page

Table 4.13 – continued from previous page

Id	Type	Description
——continuity	Attribute	Optionally describe the continuity of the data. Can be "continuous", "instantaneous", or "step". For example, a voltage trace would be "continuous", because samples are recorded from a continuous process. An array of lick times would be "instantaneous", because the data represents distinct moments in time. Times of image presentations would be "step" because the picture remains the same until the next timepoint. This field is optional, but is useful in providing information about the underlying data. It may inform the way this data is interpreted, the way it is visualized, and what analysis methods are applicable. • **Data Type:** text • **Required:** False • **Name:** continuity
—dimension	Dataset	Number of pixels on x, y, (and z) axes. • **Quantity:** 0 or 1 • **Data Type:** int32 • **Dimensions:** ['rank'] • **Shape:** [None] • **Name:** dimension
—external_file	Dataset	Paths to one or more external file(s). The field is only present if format='external'. This is only relevant if the image series 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 NWB file and that file is linked to this file. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** ['num_files'] • **Shape:** [None] • **Name:** external_file
——starting_frame	Attribute	Each external image may contain one or more consecutive frames of the full ImageSeries. This attribute serves as an index to indicate which frames each file contains, to faciliate random access. The 'starting_frame' attribute, hence, contains a list of frame numbers within the full ImageSeries of the first frame of each file listed in the parent 'external_file' dataset. Zero-based indexing is used (hence, the first element will always be zero). For example, if the 'external_file' dataset has three paths to files and the first file has 5 frames, the second file has 10 frames, and the third file has 20 frames, then this attribute will have values [0, 5, 15]. If there is a single external file that holds all of the frames of the ImageSeries (and so there is a single element in the 'external_file' dataset), then this attribute should have value [0]. • **Data Type:** int32 • **Dimensions:** ['num_files'] • **Shape:** [None] • **Name:** starting_frame

continues on next page

Table 4.13 – continued from previous page

Id	Type	Description
—format	Dataset	Format of image. If this is 'external', then the attribute 'external_file' contains the path information to the image files. If this is 'raw', then the raw (single-channel) binary data is stored in the 'data' dataset. If this attribute is not present, then the default format='raw' case is assumed. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** format
—starting_time	Dataset	Timestamp of the first sample in seconds. When timestamps are uniformly spaced, the timestamp of the first sample can be specified and all subsequent ones calculated from the sampling rate attribute. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Name:** starting_time
——rate	Attribute	Sampling rate, in Hz. • **Data Type:** float32 • **Name:** rate
——unit	Attribute	Unit of measurement for time, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—timestamps	Dataset	Timestamps for samples stored in data, in seconds, relative to the common experiment master-clock stored in NWBFile.timestamps_reference_time. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** timestamps
——interval	Attribute	Value is '1' • **Data Type:** int32 • **Value:** 1 • **Name:** interval
——unit	Attribute	Unit of measurement for timestamps, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit

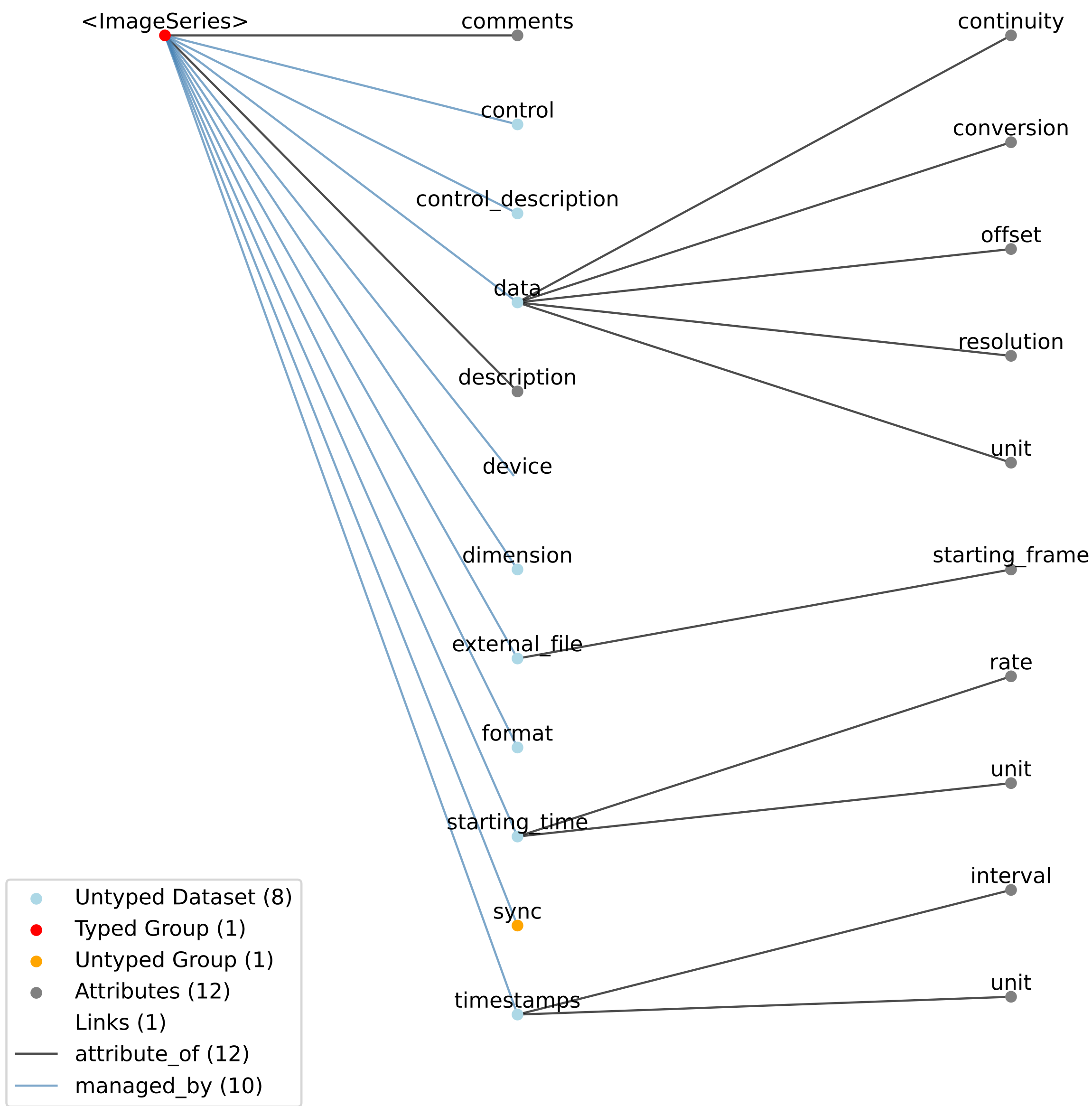
continues on next page

Table 4.13 – continued from previous page

Id	Type	Description
—control	Dataset	Numerical labels that apply to each time point in data for the purpose of querying and slicing data by these values. If present, the length of this array should be the same size as the first dimension of data. • **Quantity:** 0 or 1 • **Data Type:** uint8 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** control
—control_description	Dataset	Description of each control value. Must be present if control is present. If present, control_description[0] should describe time points where control == 0. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** ['num_control_values'] • **Shape:** [None] • **Name:** control_description
—device	Link	Link to the Device object that was used to capture these images. • **Target Type** *Device* • **Name:** device

Table 4.14: Groups contained in <ImageSeries>

Id	Type	Description
<ImageSeries>	Group	Top level Group for <ImageSeries> • **Neurodata Type:** *ImageSeries* • **Extends:** *TimeSeries*
—device	Link	Link to the Device object that was used to capture these images. • **Target Type** *Device* • **Name:** device
—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. • **Quantity:** 0 or 1 • **Name:** sync


<ImageSeries>
comments
control
control_description
data
description
device
dimension
external_file
format
starting_time
sync
timestamps
continuity
conversion
offset
resolution
unit
starting_frame
rate
unit
interval
unit
Untyped Dataset (8)
Typed Group (1)
Untyped Group (1)
Attributes (12)
Links (1)
attribute_of (12)
managed_by (10)

4.4.4.1 Groups: sync

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.

- **Quantity:** 0 or 1
- **Name:** sync

4.4.5 ImageMaskSeries

Overview: 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` extends `ImageSeries` and includes all elements of *ImageSeries* with the following additions or changes.

- **Extends:** *ImageSeries*
- **Primitive Type:** Group
- **Inherits from:** *ImageSeries*, *TimeSeries*, *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.image.yaml
- **Source Specification:** see Section 5.5.5

Table 4.15: Datasets, Links, and Attributes contained in <ImageMaskSeries>

Id	Type	Description
<ImageMaskSeries>	Group	Top level Group for <ImageMaskSeries> • **Neurodata Type:** *ImageMaskSeries* • **Extends:** *ImageSeries*
—description	Attribute	Description of the time series. • **Data Type:** text • **Required:** False • **Default Value:** no description • **Name:** description
—comments	Attribute	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. • **Data Type:** text • **Required:** False • **Default Value:** no comments • **Name:** comments
—data	Dataset	Binary data representing images across frames. If data are stored in an external file, this should be an empty 3D array. • **Data Type:** numeric • **Dimensions:** [['frame', 'x', 'y'], ['frame', 'x', 'y', 'z']] • **Shape:** [[None, None, None], [None, None, None, None]] • **Name:** data

continues on next page

Table 4.15 – continued from previous page

Id	Type	Description
——conversion	Attribute	Scalar to multiply each element in data to convert it to the specified 'unit'. If the data are stored in acquisition system units or other units that require a conversion to be interpretable, multiply the data by 'conversion' to convert the data to the specified 'unit'. e.g. if the data acquisition system stores values in this object as signed 16-bit integers (int16 range -32,768 to 32,767) that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition values to recorded volts is 2.5/32768/8000 = 9.5367e-9. • **Data Type:** float32 • **Required:** False • **Default Value:** 1.0 • **Name:** conversion
——offset	Attribute	Scalar to add to the data after scaling by 'conversion' to finalize its coercion to the specified 'unit'. Two common examples of this include (a) data stored in an unsigned type that requires a shift after scaling to re-center the data, and (b) specialized recording devices that naturally cause a scalar offset with respect to the true units. • **Data Type:** float32 • **Required:** False • **Default Value:** 0.0 • **Name:** offset
——resolution	Attribute	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 -1.0. • **Data Type:** float32 • **Required:** False • **Default Value:** -1.0 • **Name:** resolution
——unit	Attribute	Base unit of measurement for working with the data. Actual stored values are not necessarily stored in these units. To access the data in these units, multiply 'data' by 'conversion' and add 'offset'. • **Data Type:** text • **Name:** unit

continues on next page

Table 4.15 – continued from previous page

Id	Type	Description
——continuity	Attribute	Optionally describe the continuity of the data. Can be “continuous”, “instantaneous”, or “step”. For example, a voltage trace would be “continuous”, because samples are recorded from a continuous process. An array of lick times would be “instantaneous”, because the data represents distinct moments in time. Times of image presentations would be “step” because the picture remains the same until the next timepoint. This field is optional, but is useful in providing information about the underlying data. It may inform the way this data is interpreted, the way it is visualized, and what analysis methods are applicable. • **Data Type:** text • **Required:** False • **Name:** continuity
—dimension	Dataset	Number of pixels on x, y, (and z) axes. • **Quantity:** 0 or 1 • **Data Type:** int32 • **Dimensions:** [‘rank’] • **Shape:** [None] • **Name:** dimension
—external_file	Dataset	Paths to one or more external file(s). The field is only present if format=’external’. This is only relevant if the image series 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 NWB file and that file is linked to this file. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** [‘num_files’] • **Shape:** [None] • **Name:** external_file
——starting_frame	Attribute	Each external image may contain one or more consecutive frames of the full ImageSeries. This attribute serves as an index to indicate which frames each file contains, to faciliate random access. The ‘starting_frame’ attribute, hence, contains a list of frame numbers within the full ImageSeries of the first frame of each file listed in the parent ‘external_file’ dataset. Zero-based indexing is used (hence, the first element will always be zero). For example, if the ‘external_file’ dataset has three paths to files and the first file has 5 frames, the second file has 10 frames, and the third file has 20 frames, then this attribute will have values [0, 5, 15]. If there is a single external file that holds all of the frames of the ImageSeries (and so there is a single element in the ‘external_file’ dataset), then this attribute should have value [0]. • **Data Type:** int32 • **Dimensions:** [‘num_files’] • **Shape:** [None] • **Name:** starting_frame

continues on next page

Table 4.15 – continued from previous page

Id	Type	Description
—format	Dataset	Format of image. If this is 'external', then the attribute 'external_file' contains the path information to the image files. If this is 'raw', then the raw (single-channel) binary data is stored in the 'data' dataset. If this attribute is not present, then the default format='raw' case is assumed. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** format
—starting_time	Dataset	Timestamp of the first sample in seconds. When timestamps are uniformly spaced, the timestamp of the first sample can be specified and all subsequent ones calculated from the sampling rate attribute. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Name:** starting_time
——rate	Attribute	Sampling rate, in Hz. • **Data Type:** float32 • **Name:** rate
——unit	Attribute	Unit of measurement for time, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—timestamps	Dataset	Timestamps for samples stored in data, in seconds, relative to the common experiment master-clock stored in NWBFile.timestamps_reference_time. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** timestamps
——interval	Attribute	Value is '1' • **Data Type:** int32 • **Value:** 1 • **Name:** interval
——unit	Attribute	Unit of measurement for timestamps, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit

continues on next page

Table 4.15 – continued from previous page

Id	Type	Description
—control	Dataset	Numerical labels that apply to each time point in data for the purpose of querying and slicing data by these values. If present, the length of this array should be the same size as the first dimension of data. • **Quantity:** 0 or 1 • **Data Type:** uint8 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** control
—control_description	Dataset	Description of each control value. Must be present if control is present. If present, control_description[0] should describe time points where control == 0. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** ['num_control_values'] • **Shape:** [None] • **Name:** control_description
—masked_imageseries	Link	Link to ImageSeries object that this image mask is applied to. • **Target Type** *ImageSeries* • **Name:** masked_imageseries
—device	Link	Link to the Device object that was used to capture these images. • **Target Type** *Device* • **Name:** device

Table 4.16: Groups contained in <ImageMaskSeries>

Id	Type	Description
<ImageMaskSeries>	Group	Top level Group for <ImageMaskSeries> • **Neurodata Type:** *ImageMaskSeries* • **Extends:** *ImageSeries*
—masked_imageseries	Link	Link to ImageSeries object that this image mask is applied to. • **Target Type** *ImageSeries* • **Name:** masked_imageseries
—device	Link	Link to the Device object that was used to capture these images. • **Target Type** *Device* • **Name:** device

continues on next page

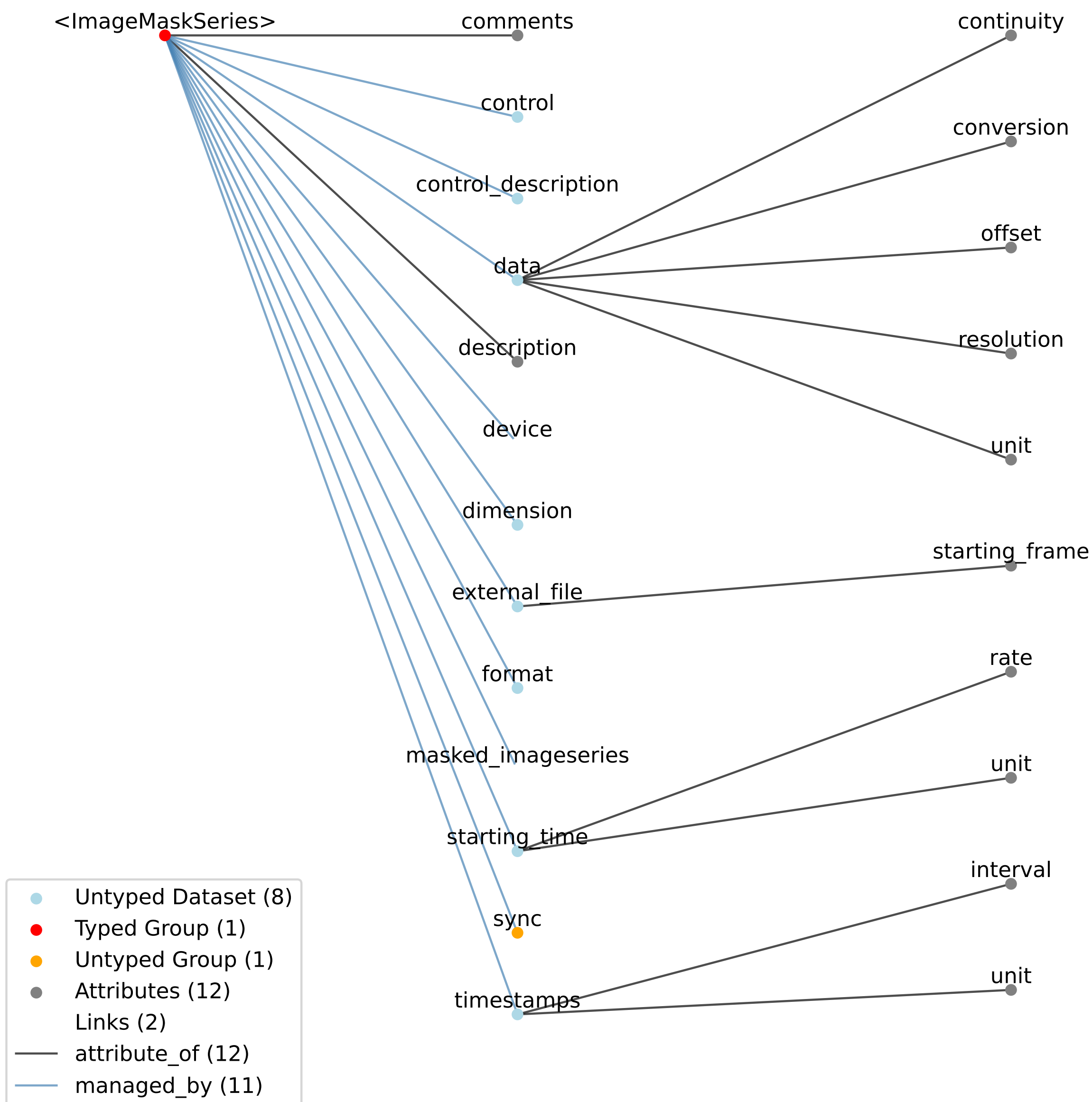
Table 4.16 – continued from previous page

Id	Type	Description
—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. • **Quantity:** 0 or 1 • **Name:** sync

4.4.5.1 Groups: sync

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.

- **Quantity:** 0 or 1
- **Name:** sync


<ImageMaskSeries>
comments
control
control_description
data
description
device
dimension
external_file
format
masked_imageseries
starting_time
sync
timestamps
continuity
conversion
offset
resolution
unit
starting_frame
rate
unit
interval
unit
Untyped Dataset (8)
Typed Group (1)
Untyped Group (1)
Attributes (12)
Links (2)
attribute_of (12)
managed_by (11)

4.4.6 OpticalSeries

Overview: 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 (e.g., 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` extends `ImageSeries` and includes all elements of *ImageSeries* with the following additions or changes.

- **Extends:** *ImageSeries*
- **Primitive Type:** Group
- **Inherits from:** *ImageSeries*, *TimeSeries*, *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.image.yaml
- **Source Specification:** see Section 5.5.6

Table 4.17: Datasets, Links, and Attributes contained in <OpticalSeries>

Id	Type	Description
<OpticalSeries>	Group	Top level Group for <OpticalSeries> • **Neurodata Type:** *OpticalSeries* • **Extends:** *ImageSeries*
—description	Attribute	Description of the time series. • **Data Type:** text • **Required:** False • **Default Value:** no description • **Name:** description
—comments	Attribute	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. • **Data Type:** text • **Required:** False • **Default Value:** no comments • **Name:** comments
—distance	Dataset	Distance from camera/monitor to target/eye. • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** distance
—field_of_view	Dataset	Width, height and depth of image, or imaged area, in meters. • **Quantity:** 0 or 1 • **Data Type:** float32 • **Dimensions:** [['width, height'], ['width, height, depth']] • **Shape:** [[2], [3]] • **Name:** field_of_view

continues on next page

Table 4.17 – continued from previous page

Id	Type	Description
—data	Dataset	Images presented to subject, either grayscale or RGB • **Data Type:** numeric • **Dimensions:** [['frame', 'x', 'y'], ['frame', 'x', 'y', 'r, g, b']] • **Shape:** [[None, None, None], [None, None, None, 3]] • **Name:** data
——conversion	Attribute	Scalar to multiply each element in data to convert it to the specified 'unit'. If the data are stored in acquisition system units or other units that require a conversion to be interpretable, multiply the data by 'conversion' to convert the data to the specified 'unit'. e.g. if the data acquisition system stores values in this object as signed 16-bit integers (int16 range -32,768 to 32,767) that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition values to recorded volts is 2.5/32768/8000 = 9.5367e-9. • **Data Type:** float32 • **Required:** False • **Default Value:** 1.0 • **Name:** conversion
——offset	Attribute	Scalar to add to the data after scaling by 'conversion' to finalize its coercion to the specified 'unit'. Two common examples of this include (a) data stored in an unsigned type that requires a shift after scaling to re-center the data, and (b) specialized recording devices that naturally cause a scalar offset with respect to the true units. • **Data Type:** float32 • **Required:** False • **Default Value:** 0.0 • **Name:** offset
——resolution	Attribute	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 -1.0. • **Data Type:** float32 • **Required:** False • **Default Value:** -1.0 • **Name:** resolution
——unit	Attribute	Base unit of measurement for working with the data. Actual stored values are not necessarily stored in these units. To access the data in these units, multiply 'data' by 'conversion' and add 'offset'. • **Data Type:** text • **Name:** unit

continues on next page

Table 4.17 – continued from previous page

Id	Type	Description
——continuity	Attribute	Optionally describe the continuity of the data. Can be “continuous”, “instantaneous”, or “step”. For example, a voltage trace would be “continuous”, because samples are recorded from a continuous process. An array of lick times would be “instantaneous”, because the data represents distinct moments in time. Times of image presentations would be “step” because the picture remains the same until the next timepoint. This field is optional, but is useful in providing information about the underlying data. It may inform the way this data is interpreted, the way it is visualized, and what analysis methods are applicable. • **Data Type:** text • **Required:** False • **Name:** continuity
—orientation	Dataset	Description of image relative to some reference frame (e.g., which way is up). Must also specify frame of reference. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** orientation
—dimension	Dataset	Number of pixels on x, y, (and z) axes. • **Quantity:** 0 or 1 • **Data Type:** int32 • **Dimensions:** [‘rank’] • **Shape:** [None] • **Name:** dimension
—external_file	Dataset	Paths to one or more external file(s). The field is only present if format=’external’. This is only relevant if the image series 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 NWB file and that file is linked to this file. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** [‘num_files’] • **Shape:** [None] • **Name:** external_file

continues on next page

Table 4.17 – continued from previous page

Id	Type	Description
——starting_frame	Attribute	Each external image may contain one or more consecutive frames of the full ImageSeries. This attribute serves as an index to indicate which frames each file contains, to faciliate random access. The 'starting_frame' attribute, hence, contains a list of frame numbers within the full ImageSeries of the first frame of each file listed in the parent 'external_file' dataset. Zero-based indexing is used (hence, the first element will always be zero). For example, if the 'external_file' dataset has three paths to files and the first file has 5 frames, the second file has 10 frames, and the third file has 20 frames, then this attribute will have values [0, 5, 15]. If there is a single external file that holds all of the frames of the ImageSeries (and so there is a single element in the 'external_file' dataset), then this attribute should have value [0]. • **Data Type:** int32 • **Dimensions:** ['num_files'] • **Shape:** [None] • **Name:** starting_frame
—format	Dataset	Format of image. If this is 'external', then the attribute 'external_file' contains the path information to the image files. If this is 'raw', then the raw (single-channel) binary data is stored in the 'data' dataset. If this attribute is not present, then the default format='raw' case is assumed. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** format
—starting_time	Dataset	Timestamp of the first sample in seconds. When timestamps are uniformly spaced, the timestamp of the first sample can be specified and all subsequent ones calculated from the sampling rate attribute. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Name:** starting_time
——rate	Attribute	Sampling rate, in Hz. • **Data Type:** float32 • **Name:** rate
——unit	Attribute	Unit of measurement for time, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—timestamps	Dataset	Timestamps for samples stored in data, in seconds, relative to the common experiment master-clock stored in NWBFile.timestamps_reference_time. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** timestamps

continues on next page

Table 4.17 – continued from previous page

Id	Type	Description
——interval	Attribute	Value is '1' • **Data Type:** int32 • **Value:** 1 • **Name:** interval
——unit	Attribute	Unit of measurement for timestamps, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—control	Dataset	Numerical labels that apply to each time point in data for the purpose of querying and slicing data by these values. If present, the length of this array should be the same size as the first dimension of data. • **Quantity:** 0 or 1 • **Data Type:** uint8 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** control
—control_description	Dataset	Description of each control value. Must be present if control is present. If present, control_description[0] should describe time points where control == 0. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** ['num_control_values'] • **Shape:** [None] • **Name:** control_description
—device	Link	Link to the Device object that was used to capture these images. • **Target Type** *Device* • **Name:** device

Table 4.18: Groups contained in <OpticalSeries>

Id	Type	Description
<OpticalSeries>	Group	Top level Group for <OpticalSeries> • **Neurodata Type:** *OpticalSeries* • **Extends:** *ImageSeries*
—device	Link	Link to the Device object that was used to capture these images. • **Target Type** *Device* • **Name:** device

continues on next page

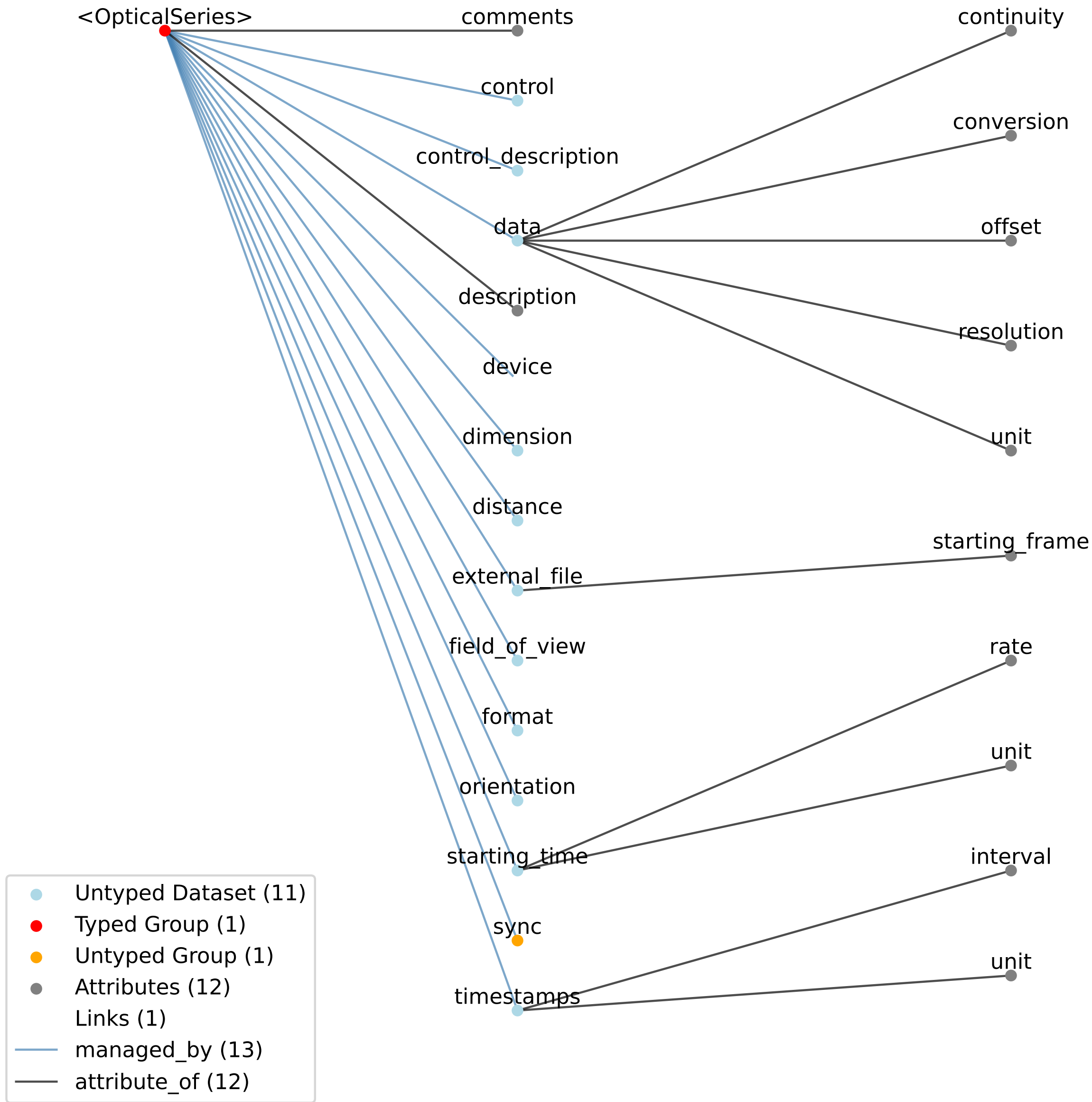
Table 4.18 – continued from previous page

Id	Type	Description
—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. • **Quantity:** 0 or 1 • **Name:** sync

4.4.6.1 Groups: sync

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.

- **Quantity:** 0 or 1
- **Name:** sync


<OpticalSeries>
comments
control
control_description
data
description
device
dimension
distance
external_file
field_of_view
format
orientation
starting_time
sync
timestamps
continuity
conversion
offset
resolution
unit
starting_frame
rate
unit
interval
unit
Untyped Dataset (11)
Typed Group (1)
Untyped Group (1)
Attributes (12)
Links (1)
managed_by (13)
attribute_of (12)

4.4.7 IndexSeries

Overview: Stores indices to image frames stored in an ImageSeries. The purpose of the IndexSeries is to allow a static image stack to be stored in an Images object, 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 Images object, and the timestamps array indicates when that image was displayed.

`IndexSeries` extends `TimeSeries` and includes all elements of *TimeSeries* with the following additions or changes.

- **Extends:** *TimeSeries*
- **Primitive Type:** Group
- **Inherits from:** *TimeSeries*, *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.image.yaml
- **Source Specification:** see Section 5.5.7

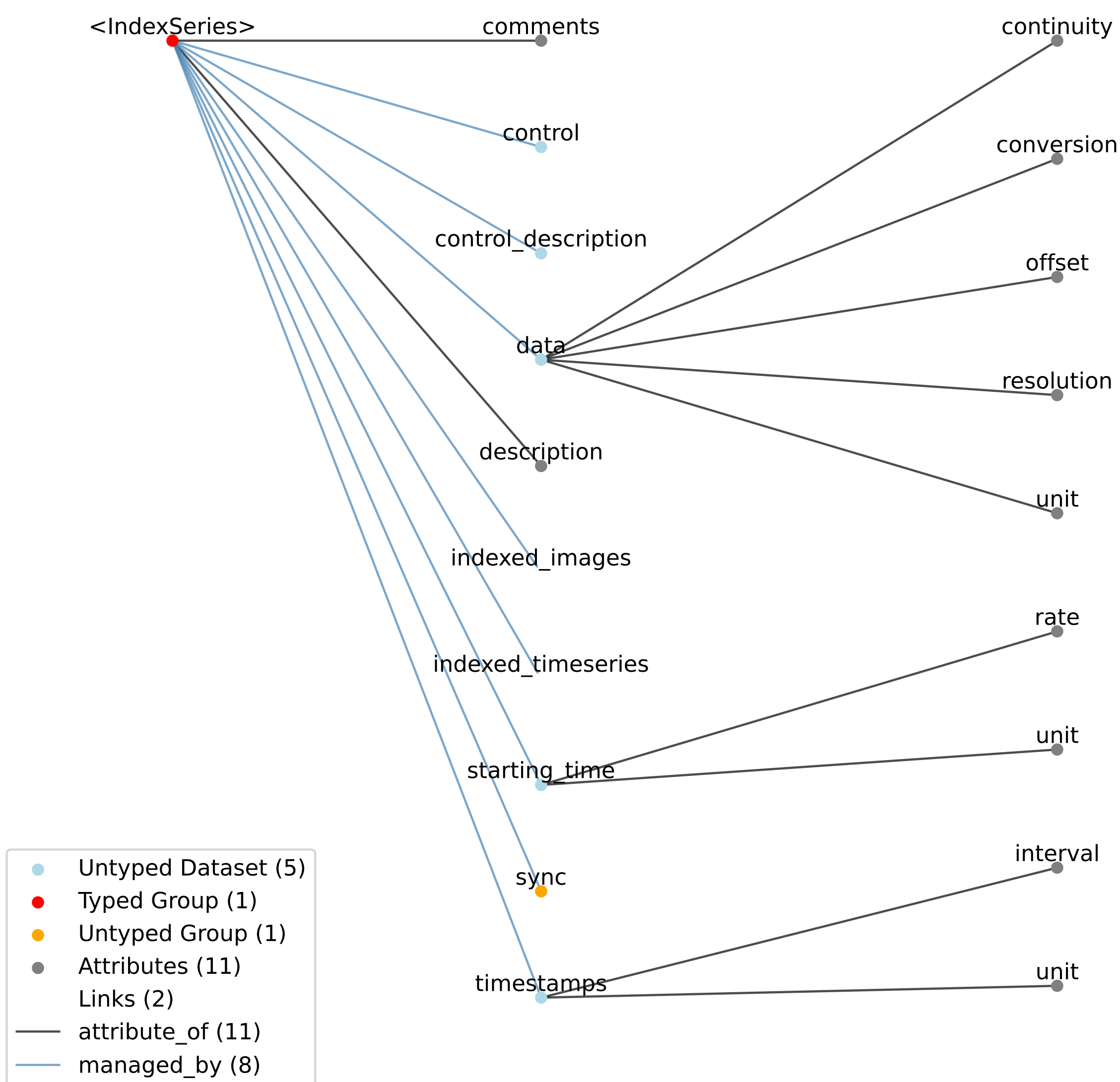


Table 4.19: Datasets, Links, and Attributes contained in <IndexSeries>

Id	Type	Description
<IndexSeries>	Group	Top level Group for <IndexSeries> • **Neurodata Type:** *IndexSeries* • **Extends:** *TimeSeries*
—description	Attribute	Description of the time series. • **Data Type:** text • **Required:** False • **Default Value:** no description • **Name:** description
—comments	Attribute	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. • **Data Type:** text • **Required:** False • **Default Value:** no comments • **Name:** comments
—data	Dataset	Index of the image (using zero-indexing) in the linked Images object. • **Data Type:** uint32 • **Dimensions:** [‘num_times’] • **Shape:** [None] • **Name:** data
——conversion	Attribute	This field is unused by IndexSeries. • **Data Type:** float32 • **Required:** False • **Name:** conversion
——resolution	Attribute	This field is unused by IndexSeries. • **Data Type:** float32 • **Required:** False • **Name:** resolution
——offset	Attribute	This field is unused by IndexSeries. • **Data Type:** float32 • **Required:** False • **Name:** offset
——unit	Attribute	This field is unused by IndexSeries and has the value N/A. • **Data Type:** text • **Value:** N/A • **Name:** unit

continues on next page

Table 4.19 – continued from previous page

Id	Type	Description
——continuity	Attribute	Optionally describe the continuity of the data. Can be “continuous”, “instantaneous”, or “step”. For example, a voltage trace would be “continuous”, because samples are recorded from a continuous process. An array of lick times would be “instantaneous”, because the data represents distinct moments in time. Times of image presentations would be “step” because the picture remains the same until the next timepoint. This field is optional, but is useful in providing information about the underlying data. It may inform the way this data is interpreted, the way it is visualized, and what analysis methods are applicable. • **Data Type:** text • **Required:** False • **Name:** continuity
—starting_time	Dataset	Timestamp of the first sample in seconds. When timestamps are uniformly spaced, the timestamp of the first sample can be specified and all subsequent ones calculated from the sampling rate attribute. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Name:** starting_time
——rate	Attribute	Sampling rate, in Hz. • **Data Type:** float32 • **Name:** rate
——unit	Attribute	Unit of measurement for time, which is fixed to ‘seconds’. • **Data Type:** text • **Value:** seconds • **Name:** unit
—timestamps	Dataset	Timestamps for samples stored in data, in seconds, relative to the common experiment master-clock stored in NWBFile.timestamps_reference_time. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Dimensions:** [‘num_times’] • **Shape:** [None] • **Name:** timestamps
——interval	Attribute	Value is ‘1’ • **Data Type:** int32 • **Value:** 1 • **Name:** interval
——unit	Attribute	Unit of measurement for timestamps, which is fixed to ‘seconds’. • **Data Type:** text • **Value:** seconds • **Name:** unit

continues on next page

Table 4.19 – continued from previous page

Id	Type	Description
—control	Dataset	Numerical labels that apply to each time point in data for the purpose of querying and slicing data by these values. If present, the length of this array should be the same size as the first dimension of data. • **Quantity:** 0 or 1 • **Data Type:** uint8 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** control
—control_description	Dataset	Description of each control value. Must be present if control is present. If present, control_description[0] should describe time points where control == 0. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** ['num_control_values'] • **Shape:** [None] • **Name:** control_description
—indexed_timeseries	Link	Link to ImageSeries object containing images that are indexed. Use of this link is discouraged and will be deprecated. Link to an Images type instead. • **Target Type** *ImageSeries* • **Name:** indexed_timeseries
—indexed_images	Link	Link to Images object containing an ordered set of images that are indexed. The Images object must contain a 'ordered_images' dataset specifying the order of the images in the Images type. • **Target Type** *Images* • **Name:** indexed_images

Table 4.20: Groups contained in <IndexSeries>

Id	Type	Description
<IndexSeries>	Group	Top level Group for <IndexSeries> • **Neurodata Type:** *IndexSeries* • **Extends:** *TimeSeries*
—indexed_timeseries	Link	Link to ImageSeries object containing images that are indexed. Use of this link is discouraged and will be deprecated. Link to an Images type instead. • **Target Type** *ImageSeries* • **Name:** indexed_timeseries

continues on next page

Table 4.20 – continued from previous page

Id	Type	Description
—indexed_images	Link	Link to Images object containing an ordered set of images that are indexed. The Images object must contain a 'ordered_images' dataset specifying the order of the images in the Images type. • **Target Type** *Images* • **Name:** indexed_images
—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. • **Quantity:** 0 or 1 • **Name:** sync

4.4.7.1 Groups: sync

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.

- **Quantity:** 0 or 1
- **Name:** sync

4.5 NWB file

Main NWB file specification.

4.5.1 ScratchData

Overview: Any one-off datasets

`ScratchData` extends `NWBData` and includes all elements of *NWBData* with the following additions or changes.

- **Extends:** *NWBData*
- **Primitive Type:** Dataset
- **Inherits from:** *NWBData*, Data
- **Source filename:** nwb.file.yaml
- **Source Specification:** see Section 5.6.1

Table 4.21: Datasets, Links, and Attributes contained in <ScratchData>

Id	Type	Description
<ScratchData>	Dataset	Top level Dataset for <ScratchData> • **Neurodata Type:** ScratchData • **Extends:** *NWBData*
—notes	Attribute	Any notes the user has about the dataset being stored • **Data Type:** text • **Name:** notes

4.5.2 NWBFile

Overview: An NWB file storing cellular-based neurophysiology data from a single experimental session.

`NWBFile` extends `NWBContainer` and includes all elements of *NWBContainer* with the following additions or changes.

- **Extends:** *NWBContainer*
- **Primitive Type:** Group
- **Name:** root
- **Inherits from:** *NWBContainer*, Container
- **Source filename:** nwb.file.yaml
- **Source Specification:** see Section 5.6.2

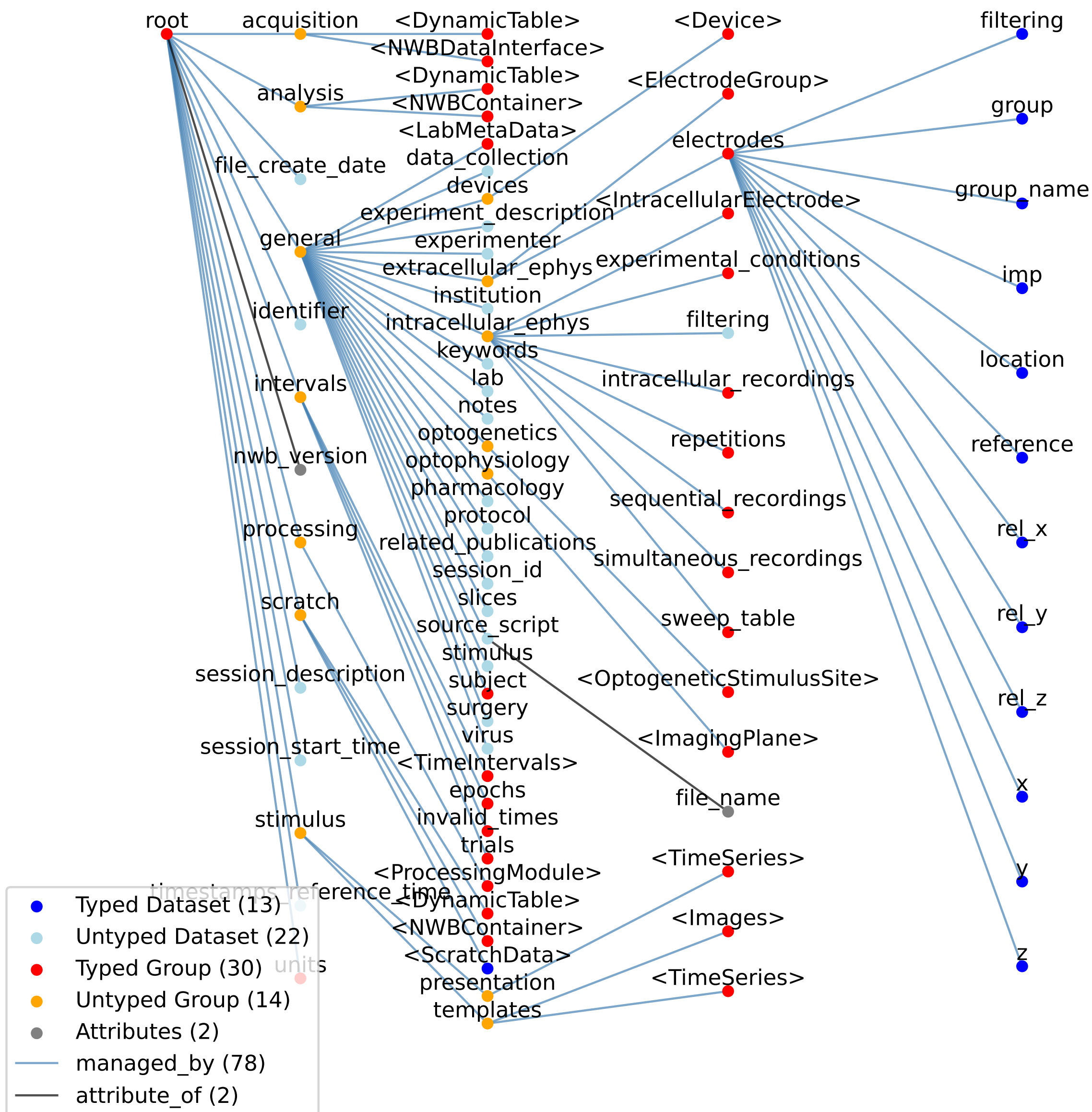


Table 4.22: Datasets, Links, and Attributes contained in <NWBFile>

Id	Type	Description
root	Group	Top level Group for root • **Neurodata Type:** *NWBFile* • **Extends:** *NWBContainer* • **Name:** root
—nwb_version	Attribute	File version string. Use semantic versioning, e.g. 1.2.1. This will be the name of the format with trailing major, minor and patch numbers. • **Data Type:** text • **Value:** 2.6.0 • **Name:** nwb_version
—file_create_date	Dataset	A record of the date the file was created and of subsequent modifications. The date is stored in UTC with local timezone offset as ISO 8601 extended formatted strings: 2018-09-28T14:43:54.123+02:00. Dates stored in UTC end in "Z" with no timezone offset. Date accuracy is up to milliseconds. The file can be created after the experiment was run, so this may differ from the experiment start time. Each modification to the nwb file adds a new entry to the array. • **Data Type:** isodatetime • **Dimensions:** ['num_modifications'] • **Shape:** [None] • **Name:** file_create_date
—identifier	Dataset	A unique text identifier for the file. For example, 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. • **Data Type:** text • **Name:** identifier
—session_description	Dataset	A description of the experimental session and data in the file. • **Data Type:** text • **Name:** session_description
—session_start_time	Dataset	Date and time of the experiment/session start. The date is stored in UTC with local timezone offset as ISO 8601 extended formatted string: 2018-09-28T14:43:54.123+02:00. Dates stored in UTC end in "Z" with no timezone offset. Date accuracy is up to milliseconds. • **Data Type:** isodatetime • **Name:** session_start_time
—timestamps_reference_time	Dataset	Date and time corresponding to time zero of all timestamps. The date is stored in UTC with local timezone offset as ISO 8601 extended formatted string: 2018-09-28T14:43:54.123+02:00. Dates stored in UTC end in "Z" with no timezone offset. Date accuracy is up to milliseconds. All times stored in the file use this time as reference (i.e., time zero). • **Data Type:** isodatetime • **Name:** timestamps_reference_time

Table 4.23: Groups contained in <NWBFile>

Id	Type	Description
root	Group	Top level Group for root • **Neurodata Type:** *NWBFile* • **Extends:** *NWBContainer* • **Name:** root
—acquisition	Group	Data streams recorded from the system, including ephys, ophys, tracking, etc. This group should be 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 NWB files. This will allow keeping bulky raw data out of the file while preserving the option of keeping some/all in the file. Acquired data includes tracking and experimental data streams (i.e., everything measured from the system). If bulky data is stored in the /acquisition group, the data can exist in a separate NWB file that is linked to by the file being used for processing and analysis. • **Name:** acquisition
—analysis	Group	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 (e.g., neurodata_types) and appropriately documented. 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 could be shared with other labs. • **Name:** analysis
—scratch	Group	A place to store one-off analysis results. Data placed here is not intended for sharing. By placing data here, users acknowledge that there is no guarantee that their data meets any standard. • **Quantity:** 0 or 1 • **Name:** scratch
—processing	Group	The home for ProcessingModules. 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. ProcessingModules 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. 'Processing' refers to intermediate analysis of the acquired data to make it more amenable to scientific analysis. • **Name:** processing

continues on next page

Table 4.23 – continued from previous page

Id	Type	Description
—stimulus	Group	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 should be made 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. 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 linked to a remote library file. • **Name:** stimulus
—general	Group	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. 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 (e.g., 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. • **Name:** general
—intervals	Group	Experimental intervals, whether that be logically distinct sub-experiments having a particular scientific goal, trials (see trials subgroup) during an experiment, or epochs (see epochs subgroup) deriving from analysis of data. • **Quantity:** 0 or 1 • **Name:** intervals
—units	Group	Data about sorted spike units. • **Extends:** *Units* • **Quantity:** 0 or 1 • **Name:** units

4.5.2.1 Groups: /acquisition

Data streams recorded from the system, including ephys, ophys, tracking, etc. This group should be 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 NWB files. This will allow keeping bulky raw data out of the file while preserving the option of keeping some/all in the file. Acquired data includes tracking and experimental data streams (i.e., everything measured from the system). If bulky data is stored in the /acquisition group, the data can exist in a separate NWB file that is linked to by the file being used for processing and analysis.

- **Name:** acquisition

Table 4.24: Groups contained in <acquisition>

Id	Type	Description
acquisition	Group	Top level Group for acquisition • **Name:** acquisition
—<*NWBDataInterface*>	Group	Acquired, raw data. • **Extends:** *NWBDataInterface* • **Quantity:** 0 or more
—<DynamicTable>	Group	Tabular data that is relevent to acquisition • **Extends:** DynamicTable • **Quantity:** 0 or more

4.5.2.2 Groups: /acquisition/<NWBDataInterface>

Acquired, raw data.

- **Extends:** *NWBDataInterface*
- **Quantity:** 0 or more

4.5.2.3 Groups: /acquisition/<DynamicTable>

Tabular data that is relevent to acquisition

- **Extends:** DynamicTable
- **Quantity:** 0 or more

4.5.2.4 Groups: /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 (e.g., neurodata_types) and appropriately documented. 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 could be shared with other labs.

- **Name:** analysis

Table 4.25: Groups contained in <analysis>

Id	Type	Description
analysis	Group	Top level Group for analysis • **Name:** analysis
—<*NWBContainer*>	Group	Custom analysis results. • **Extends:** *NWBContainer* • **Quantity:** 0 or more
—<DynamicTable>	Group	Tabular data that is relevent to data stored in analysis • **Extends:** DynamicTable • **Quantity:** 0 or more

4.5.2.5 Groups: /analysis/<NWBContainer>

Custom analysis results.

- **Extends:** *NWBContainer*
- **Quantity:** 0 or more

4.5.2.6 Groups: /analysis/<DynamicTable>

Tabular data that is relevent to data stored in analysis

- **Extends:** DynamicTable
- **Quantity:** 0 or more

4.5.2.7 Groups: /scratch

A place to store one-off analysis results. Data placed here is not intended for sharing. By placing data here, users acknowledge that there is no guarantee that their data meets any standard.

- **Quantity:** 0 or 1
- **Name:** scratch

Table 4.26: Datasets, Links, and Attributes contained in `/scratch`

Id	Type	Description
scratch	Group	Top level Group for scratch • **Quantity:** 0 or 1 • **Name:** scratch
—<*ScratchData*>	Dataset	Any one-off datasets • **Extends:** *ScratchData* • **Quantity:** 0 or more

Table 4.27: Groups contained in <scratch>

Id	Type	Description
scratch	Group	Top level Group for scratch • **Quantity:** 0 or 1 • **Name:** scratch
—<*NWBContainer*>	Group	Any one-off containers • **Extends:** *NWBContainer* • **Quantity:** 0 or more
—<DynamicTable>	Group	Any one-off tables • **Extends:** DynamicTable • **Quantity:** 0 or more

4.5.2.8 Groups: /scratch/<NWBContainer>

Any one-off containers

- **Extends:** *NWBContainer*
- **Quantity:** 0 or more

4.5.2.9 Groups: /scratch/<DynamicTable>

Any one-off tables

- **Extends:** DynamicTable
- **Quantity:** 0 or more

4.5.2.10 Groups: /processing

The home for ProcessingModules. 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. ProcessingModules 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. 'Processing' refers to intermediate analysis of the acquired data to make it more amenable to scientific analysis.

- **Name:** processing

Table 4.28: Groups contained in <processing>

Id	Type	Description
processing	Group	Top level Group for processing • **Name:** processing

continues on next page

Table 4.28 – continued from previous page

Id	Type	Description
—<*ProcessingModule*>	Group	Intermediate analysis of acquired data. • **Extends:** *ProcessingModule* • **Quantity:** 0 or more

4.5.2.11 Groups: /processing/<ProcessingModule>

Intermediate analysis of acquired data.

- **Extends:** *ProcessingModule*
- **Quantity:** 0 or more

4.5.2.12 Groups: /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 should be made 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. 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 linked to a remote library file.

- **Name:** stimulus

Table 4.29: Groups contained in <stimulus>

Id	Type	Description
stimulus	Group	Top level Group for stimulus • **Name:** stimulus
—presentation	Group	Stimuli presented during the experiment. • **Name:** presentation
—templates	Group	Template stimuli. Timestamps 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. • **Name:** templates

4.5.2.13 Groups: /stimulus/presentation

Stimuli presented during the experiment.

- **Name:** presentation

Table 4.30: Groups contained in <presentation>

Id	Type	Description
presentation	Group	Top level Group for presentation • **Name:** presentation
—<*TimeSeries*>	Group	TimeSeries objects containing data of presented stimuli. • **Extends:** *TimeSeries* • **Quantity:** 0 or more

4.5.2.14 Groups: /stimulus/presentation/<TimeSeries>

TimeSeries objects containing data of presented stimuli.

- **Extends:** *TimeSeries*
- **Quantity:** 0 or more

4.5.2.15 Groups: /stimulus/templates

Template stimuli. Timestamps 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.

- **Name:** templates

Table 4.31: Groups contained in <templates>

Id	Type	Description
templates	Group	Top level Group for templates • **Name:** templates
—<*TimeSeries*>	Group	TimeSeries objects containing template data of presented stimuli. • **Extends:** *TimeSeries* • **Quantity:** 0 or more
—<*Images*>	Group	Images objects containing images of presented stimuli. • **Extends:** *Images* • **Quantity:** 0 or more

4.5.2.16 Groups: /stimulus/templates/<TimeSeries>

TimeSeries objects containing template data of presented stimuli.

- **Extends:** *TimeSeries*
- **Quantity:** 0 or more

4.5.2.17 Groups: /stimulus/templates/<Images>

Images objects containing images of presented stimuli.

- **Extends:** *Images*
- **Quantity:** 0 or more

4.5.2.18 Groups: /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. 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 (e.g., 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.

- **Name:** general

Table 4.32: Datasets, Links, and Attributes contained in `/general`

Id	Type	Description
general	Group	Top level Group for general • **Name:** general
—data_collection	Dataset	Notes about data collection and analysis. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** data_collection
—experiment_description	Dataset	General description of the experiment. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** experiment_description

continues on next page

Table 4.32 – continued from previous page

Id	Type	Description
—experimenter	Dataset	Name of person(s) who performed the experiment. Can also specify roles of different people involved. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** [‘num_experimenters’] • **Shape:** [None] • **Name:** experimenter
—institution	Dataset	Institution(s) where experiment was performed. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** institution
—keywords	Dataset	Terms to search over. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** [‘num_keywords’] • **Shape:** [None] • **Name:** keywords
—lab	Dataset	Laboratory where experiment was performed. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** lab
—notes	Dataset	Notes about the experiment. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** notes
—pharmacology	Dataset	Description of drugs used, including how and when they were administered. Anesthesia(s), painkiller(s), etc., plus dosage, concentration, etc. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** pharmacology
—protocol	Dataset	Experimental protocol, if applicable. e.g., include IACUC protocol number. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** protocol
—related_publications	Dataset	Publication information. PMID, DOI, URL, etc. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** [‘num_publications’] • **Shape:** [None] • **Name:** related_publications

continues on next page

Table 4.32 – continued from previous page

Id	Type	Description
—session_id	Dataset	Lab-specific ID for the session. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** session_id
—slices	Dataset	Description of slices, including information about preparation thickness, orientation, temperature, and bath solution. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** slices
—source_script	Dataset	Script file or link to public source code used to create this NWB file. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** source_script
——file_name	Attribute	Name of script file. • **Data Type:** text • **Name:** file_name
—stimulus	Dataset	Notes about stimuli, such as how and where they were presented. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** stimulus
—surgery	Dataset	Narrative description about surgery/surgeries, including date(s) and who performed surgery. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** surgery
—virus	Dataset	Information about virus(es) used in experiments, including virus ID, source, date made, injection location, volume, etc. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** virus

Table 4.33: Groups contained in <general>

Id	Type	Description
general	Group	Top level Group for general • **Name:** general
—<*LabMetaData*>	Group	Place-holder than can be extended so that lab-specific meta-data can be placed in /general. • **Extends:** *LabMetaData* • **Quantity:** 0 or more

continues on next page

Table 4.33 – continued from previous page

Id	Type	Description
—devices	Group	Description of hardware devices used during experiment, e.g., monitors, ADC boards, microscopes, etc. • **Quantity:** 0 or 1 • **Name:** devices
—subject	Group	Information about the animal or person from which the data was measured. • **Extends:** *Subject* • **Quantity:** 0 or 1 • **Name:** subject
—extracellular_ephys	Group	Metadata related to extracellular electrophysiology. • **Quantity:** 0 or 1 • **Name:** extracellular_ephys
—intracellular_ephys	Group	Metadata related to intracellular electrophysiology. • **Quantity:** 0 or 1 • **Name:** intracellular_ephys
—optogenetics	Group	Metadata describing optogenetic stimuluation. • **Quantity:** 0 or 1 • **Name:** optogenetics
—optophysiology	Group	Metadata related to optophysiology. • **Quantity:** 0 or 1 • **Name:** optophysiology

4.5.2.19 Groups: /general/<LabMetaData>

Place-holder than can be extended so that lab-specific meta-data can be placed in /general.

- **Extends:** *LabMetaData*
- **Quantity:** 0 or more

4.5.2.20 Groups: /general/devices

Description of hardware devices used during experiment, e.g., monitors, ADC boards, microscopes, etc.

- **Quantity:** 0 or 1
- **Name:** devices

Table 4.34: Groups contained in <devices>

Id	Type	Description
devices	Group	Top level Group for devices • **Quantity:** 0 or 1 • **Name:** devices
—<*Device*>	Group	Data acquisition devices. • **Extends:** *Device* • **Quantity:** 0 or more

4.5.2.21 Groups: /general/devices/<Device>

Data acquisition devices.

- **Extends:** *Device*
- **Quantity:** 0 or more

4.5.2.22 Groups: /general/subject

Information about the animal or person from which the data was measured.

- **Extends:** *Subject*
- **Quantity:** 0 or 1
- **Name:** subject

4.5.2.23 Groups: /general/extracellular_ephys

Metadata related to extracellular electrophysiology.

- **Quantity:** 0 or 1
- **Name:** extracellular_ephys

Table 4.35: Groups contained in <extracellular_ephys>

Id	Type	Description
extracellular_ephys	Group	Top level Group for extracellular_ephys • **Quantity:** 0 or 1 • **Name:** extracellular_ephys
—<*ElectrodeGroup*>	Group	Physical group of electrodes. • **Extends:** *ElectrodeGroup* • **Quantity:** 0 or more

continues on next page

Table 4.35 – continued from previous page

Id	Type	Description
—electrodes	Group	A table of all electrodes (i.e. channels) used for recording. • **Extends:** DynamicTable • **Quantity:** 0 or 1 • **Name:** electrodes

4.5.2.24 Groups: /general/extracellular_ephys/<ElectrodeGroup>

Physical group of electrodes.

- **Extends:** *ElectrodeGroup*
- **Quantity:** 0 or more

4.5.2.25 Groups: /general/extracellular_ephys/electrodes

A table of all electrodes (i.e. channels) used for recording.

- **Extends:** DynamicTable
- **Quantity:** 0 or 1
- **Name:** electrodes

Table 4.36: Datasets, Links, and Attributes contained in `/general/extracellular_ephys/electrodes`

Id	Type	Description
electrodes	Group	Top level Group for electrodes • **Extends:** DynamicTable • **Quantity:** 0 or 1 • **Name:** electrodes
—x	Dataset	x coordinate of the channel location in the brain (+x is posterior). • **Extends:** VectorData • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** x
—y	Dataset	y coordinate of the channel location in the brain (+y is inferior). • **Extends:** VectorData • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** y
—z	Dataset	z coordinate of the channel location in the brain (+z is right). • **Extends:** VectorData • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** z

continues on next page

Table 4.36 – continued from previous page

Id	Type	Description
—imp	Dataset	Impedance of the channel, in ohms. • **Extends:** VectorData • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** imp
—location	Dataset	Location of the electrode (channel). Specify the area, layer, comments on estimation of area/layer, stereotaxic coordinates if in vivo, etc. Use standard atlas names for anatomical regions when possible. • **Extends:** VectorData • **Data Type:** text • **Name:** location
—filtering	Dataset	Description of hardware filtering, including the filter name and frequency cutoffs. • **Extends:** VectorData • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** filtering
—group	Dataset	Reference to the ElectrodeGroup this electrode is a part of. • **Extends:** VectorData • **Data Type:** object reference to *ElectrodeGroup* • **Name:** group
—group_name	Dataset	Name of the ElectrodeGroup this electrode is a part of. • **Extends:** VectorData • **Data Type:** text • **Name:** group_name
—rel_x	Dataset	x coordinate in electrode group • **Extends:** VectorData • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** rel_x
—rel_y	Dataset	y coordinate in electrode group • **Extends:** VectorData • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** rel_y
—rel_z	Dataset	z coordinate in electrode group • **Extends:** VectorData • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** rel_z

continues on next page

Table 4.36 – continued from previous page

Id	Type	Description
—reference	Dataset	Description of the reference electrode and/or reference scheme used for this electrode, e.g., "stainless steel skull screw" or "online common average referencing". • **Extends:** VectorData • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** reference

4.5.2.26 Groups: /general/intracellular_ephys

Metadata related to intracellular electrophysiology.

- **Quantity:** 0 or 1
- **Name:** intracellular_ephys

Table 4.37: Datasets, Links, and Attributes contained in `/general/intracellular_ephys`

Id	Type	Description
intracellular_ephys	Group	Top level Group for intracellular_ephys • **Quantity:** 0 or 1 • **Name:** intracellular_ephys
—filtering	Dataset	[DEPRECATED] Use IntracellularElectrode.filtering instead. 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. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** filtering

Table 4.38: Groups contained in <intracellular_ephys>

Id	Type	Description
intracellular_ephys	Group	Top level Group for intracellular_ephys • **Quantity:** 0 or 1 • **Name:** intracellular_ephys
—<*IntracellularElectrode*>	Group	An intracellular electrode. • **Extends:** *IntracellularElectrode* • **Quantity:** 0 or more

continues on next page

Table 4.38 – continued from previous page

Id	Type	Description
—sweep_table	Group	[DEPRECATED] Table used to group different PatchClampSeries. SweepTable is being replaced by IntracellularRecordingsTable and SimultaneousRecordingsTable tabels. Additional SequentialRecordingsTable, RepetitionsTable and ExperimentalConditions tables provide enhanced support for experiment metadata. • **Extends:** *SweepTable* • **Quantity:** 0 or 1 • **Name:** sweep_table
—intracellular_recordings	Group	A table to group together a stimulus and response from a single electrode and a single simultaneous recording. Each row in the table represents a single recording consisting typically of a stimulus and a corresponding response. In some cases, however, only a stimulus or a response are recorded as as part of an experiment. In this case both, the stimulus and response will point to the same TimeSeries while the idx_start and count of the invalid column will be set to -1, thus, indicating that no values have been recorded for the stimulus or response, respectively. Note, a recording MUST contain at least a stimulus or a response. Typically the stimulus and response are PatchClampSeries. However, the use of AD/DA channels that are not associated to an electrode is also common in intracellular electrophysiology, in which case other TimeSeries may be used. • **Extends:** *IntracellularRecordingsTable* • **Quantity:** 0 or 1 • **Name:** intracellular_recordings
—simultaneous_recordings	Group	A table for grouping different intracellular recordings from the IntracellularRecordingsTable table together that were recorded simultaneously from different electrodes • **Extends:** *SimultaneousRecordingsTable* • **Quantity:** 0 or 1 • **Name:** simultaneous_recordings
—sequential_recordings	Group	A table for grouping different sequential recordings from the SimultaneousRecordingsTable table together. This is typically used to group together sequential recordings where the a sequence of stimuli of the same type with varying parameters have been presented in a sequence. • **Extends:** *SequentialRecordingsTable* • **Quantity:** 0 or 1 • **Name:** sequential_recordings
—repetitions	Group	A table for grouping different sequential intracellular recordings together. With each SequentialRecording typically representing a particular type of stimulus, the RepetitionsTable table is typically used to group sets of stimuli applied in sequence. • **Extends:** *RepetitionsTable* • **Quantity:** 0 or 1 • **Name:** repetitions

continues on next page

Table 4.38 – continued from previous page

Id	Type	Description
—experimental_conditions	Group	A table for grouping different intracellular recording repetitions together that belong to the same experimental experimental_conditions. • **Extends:** *ExperimentalConditionsTable* • **Quantity:** 0 or 1 • **Name:** experimental_conditions

4.5.2.27 Groups: /general/intracellular_ephys/<IntracellularElectrode>

An intracellular electrode.

- **Extends:** *IntracellularElectrode*
- **Quantity:** 0 or more

4.5.2.28 Groups: /general/intracellular_ephys/sweep_table

[DEPRECATED] Table used to group different PatchClampSeries. SweepTable is being replaced by IntracellularRecordingsTable and SimultaneousRecordingsTable tabels. Additional SequentialRecordingsTable, RepetitionsTable and ExperimentalConditions tables provide enhanced support for experiment metadata.

- **Extends:** *SweepTable*
- **Quantity:** 0 or 1
- **Name:** sweep_table

4.5.2.29 Groups: /general/intracellular_ephys/intracellular_recordings

A table to group together a stimulus and response from a single electrode and a single simultaneous recording. Each row in the table represents a single recording consisting typically of a stimulus and a corresponding response. In some cases, however, only a stimulus or a response are recorded as as part of an experiment. In this case both, the stimulus and response will point to the same TimeSeries while the idx_start and count of the invalid column will be set to -1, thus, indicating that no values have been recorded for the stimulus or response, respectively. Note, a recording MUST contain at least a stimulus or a response. Typically the stimulus and response are PatchClampSeries. However, the use of AD/DA channels that are not associated to an electrode is also common in intracellular electrophysiology, in which case other TimeSeries may be used.

- **Extends:** *IntracellularRecordingsTable*
- **Quantity:** 0 or 1
- **Name:** intracellular_recordings

4.5.2.30 Groups: /general/intracellular_ephys/simultaneous_recordings

A table for grouping different intracellular recordings from the IntracellularRecordingsTable table together that were recorded simultaneously from different electrodes

- **Extends:** *SimultaneousRecordingsTable*
- **Quantity:** 0 or 1
- **Name:** simultaneous_recordings

4.5.2.31 Groups: /general/intracellular_ephys/sequential_recordings

A table for grouping different sequential recordings from the SimultaneousRecordingsTable table together. This is typically used to group together sequential recordings where the a sequence of stimuli of the same type with varying parameters have been presented in a sequence.

- **Extends:** *SequentialRecordingsTable*
- **Quantity:** 0 or 1
- **Name:** sequential_recordings

4.5.2.32 Groups: /general/intracellular_ephys/repetitions

A table for grouping different sequential intracellular recordings together. With each SequentialRecording typically representing a particular type of stimulus, the RepetitionsTable table is typically used to group sets of stimuli applied in sequence.

- **Extends:** *RepetitionsTable*
- **Quantity:** 0 or 1
- **Name:** repetitions

4.5.2.33 Groups: /general/intracellular_ephys/experimental_conditions

A table for grouping different intracellular recording repetitions together that belong to the same experimental experimental_conditions.

- **Extends:** *ExperimentalConditionsTable*
- **Quantity:** 0 or 1
- **Name:** experimental_conditions

4.5.2.34 Groups: /general/optogenetics

Metadata describing optogenetic stimuluation.

- **Quantity:** 0 or 1
- **Name:** optogenetics

Table 4.39: Groups contained in <optogenetics>

Id	Type	Description
optogenetics	Group	Top level Group for optogenetics • **Quantity:** 0 or 1 • **Name:** optogenetics
—<*OptogeneticStimulusSite*>	Group	An optogenetic stimulation site. • **Extends:** *OptogeneticStimulusSite* • **Quantity:** 0 or more

4.5.2.35 Groups: /general/optogenetics/<OptogeneticStimulusSite>

An optogenetic stimulation site.

- **Extends:** *OptogeneticStimulusSite*
- **Quantity:** 0 or more

4.5.2.36 Groups: /general/optophysiology

Metadata related to optophysiology.

- **Quantity:** 0 or 1
- **Name:** optophysiology

Table 4.40: Groups contained in <optophysiology>

Id	Type	Description
optophysiology	Group	Top level Group for optophysiology • **Quantity:** 0 or 1 • **Name:** optophysiology
—<*ImagingPlane*>	Group	An imaging plane. • **Extends:** *ImagingPlane* • **Quantity:** 0 or more

4.5.2.37 Groups: /general/optophysiology/<ImagingPlane>

An imaging plane.

- **Extends:** *ImagingPlane*
- **Quantity:** 0 or more

4.5.2.38 Groups: /intervals

Experimental intervals, whether that be logically distinct sub-experiments having a particular scientific goal, trials (see trials subgroup) during an experiment, or epochs (see epochs subgroup) deriving from analysis of data.

- **Quantity:** 0 or 1
- **Name:** intervals

Table 4.41: Groups contained in <intervals>

Id	Type	Description
intervals	Group	Top level Group for intervals • **Quantity:** 0 or 1 • **Name:** intervals
—epochs	Group	Divisions in time marking experimental stages or sub-divisions of a single recording session. • **Extends:** *TimeIntervals* • **Quantity:** 0 or 1 • **Name:** epochs
—trials	Group	Repeated experimental events that have a logical grouping. • **Extends:** *TimeIntervals* • **Quantity:** 0 or 1 • **Name:** trials
—invalid_times	Group	Time intervals that should be removed from analysis. • **Extends:** *TimeIntervals* • **Quantity:** 0 or 1 • **Name:** invalid_times
—<*TimeIntervals*>	Group	Optional additional table(s) for describing other experimental time intervals. • **Extends:** *TimeIntervals* • **Quantity:** 0 or more

4.5.2.39 Groups: /intervals/epochs

Divisions in time marking experimental stages or sub-divisions of a single recording session.

- **Extends:** *TimeIntervals*
- **Quantity:** 0 or 1
- **Name:** epochs

4.5.2.40 Groups: /intervals/trials

Repeated experimental events that have a logical grouping.

- **Extends:** *TimeIntervals*
- **Quantity:** 0 or 1
- **Name:** trials

4.5.2.41 Groups: /intervals/invalid_times

Time intervals that should be removed from analysis.

- **Extends:** *TimeIntervals*
- **Quantity:** 0 or 1
- **Name:** invalid_times

4.5.2.42 Groups: /intervals/<TimeIntervals>

Optional additional table(s) for describing other experimental time intervals.

- **Extends:** *TimeIntervals*
- **Quantity:** 0 or more

4.5.2.43 Groups: /units

Data about sorted spike units.

- **Extends:** *Units*
- **Quantity:** 0 or 1
- **Name:** units

4.5.3 LabMetaData

Overview: Lab-specific meta-data.

`LabMetaData` extends `NWBContainer` and includes all elements of *NWBContainer* with the following additions or changes.

- **Extends:** *NWBContainer*
- **Primitive Type:** Group
- **Inherits from:** *NWBContainer*, Container
- **Source filename:** nwb.file.yaml
- **Source Specification:** see Section 5.6.3

4.5.4 Subject

Overview: Information about the animal or person from which the data was measured.

`Subject` extends `NWBContainer` and includes all elements of *NWBContainer* with the following additions or changes.

- **Extends:** *NWBContainer*
- **Primitive Type:** Group
- **Inherits from:** *NWBContainer*, Container
- **Source filename:** nwb.file.yaml
- **Source Specification:** see Section 5.6.4

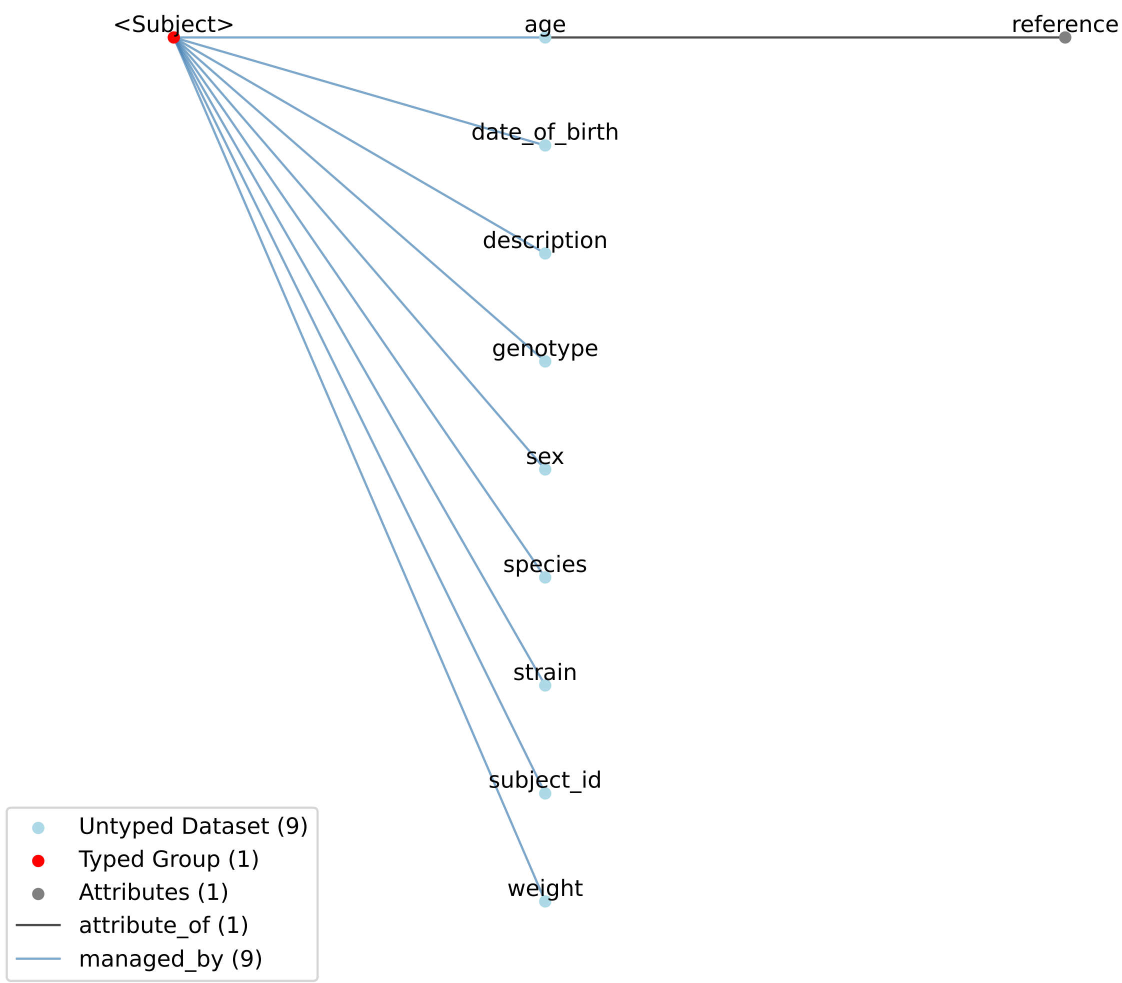


Table 4.42: Datasets, Links, and Attributes contained in <Subject>

Id	Type	Description
<Subject>	Group	Top level Group for <Subject> • **Neurodata Type:** *Subject* • **Extends:** *NWBContainer*
—age	Dataset	Age of subject. Can be supplied instead of 'date_of_birth'. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** age
——reference	Attribute	Age is with reference to this event. Can be 'birth' or 'gestational'. If reference is omitted, 'birth' is implied. • **Data Type:** text • **Required:** False • **Default Value:** birth • **Name:** reference
—date_of_birth	Dataset	Date of birth of subject. Can be supplied instead of 'age'. • **Quantity:** 0 or 1 • **Data Type:** isodatetime • **Name:** date_of_birth
—description	Dataset	Description of subject and where subject came from (e.g., breeder, if animal). • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** description
—genotype	Dataset	Genetic strain. If absent, assume Wild Type (WT). • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** genotype
—sex	Dataset	Gender of subject. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** sex
—species	Dataset	Species of subject. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** species
—strain	Dataset	Strain of subject. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** strain

continues on next page

Table 4.42 – continued from previous page

Id	Type	Description
—subject_id	Dataset	ID of animal/person used/participating in experiment (lab convention). • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** subject_id
—weight	Dataset	Weight at time of experiment, at time of surgery and at other important times. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** weight

4.6 Miscellaneous neurodata_types.

Miscellaneous types.

4.6.1 AbstractFeatureSeries

Overview: 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. This is useful when storing the raw stimulus is impractical.

`AbstractFeatureSeries` extends `TimeSeries` and includes all elements of *TimeSeries* with the following additions or changes.

- **Extends:** *TimeSeries*
- **Primitive Type:** Group
- **Inherits from:** *TimeSeries*, *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.misc.yaml
- **Source Specification:** see Section 5.7.1

Table 4.43: Datasets, Links, and Attributes contained in <AbstractFeatureSeries>

Id	Type	Description
<AbstractFeatureSeries>	Group	Top level Group for <AbstractFeatureSeries> • **Neurodata Type:** *AbstractFeatureSeries* • **Extends:** *TimeSeries*
—description	Attribute	Description of the time series. • **Data Type:** text • **Required:** False • **Default Value:** no description • **Name:** description
—comments	Attribute	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. • **Data Type:** text • **Required:** False • **Default Value:** no comments • **Name:** comments

continues on next page

Table 4.43 – continued from previous page

Id	Type	Description
—data	Dataset	Values of each feature at each time. • **Data Type:** numeric • **Dimensions:** [['num_times'], ['num_times', 'num_features']] • **Shape:** [[None], [None, None]] • **Name:** data
——unit	Attribute	Since there can be different units for different features, store the units in 'feature_units'. The default value for this attribute is "see 'feature_units'". • **Data Type:** text • **Required:** False • **Default Value:** see 'feature_units' • **Name:** unit
——conversion	Attribute	Scalar to multiply each element in data to convert it to the specified 'unit'. If the data are stored in acquisition system units or other units that require a conversion to be interpretable, multiply the data by 'conversion' to convert the data to the specified 'unit'. e.g. if the data acquisition system stores values in this object as signed 16-bit integers (int16 range -32,768 to 32,767) that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition values to recorded volts is 2.5/32768/8000 = 9.5367e-9. • **Data Type:** float32 • **Required:** False • **Default Value:** 1.0 • **Name:** conversion
——offset	Attribute	Scalar to add to the data after scaling by 'conversion' to finalize its coercion to the specified 'unit'. Two common examples of this include (a) data stored in an unsigned type that requires a shift after scaling to re-center the data, and (b) specialized recording devices that naturally cause a scalar offset with respect to the true units. • **Data Type:** float32 • **Required:** False • **Default Value:** 0.0 • **Name:** offset
——resolution	Attribute	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 -1.0. • **Data Type:** float32 • **Required:** False • **Default Value:** -1.0 • **Name:** resolution

continues on next page

Table 4.43 – continued from previous page

Id	Type	Description
——continuity	Attribute	Optionally describe the continuity of the data. Can be “continuous”, “instantaneous”, or “step”. For example, a voltage trace would be “continuous”, because samples are recorded from a continuous process. An array of lick times would be “instantaneous”, because the data represents distinct moments in time. Times of image presentations would be “step” because the picture remains the same until the next timepoint. This field is optional, but is useful in providing information about the underlying data. It may inform the way this data is interpreted, the way it is visualized, and what analysis methods are applicable. • **Data Type:** text • **Required:** False • **Name:** continuity
—feature_units	Dataset	Units of each feature. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** [‘num_features’] • **Shape:** [None] • **Name:** feature_units
—features	Dataset	Description of the features represented in TimeSeries::data. • **Data Type:** text • **Dimensions:** [‘num_features’] • **Shape:** [None] • **Name:** features
—starting_time	Dataset	Timestamp of the first sample in seconds. When timestamps are uniformly spaced, the timestamp of the first sample can be specified and all subsequent ones calculated from the sampling rate attribute. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Name:** starting_time
——rate	Attribute	Sampling rate, in Hz. • **Data Type:** float32 • **Name:** rate
——unit	Attribute	Unit of measurement for time, which is fixed to ‘seconds’. • **Data Type:** text • **Value:** seconds • **Name:** unit

continues on next page

Table 4.43 – continued from previous page

Id	Type	Description
—timestamps	Dataset	Timestamps for samples stored in data, in seconds, relative to the common experiment master-clock stored in NWBFile.timestamps_reference_time. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** timestamps
——interval	Attribute	Value is '1' • **Data Type:** int32 • **Value:** 1 • **Name:** interval
——unit	Attribute	Unit of measurement for timestamps, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—control	Dataset	Numerical labels that apply to each time point in data for the purpose of querying and slicing data by these values. If present, the length of this array should be the same size as the first dimension of data. • **Quantity:** 0 or 1 • **Data Type:** uint8 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** control
—control_description	Dataset	Description of each control value. Must be present if control is present. If present, control_description[0] should describe time points where control == 0. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** ['num_control_values'] • **Shape:** [None] • **Name:** control_description

Table 4.44: Groups contained in <AbstractFeatureSeries>

Id	Type	Description
<AbstractFeatureSeries>	Group	Top level Group for <AbstractFeatureSeries> • **Neurodata Type:** *AbstractFeatureSeries* • **Extends:** *TimeSeries*

continues on next page

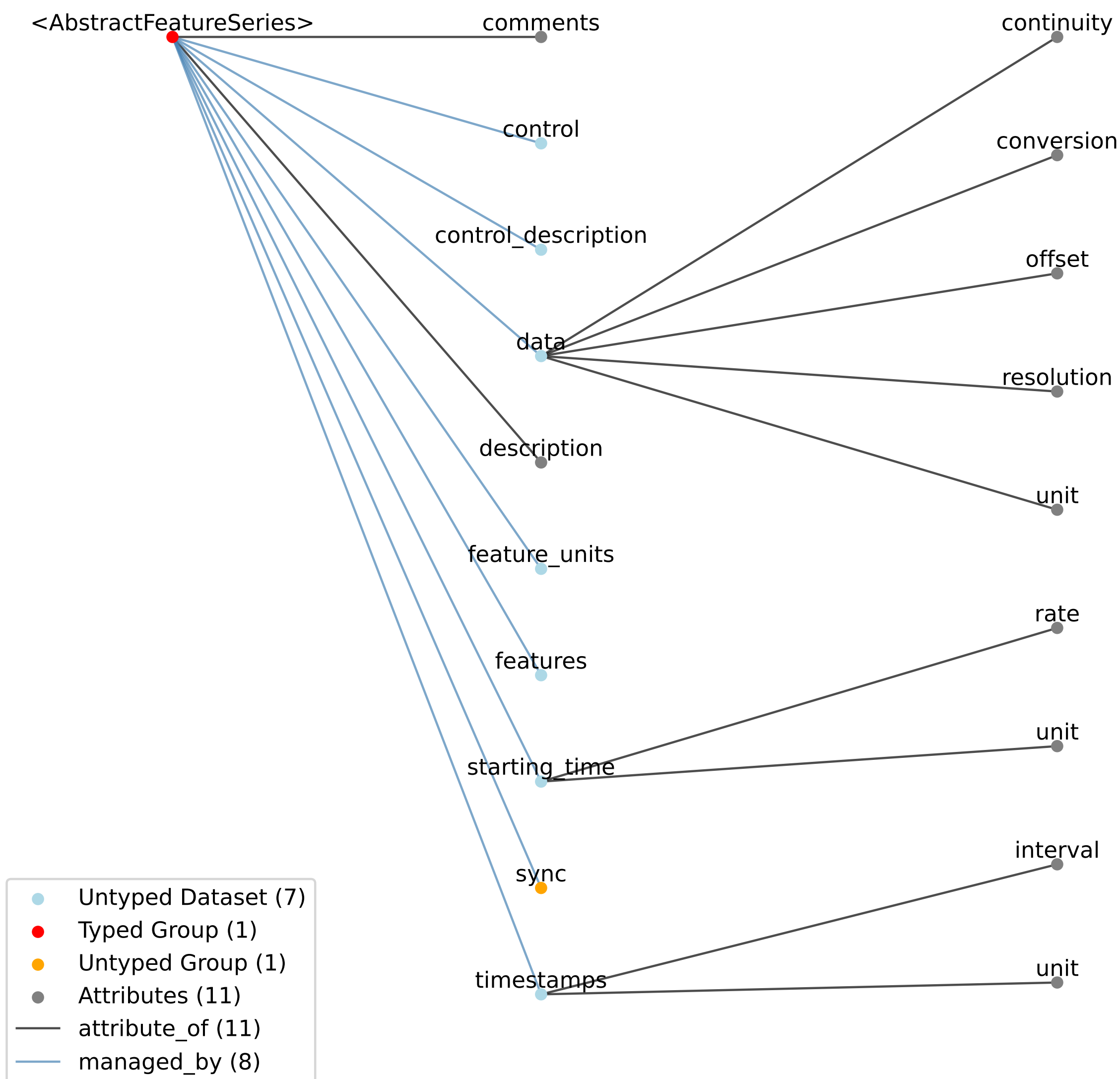
Table 4.44 – continued from previous page

Id	Type	Description
—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. • **Quantity:** 0 or 1 • **Name:** sync

4.6.1.1 Groups: sync

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.

- **Quantity:** 0 or 1
- **Name:** sync


<AbstractFeatureSeries>
comments
control
control_description
data
description
feature_units
features
starting_time
sync
timestamps
continuity
conversion
offset
resolution
unit
rate
unit
interval
unit
Untyped Dataset (7)
Typed Group (1)
Untyped Group (1)
Attributes (11)
attribute_of (11)
managed_by (8)

4.6.2 AnnotationSeries

Overview: Stores user annotations made during an experiment. The 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` extends `TimeSeries` and includes all elements of *TimeSeries* with the following additions or changes.

- **Extends:** *TimeSeries*
- **Primitive Type:** Group
- **Inherits from:** *TimeSeries*, *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.misc.yaml
- **Source Specification:** see Section 5.7.2

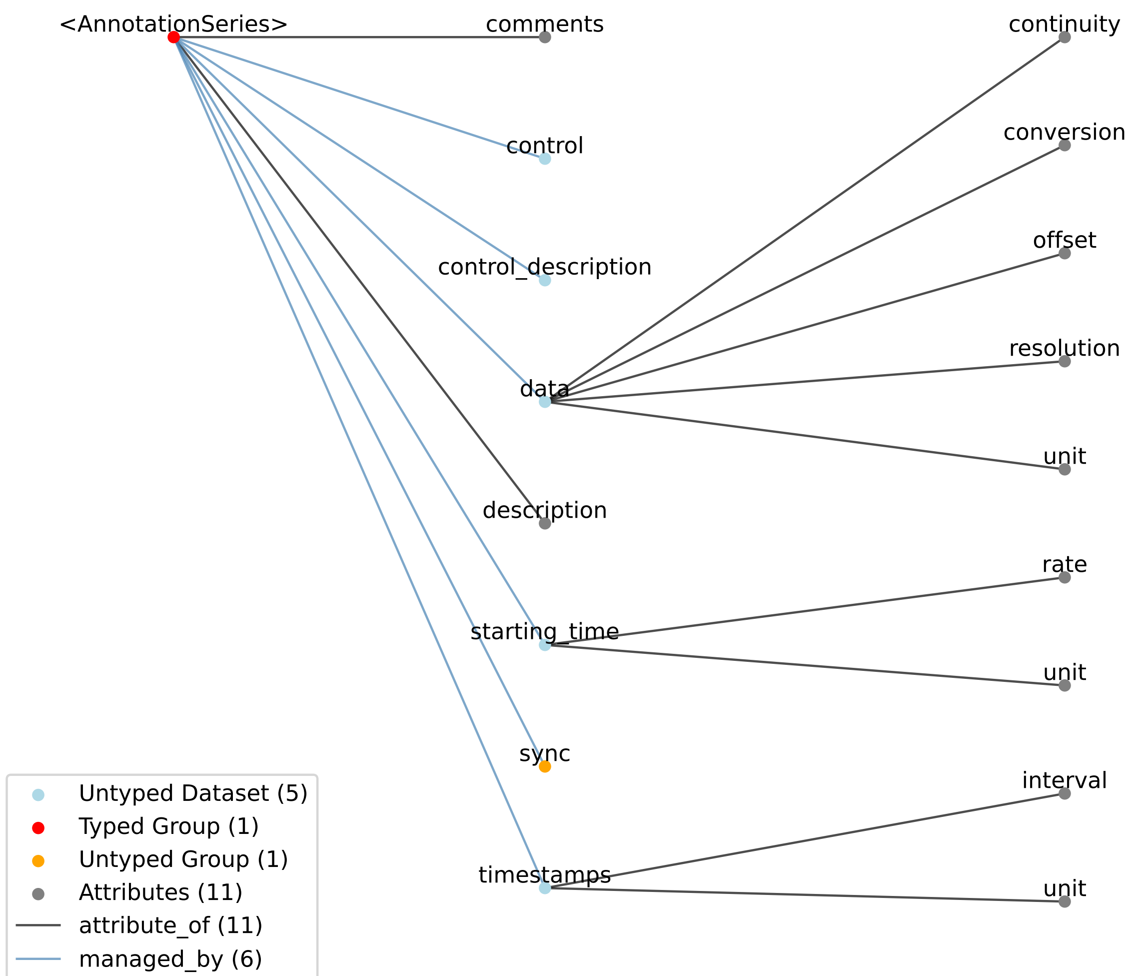


Table 4.45: Datasets, Links, and Attributes contained in <AnnotationSeries>

Id	Type	Description
<AnnotationSeries>	Group	Top level Group for <AnnotationSeries> • **Neurodata Type:** *AnnotationSeries* • **Extends:** *TimeSeries*
—description	Attribute	Description of the time series. • **Data Type:** text • **Required:** False • **Default Value:** no description • **Name:** description
—comments	Attribute	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. • **Data Type:** text • **Required:** False • **Default Value:** no comments • **Name:** comments
—data	Dataset	Annotations made during an experiment. • **Data Type:** text • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** data
——resolution	Attribute	Smallest meaningful difference between values in data. Annotations have no units, so the value is fixed to -1.0. • **Data Type:** float32 • **Value:** -1.0 • **Name:** resolution
——unit	Attribute	Base unit of measurement for working with the data. Annotations have no units, so the value is fixed to 'n/a'. • **Data Type:** text • **Value:** n/a • **Name:** unit

continues on next page

Table 4.45 – continued from previous page

Id	Type	Description
——conversion	Attribute	Scalar to multiply each element in data to convert it to the specified 'unit'. If the data are stored in acquisition system units or other units that require a conversion to be interpretable, multiply the data by 'conversion' to convert the data to the specified 'unit'. e.g. if the data acquisition system stores values in this object as signed 16-bit integers (int16 range -32,768 to 32,767) that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition values to recorded volts is 2.5/32768/8000 = 9.5367e-9. • **Data Type:** float32 • **Required:** False • **Default Value:** 1.0 • **Name:** conversion
——offset	Attribute	Scalar to add to the data after scaling by 'conversion' to finalize its coercion to the specified 'unit'. Two common examples of this include (a) data stored in an unsigned type that requires a shift after scaling to re-center the data, and (b) specialized recording devices that naturally cause a scalar offset with respect to the true units. • **Data Type:** float32 • **Required:** False • **Default Value:** 0.0 • **Name:** offset
——continuity	Attribute	Optionally describe the continuity of the data. Can be "continuous", "instantaneous", or "step". For example, a voltage trace would be "continuous", because samples are recorded from a continuous process. An array of lick times would be "instantaneous", because the data represents distinct moments in time. Times of image presentations would be "step" because the picture remains the same until the next timepoint. This field is optional, but is useful in providing information about the underlying data. It may inform the way this data is interpreted, the way it is visualized, and what analysis methods are applicable. • **Data Type:** text • **Required:** False • **Name:** continuity
—starting_time	Dataset	Timestamp of the first sample in seconds. When timestamps are uniformly spaced, the timestamp of the first sample can be specified and all subsequent ones calculated from the sampling rate attribute. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Name:** starting_time
——rate	Attribute	Sampling rate, in Hz. • **Data Type:** float32 • **Name:** rate

continues on next page

Table 4.45 – continued from previous page

Id	Type	Description
——unit	Attribute	Unit of measurement for time, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—timestamps	Dataset	Timestamps for samples stored in data, in seconds, relative to the common experiment master-clock stored in NWBFile.timestamps_reference_time. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** timestamps
——interval	Attribute	Value is '1' • **Data Type:** int32 • **Value:** 1 • **Name:** interval
——unit	Attribute	Unit of measurement for timestamps, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—control	Dataset	Numerical labels that apply to each time point in data for the purpose of querying and slicing data by these values. If present, the length of this array should be the same size as the first dimension of data. • **Quantity:** 0 or 1 • **Data Type:** uint8 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** control
—control_description	Dataset	Description of each control value. Must be present if control is present. If present, control_description[0] should describe time points where control == 0. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** ['num_control_values'] • **Shape:** [None] • **Name:** control_description

Table 4.46: Groups contained in <AnnotationSeries>

Id	Type	Description
<AnnotationSeries>	Group	Top level Group for <AnnotationSeries> • **Neurodata Type:** *AnnotationSeries* • **Extends:** *TimeSeries*
—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. • **Quantity:** 0 or 1 • **Name:** sync

4.6.2.1 Groups: sync

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.

- **Quantity:** 0 or 1
- **Name:** sync

4.6.3 IntervalSeries

Overview: 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` extends `TimeSeries` and includes all elements of *TimeSeries* with the following additions or changes.

- **Extends:** *TimeSeries*
- **Primitive Type:** Group
- **Inherits from:** *TimeSeries*, *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.misc.yaml
- **Source Specification:** see Section 5.7.3

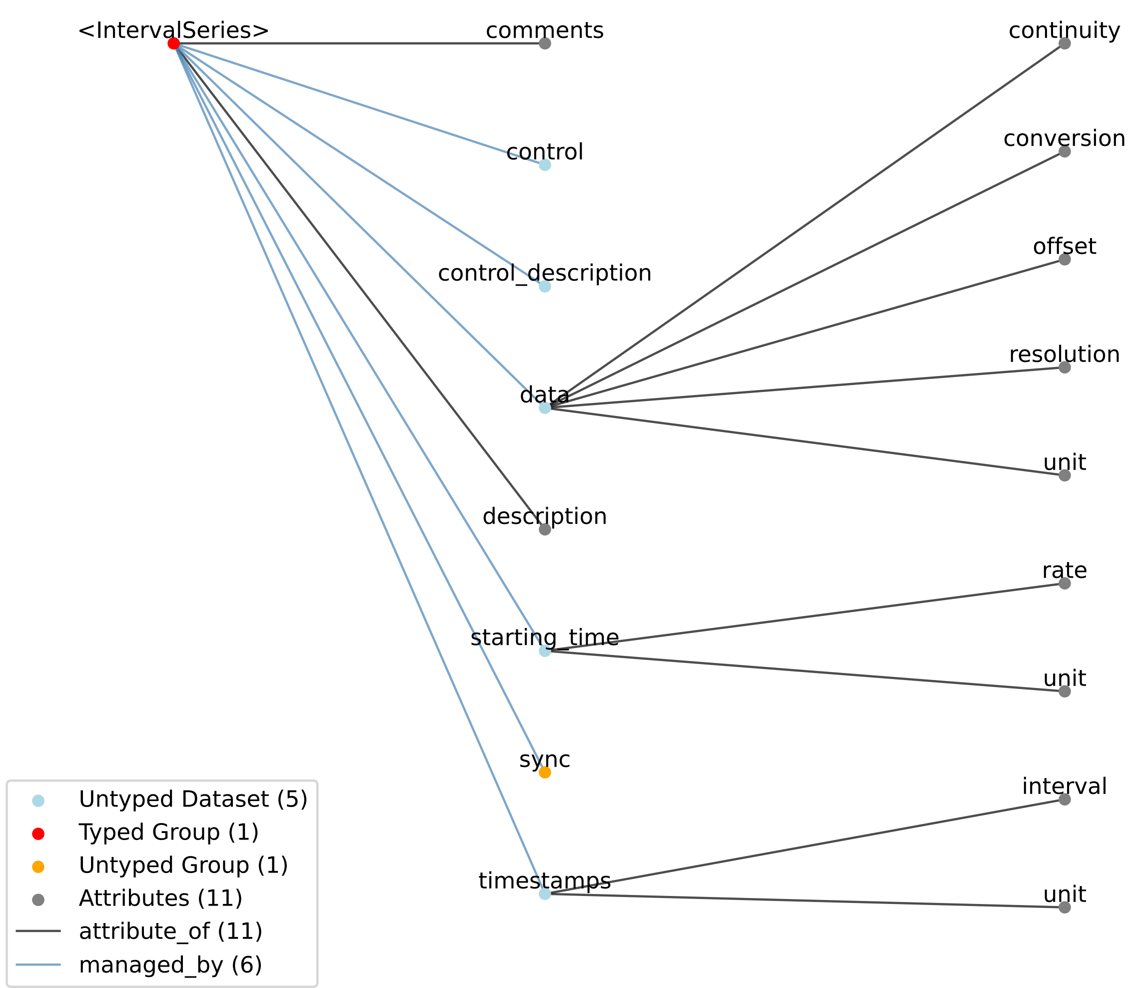


Table 4.47: Datasets, Links, and Attributes contained in <IntervalSeries>

Id	Type	Description
<IntervalSeries>	Group	Top level Group for <IntervalSeries> • **Neurodata Type:** *IntervalSeries* • **Extends:** *TimeSeries*
—description	Attribute	Description of the time series. • **Data Type:** text • **Required:** False • **Default Value:** no description • **Name:** description
—comments	Attribute	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. • **Data Type:** text • **Required:** False • **Default Value:** no comments • **Name:** comments
—data	Dataset	Use values >0 if interval started, <0 if interval ended. • **Data Type:** int8 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** data
——resolution	Attribute	Smallest meaningful difference between values in data. Annotations have no units, so the value is fixed to -1.0. • **Data Type:** float32 • **Value:** -1.0 • **Name:** resolution
——unit	Attribute	Base unit of measurement for working with the data. Annotations have no units, so the value is fixed to 'n/a'. • **Data Type:** text • **Value:** n/a • **Name:** unit

continues on next page

Table 4.47 – continued from previous page

Id	Type	Description
——conversion	Attribute	Scalar to multiply each element in data to convert it to the specified 'unit'. If the data are stored in acquisition system units or other units that require a conversion to be interpretable, multiply the data by 'conversion' to convert the data to the specified 'unit'. e.g. if the data acquisition system stores values in this object as signed 16-bit integers (int16 range -32,768 to 32,767) that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition values to recorded volts is 2.5/32768/8000 = 9.5367e-9. • **Data Type:** float32 • **Required:** False • **Default Value:** 1.0 • **Name:** conversion
——offset	Attribute	Scalar to add to the data after scaling by 'conversion' to finalize its coercion to the specified 'unit'. Two common examples of this include (a) data stored in an unsigned type that requires a shift after scaling to re-center the data, and (b) specialized recording devices that naturally cause a scalar offset with respect to the true units. • **Data Type:** float32 • **Required:** False • **Default Value:** 0.0 • **Name:** offset
——continuity	Attribute	Optionally describe the continuity of the data. Can be "continuous", "instantaneous", or "step". For example, a voltage trace would be "continuous", because samples are recorded from a continuous process. An array of lick times would be "instantaneous", because the data represents distinct moments in time. Times of image presentations would be "step" because the picture remains the same until the next timepoint. This field is optional, but is useful in providing information about the underlying data. It may inform the way this data is interpreted, the way it is visualized, and what analysis methods are applicable. • **Data Type:** text • **Required:** False • **Name:** continuity
—starting_time	Dataset	Timestamp of the first sample in seconds. When timestamps are uniformly spaced, the timestamp of the first sample can be specified and all subsequent ones calculated from the sampling rate attribute. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Name:** starting_time
——rate	Attribute	Sampling rate, in Hz. • **Data Type:** float32 • **Name:** rate

continues on next page

Table 4.47 – continued from previous page

Id	Type	Description
——unit	Attribute	Unit of measurement for time, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—timestamps	Dataset	Timestamps for samples stored in data, in seconds, relative to the common experiment master-clock stored in NWBFile.timestamps_reference_time. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** timestamps
——interval	Attribute	Value is '1' • **Data Type:** int32 • **Value:** 1 • **Name:** interval
——unit	Attribute	Unit of measurement for timestamps, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—control	Dataset	Numerical labels that apply to each time point in data for the purpose of querying and slicing data by these values. If present, the length of this array should be the same size as the first dimension of data. • **Quantity:** 0 or 1 • **Data Type:** uint8 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** control
—control_description	Dataset	Description of each control value. Must be present if control is present. If present, control_description[0] should describe time points where control == 0. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** ['num_control_values'] • **Shape:** [None] • **Name:** control_description

Table 4.48: Groups contained in <IntervalSeries>

Id	Type	Description
<IntervalSeries>	Group	Top level Group for <IntervalSeries> • **Neurodata Type:** *IntervalSeries* • **Extends:** *TimeSeries*
—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. • **Quantity:** 0 or 1 • **Name:** sync

4.6.3.1 Groups: sync

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.

- **Quantity:** 0 or 1
- **Name:** sync

4.6.4 DecompositionSeries

Overview: Spectral analysis of a time series, e.g. of an LFP or a speech signal.

`DecompositionSeries` extends `TimeSeries` and includes all elements of *TimeSeries* with the following additions or changes.

- **Extends:** *TimeSeries*
- **Primitive Type:** Group
- **Inherits from:** *TimeSeries*, *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.misc.yaml
- **Source Specification:** see Section 5.7.4

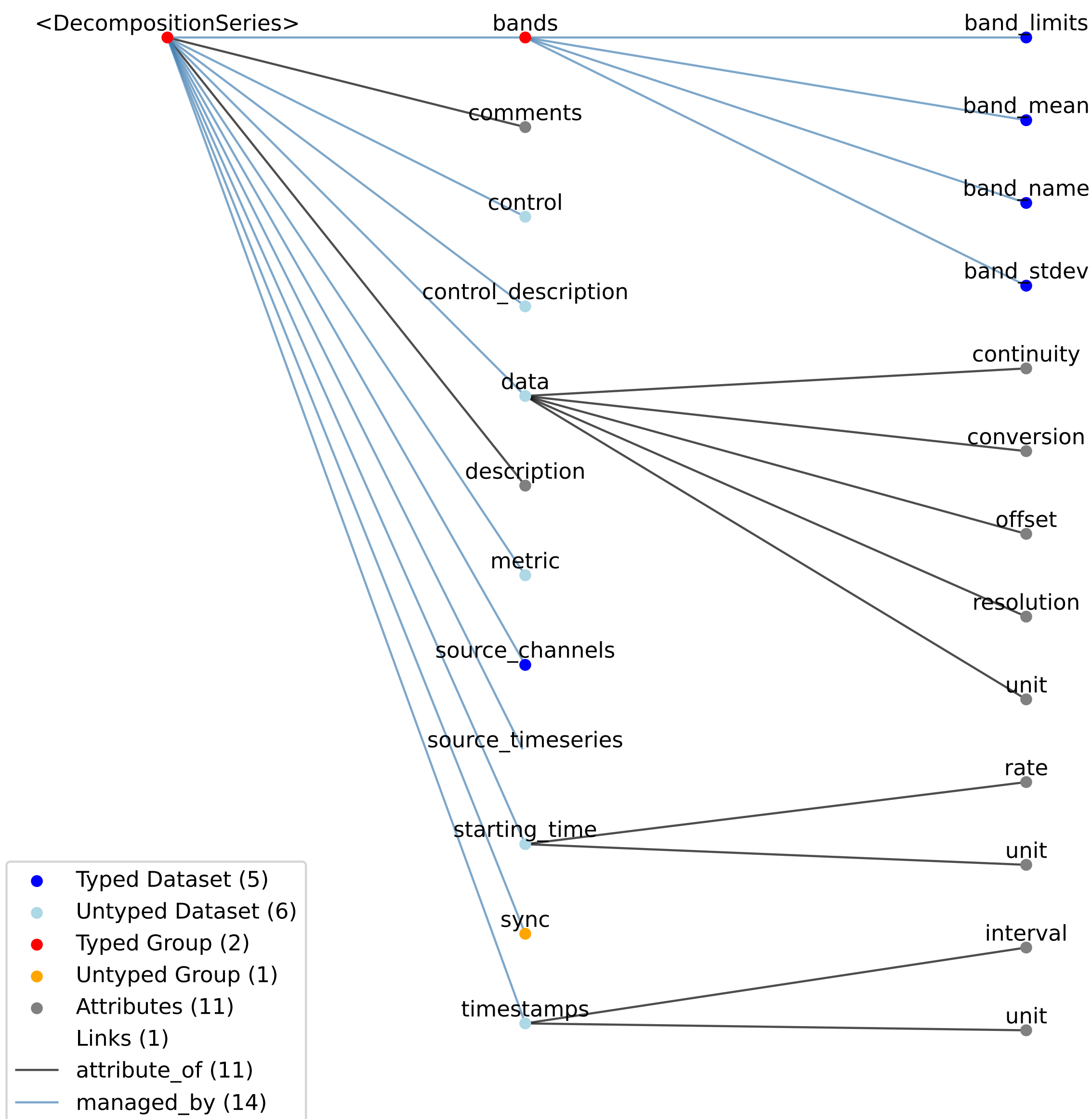


Table 4.49: Datasets, Links, and Attributes contained in <DecompositionSeries>

Id	Type	Description
<DecompositionSeries>	Group	Top level Group for <DecompositionSeries> • **Neurodata Type:** *DecompositionSeries* • **Extends:** *TimeSeries*
—description	Attribute	Description of the time series. • **Data Type:** text • **Required:** False • **Default Value:** no description • **Name:** description
—comments	Attribute	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. • **Data Type:** text • **Required:** False • **Default Value:** no comments • **Name:** comments
—data	Dataset	Data decomposed into frequency bands. • **Data Type:** numeric • **Dimensions:** ['num_times', 'num_channels', 'num_bands'] • **Shape:** [None, None, None] • **Name:** data
——unit	Attribute	Base unit of measurement for working with the data. Actual stored values are not necessarily stored in these units. To access the data in these units, multiply 'data' by 'conversion'. • **Data Type:** text • **Required:** False • **Default Value:** no unit • **Name:** unit
——conversion	Attribute	Scalar to multiply each element in data to convert it to the specified 'unit'. If the data are stored in acquisition system units or other units that require a conversion to be interpretable, multiply the data by 'conversion' to convert the data to the specified 'unit'. e.g. if the data acquisition system stores values in this object as signed 16-bit integers (int16 range -32,768 to 32,767) that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition values to recorded volts is 2.5/32768/8000 = 9.5367e-9. • **Data Type:** float32 • **Required:** False • **Default Value:** 1.0 • **Name:** conversion

continues on next page

Table 4.49 – continued from previous page

Id	Type	Description
——offset	Attribute	Scalar to add to the data after scaling by 'conversion' to finalize its coercion to the specified 'unit'. Two common examples of this include (a) data stored in an unsigned type that requires a shift after scaling to re-center the data, and (b) specialized recording devices that naturally cause a scalar offset with respect to the true units. • **Data Type:** float32 • **Required:** False • **Default Value:** 0.0 • **Name:** offset
——resolution	Attribute	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 -1.0. • **Data Type:** float32 • **Required:** False • **Default Value:** -1.0 • **Name:** resolution
——continuity	Attribute	Optionally describe the continuity of the data. Can be "continuous", "instantaneous", or "step". For example, a voltage trace would be "continuous", because samples are recorded from a continuous process. An array of lick times would be "instantaneous", because the data represents distinct moments in time. Times of image presentations would be "step" because the picture remains the same until the next timepoint. This field is optional, but is useful in providing information about the underlying data. It may inform the way this data is interpreted, the way it is visualized, and what analysis methods are applicable. • **Data Type:** text • **Required:** False • **Name:** continuity
—metric	Dataset	The metric used, e.g. phase, amplitude, power. • **Data Type:** text • **Name:** metric
—source_channels	Dataset	DynamicTableRegion pointer to the channels that this decomposition series was generated from. • **Extends:** DynamicTableRegion • **Quantity:** 0 or 1 • **Name:** source_channels
—starting_time	Dataset	Timestamp of the first sample in seconds. When timestamps are uniformly spaced, the timestamp of the first sample can be specified and all subsequent ones calculated from the sampling rate attribute. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Name:** starting_time

continues on next page

Table 4.49 – continued from previous page

Id	Type	Description
——rate	Attribute	Sampling rate, in Hz. • **Data Type:** float32 • **Name:** rate
——unit	Attribute	Unit of measurement for time, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—timestamps	Dataset	Timestamps for samples stored in data, in seconds, relative to the common experiment master-clock stored in NWBFile.timestamps_reference_time. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** timestamps
——interval	Attribute	Value is '1' • **Data Type:** int32 • **Value:** 1 • **Name:** interval
——unit	Attribute	Unit of measurement for timestamps, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—control	Dataset	Numerical labels that apply to each time point in data for the purpose of querying and slicing data by these values. If present, the length of this array should be the same size as the first dimension of data. • **Quantity:** 0 or 1 • **Data Type:** uint8 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** control
—control_description	Dataset	Description of each control value. Must be present if control is present. If present, control_description[0] should describe time points where control == 0. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** ['num_control_values'] • **Shape:** [None] • **Name:** control_description

continues on next page

Table 4.49 – continued from previous page

Id	Type	Description
—source_timeseries	Link	Link to TimeSeries object that this data was calculated from. Metadata about electrodes and their position can be read from that ElectricalSeries so it is not necessary to store that information here. • **Target Type** *TimeSeries* • **Name:** source_timeseries

Table 4.50: Groups contained in <DecompositionSeries>

Id	Type	Description
<DecompositionSeries>	Group	Top level Group for <DecompositionSeries> • **Neurodata Type:** *DecompositionSeries* • **Extends:** *TimeSeries*
—source_timeseries	Link	Link to TimeSeries object that this data was calculated from. Metadata about electrodes and their position can be read from that ElectricalSeries so it is not necessary to store that information here. • **Target Type** *TimeSeries* • **Name:** source_timeseries
—bands	Group	Table for describing the bands that this series was generated from. There should be one row in this table for each band. • **Extends:** DynamicTable • **Name:** bands
—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. • **Quantity:** 0 or 1 • **Name:** sync

4.6.4.1 Groups: bands

Table for describing the bands that this series was generated from. There should be one row in this table for each band.

- **Extends:** DynamicTable
- **Name:** bands

Table 4.51: Datasets, Links, and Attributes contained in bands

Id	Type	Description
bands	Group	Top level Group for bands • **Extends:** DynamicTable • **Name:** bands
—band_name	Dataset	Name of the band, e.g. theta. • **Extends:** VectorData • **Data Type:** text • **Name:** band_name
—band_limits	Dataset	Low and high limit of each band in Hz. If it is a Gaussian filter, use 2 SD on either side of the center. • **Extends:** VectorData • **Data Type:** float32 • **Dimensions:** ['num_bands', 'low, high'] • **Shape:** [None, 2] • **Name:** band_limits
—band_mean	Dataset	The mean Gaussian filters, in Hz. • **Extends:** VectorData • **Data Type:** float32 • **Dimensions:** ['num_bands'] • **Shape:** [None] • **Name:** band_mean
—band_stdev	Dataset	The standard deviation of Gaussian filters, in Hz. • **Extends:** VectorData • **Data Type:** float32 • **Dimensions:** ['num_bands'] • **Shape:** [None] • **Name:** band_stdev

4.6.4.2 Groups: sync

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.

- **Quantity:** 0 or 1
- **Name:** sync

4.6.5 Units

Overview: Data about spiking units. Event times of observed units (e.g. cell, synapse, etc.) should be concatenated and stored in spike_times.

`Units` extends `DynamicTable` and includes all elements of DynamicTable with the following additions or changes.

- **Extends:** DynamicTable
- **Primitive Type:** Group
- **Default Name:** Units
- **Inherits from:** DynamicTable, Container
- **Source filename:** nwb.misc.yaml
- **Source Specification:** see Section 5.7.5

Table 4.52: Datasets, Links, and Attributes contained in <Units>

Id	Type	Description
<Units>	Group	Top level Group for <Units> • **Neurodata Type:** *Units* • **Extends:** DynamicTable • **Default Name:** Units
—colnames	Attribute	The names of the columns in this table. This should be used to specify an order to the columns. • **Data Type:** text • **Dimensions:** [‘num_columns’] • **Shape:** [None] • **Name:** colnames
—description	Attribute	Description of what is in this dynamic table. • **Data Type:** text • **Name:** description
—spike_times_index	Dataset	Index into the spike_times dataset. • **Extends:** VectorIndex • **Quantity:** 0 or 1 • **Name:** spike_times_index
—spike_times	Dataset	Spike times for each unit in seconds. • **Extends:** VectorData • **Quantity:** 0 or 1 • **Data Type:** float64 • **Name:** spike_times

continues on next page

Table 4.52 – continued from previous page

Id	Type	Description
——resolution	Attribute	The smallest possible difference between two spike times. Usually 1 divided by the acquisition sampling rate from which spike times were extracted, but could be larger if the acquisition time series was downsampled or smaller if the acquisition time series was smoothed/interpolated and it is possible for the spike time to be between samples. • **Data Type:** float64 • **Required:** False • **Name:** resolution
—obs_intervals_index	Dataset	Index into the obs_intervals dataset. • **Extends:** VectorIndex • **Quantity:** 0 or 1 • **Name:** obs_intervals_index
—obs_intervals	Dataset	Observation intervals for each unit. • **Extends:** VectorData • **Quantity:** 0 or 1 • **Data Type:** float64 • **Dimensions:** [‘num_intervals’, ‘start|end’] • **Shape:** [None, 2] • **Name:** obs_intervals
—electrodes_index	Dataset	Index into electrodes. • **Extends:** VectorIndex • **Quantity:** 0 or 1 • **Name:** electrodes_index
—electrodes	Dataset	Electrode that each spike unit came from, specified using a DynamicTableRegion. • **Extends:** DynamicTableRegion • **Quantity:** 0 or 1 • **Name:** electrodes
—electrode_group	Dataset	Electrode group that each spike unit came from. • **Extends:** VectorData • **Quantity:** 0 or 1 • **Data Type:** object reference to *ElectrodeGroup* • **Name:** electrode_group
—waveform_mean	Dataset	Spike waveform mean for each spike unit. • **Extends:** VectorData • **Quantity:** 0 or 1 • **Data Type:** float32 • **Dimensions:** [[‘num_units’, ‘num_samples’], [‘num_units’, ‘num_samples’, ‘num_electrodes’]] • **Shape:** [[None, None], [None, None, None]] • **Name:** waveform_mean

continues on next page

Table 4.52 – continued from previous page

Id	Type	Description
——sampling_rate	Attribute	Sampling rate, in hertz. • **Data Type:** float32 • **Required:** False • **Name:** sampling_rate
——unit	Attribute	Unit of measurement. This value is fixed to 'volts'. • **Data Type:** text • **Value:** volts • **Name:** unit
—waveform_sd	Dataset	Spike waveform standard deviation for each spike unit. • **Extends:** VectorData • **Quantity:** 0 or 1 • **Data Type:** float32 • **Dimensions:** [['num_units', 'num_samples'], ['num_units', 'num_samples', 'num_electrodes']] • **Shape:** [[None, None], [None, None, None]] • **Name:** waveform_sd
——sampling_rate	Attribute	Sampling rate, in hertz. • **Data Type:** float32 • **Required:** False • **Name:** sampling_rate
——unit	Attribute	Unit of measurement. This value is fixed to 'volts'. • **Data Type:** text • **Value:** volts • **Name:** unit

continues on next page

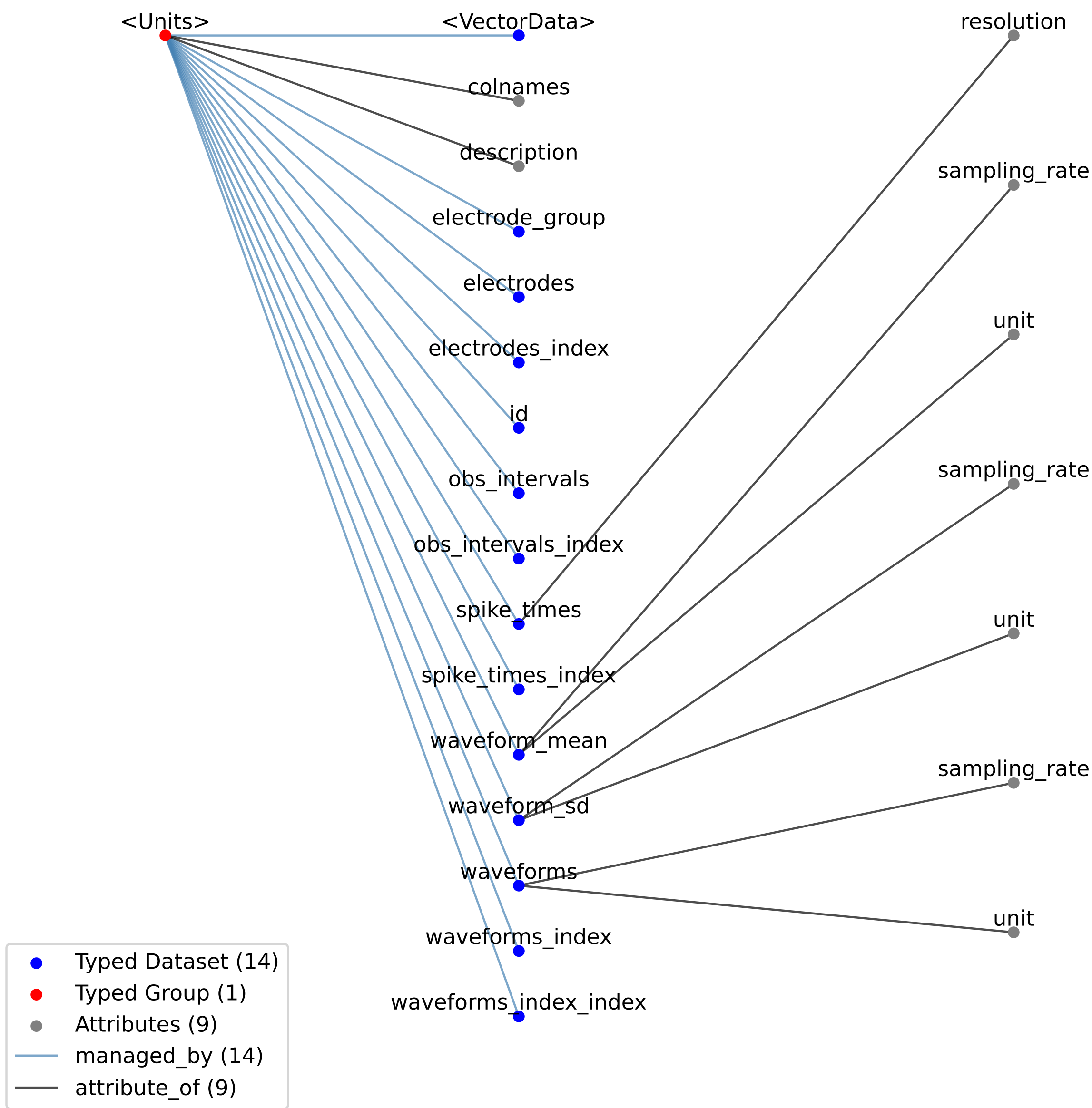
Table 4.52 – continued from previous page

Id	Type	Description
—waveforms	Dataset	Individual waveforms for each spike on each electrode. This is a doubly indexed column. The 'waveforms_index' column indexes which waveforms in this column belong to the same spike event for a given unit, where each waveform was recorded from a different electrode. The 'waveforms_index_index' column indexes the 'waveforms_index' column to indicate which spike events belong to a given unit. For example, if the 'waveforms_index_index' column has values [2, 5, 6], then the first 2 elements of the 'waveforms_index' column correspond to the 2 spike events of the first unit, the next 3 elements of the 'waveforms_index' column correspond to the 3 spike events of the second unit, and the next 1 element of the 'waveforms_index' column corresponds to the 1 spike event of the third unit. If the 'waveforms_index' column has values [3, 6, 8, 10, 12, 13], then the first 3 elements of the 'waveforms' column contain the 3 spike waveforms that were recorded from 3 different electrodes for the first spike time of the first unit. See https://nwb-schema.readthedocs.io/en/stable/format_description.html#doubly-ragged-arrays for a graphical representation of this example. When there is only one electrode for each unit (i.e., each spike time is associated with a single waveform), then the 'waveforms_index' column will have values 1, 2, ..., N, where N is the number of spike events. The number of electrodes for each spike event should be the same within a given unit. The 'electrodes' column should be used to indicate which electrodes are associated with each unit, and the order of the waveforms within a given unit x spike event should be in the same order as the electrodes referenced in the 'electrodes' column of this table. The number of samples for each waveform must be the same. • **Extends:** VectorData • **Quantity:** 0 or 1 • **Data Type:** numeric • **Dimensions:** ['num_waveforms', 'num_samples'] • **Shape:** [None, None] • **Name:** waveforms
——sampling_rate	Attribute	Sampling rate, in hertz. • **Data Type:** float32 • **Required:** False • **Name:** sampling_rate
——unit	Attribute	Unit of measurement. This value is fixed to 'volts'. • **Data Type:** text • **Value:** volts • **Name:** unit
—waveforms_index	Dataset	Index into the waveforms dataset. One value for every spike event. See 'waveforms' for more detail. • **Extends:** VectorIndex • **Quantity:** 0 or 1 • **Name:** waveforms_index

continues on next page

Table 4.52 – continued from previous page

Id	Type	Description
—waveforms_index_index	Dataset	Index into the waveforms_index dataset. One value for every unit (row in the table). See 'waveforms' for more detail. • **Extends:** VectorIndex • **Quantity:** 0 or 1 • **Name:** waveforms_index_index
—id	Dataset	Array of unique identifiers for the rows of this dynamic table. • **Extends:** ElementIdentifiers • **Data Type:** int • **Dimensions:** ['num_rows'] • **Shape:** [None] • **Name:** id
—<VectorData>	Dataset	Vector columns, including index columns, of this dynamic table. • **Extends:** VectorData • **Quantity:** 0 or more



<Units>
<VectorData>
colnames
description
electrode_group
electrodes
electrodes_index
id
obs_intervals
obs_intervals_index
spike_times
spike_times_index
waveform_mean
waveform_sd
waveforms
waveforms_index
waveforms_index_index
resolution
sampling_rate
unit
sampling_rate
unit
sampling_rate
unit
Typed Dataset (14)
Typed Group (1)
Attributes (9)
managed_by (14)
attribute_of (9)

4.7 Behavior

This source module contains neurodata_types for behavior data.

4.7.1 SpatialSeries

Overview: 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` extends `TimeSeries` and includes all elements of *TimeSeries* with the following additions or changes.

- **Extends:** *TimeSeries*
- **Primitive Type:** Group
- **Inherits from:** *TimeSeries*, *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.behavior.yaml
- **Source Specification:** see Section 5.8.1

Table 4.53: Datasets, Links, and Attributes contained in <SpatialSeries>

Id	Type	Description
<SpatialSeries>	Group	Top level Group for <SpatialSeries> • **Neurodata Type:** *SpatialSeries* • **Extends:** *TimeSeries*
—description	Attribute	Description of the time series. • **Data Type:** text • **Required:** False • **Default Value:** no description • **Name:** description
—comments	Attribute	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. • **Data Type:** text • **Required:** False • **Default Value:** no comments • **Name:** comments

continues on next page

Table 4.53 – continued from previous page

Id	Type	Description
—data	Dataset	1-D or 2-D array storing position or direction relative to some reference frame. • **Data Type:** numeric • **Dimensions:** [['num_times'], ['num_times', 'x'], ['num_times', 'x,y'], ['num_times', 'x,y,z']] • **Shape:** [[None], [None, 1], [None, 2], [None, 3]] • **Name:** data
——unit	Attribute	Base unit of measurement for working with the data. The default value is 'meters'. Actual stored values are not necessarily stored in these units. To access the data in these units, multiply 'data' by 'conversion' and add 'offset'. • **Data Type:** text • **Required:** False • **Default Value:** meters • **Name:** unit
——conversion	Attribute	Scalar to multiply each element in data to convert it to the specified 'unit'. If the data are stored in acquisition system units or other units that require a conversion to be interpretable, multiply the data by 'conversion' to convert the data to the specified 'unit'. e.g. if the data acquisition system stores values in this object as signed 16-bit integers (int16 range -32,768 to 32,767) that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition values to recorded volts is 2.5/32768/8000 = 9.5367e-9. • **Data Type:** float32 • **Required:** False • **Default Value:** 1.0 • **Name:** conversion
——offset	Attribute	Scalar to add to the data after scaling by 'conversion' to finalize its coercion to the specified 'unit'. Two common examples of this include (a) data stored in an unsigned type that requires a shift after scaling to re-center the data, and (b) specialized recording devices that naturally cause a scalar offset with respect to the true units. • **Data Type:** float32 • **Required:** False • **Default Value:** 0.0 • **Name:** offset
——resolution	Attribute	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 -1.0. • **Data Type:** float32 • **Required:** False • **Default Value:** -1.0 • **Name:** resolution

continues on next page

Table 4.53 – continued from previous page

Id	Type	Description
——continuity	Attribute	Optionally describe the continuity of the data. Can be "continuous", "instantaneous", or "step". For example, a voltage trace would be "continuous", because samples are recorded from a continuous process. An array of lick times would be "instantaneous", because the data represents distinct moments in time. Times of image presentations would be "step" because the picture remains the same until the next timepoint. This field is optional, but is useful in providing information about the underlying data. It may inform the way this data is interpreted, the way it is visualized, and what analysis methods are applicable. • **Data Type:** text • **Required:** False • **Name:** continuity
—reference_frame	Dataset	Description defining what exactly 'straight-ahead' means. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** reference_frame
—starting_time	Dataset	Timestamp of the first sample in seconds. When timestamps are uniformly spaced, the timestamp of the first sample can be specified and all subsequent ones calculated from the sampling rate attribute. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Name:** starting_time
——rate	Attribute	Sampling rate, in Hz. • **Data Type:** float32 • **Name:** rate
——unit	Attribute	Unit of measurement for time, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—timestamps	Dataset	Timestamps for samples stored in data, in seconds, relative to the common experiment master-clock stored in NWBFile.timestamps_reference_time. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** timestamps
——interval	Attribute	Value is '1' • **Data Type:** int32 • **Value:** 1 • **Name:** interval

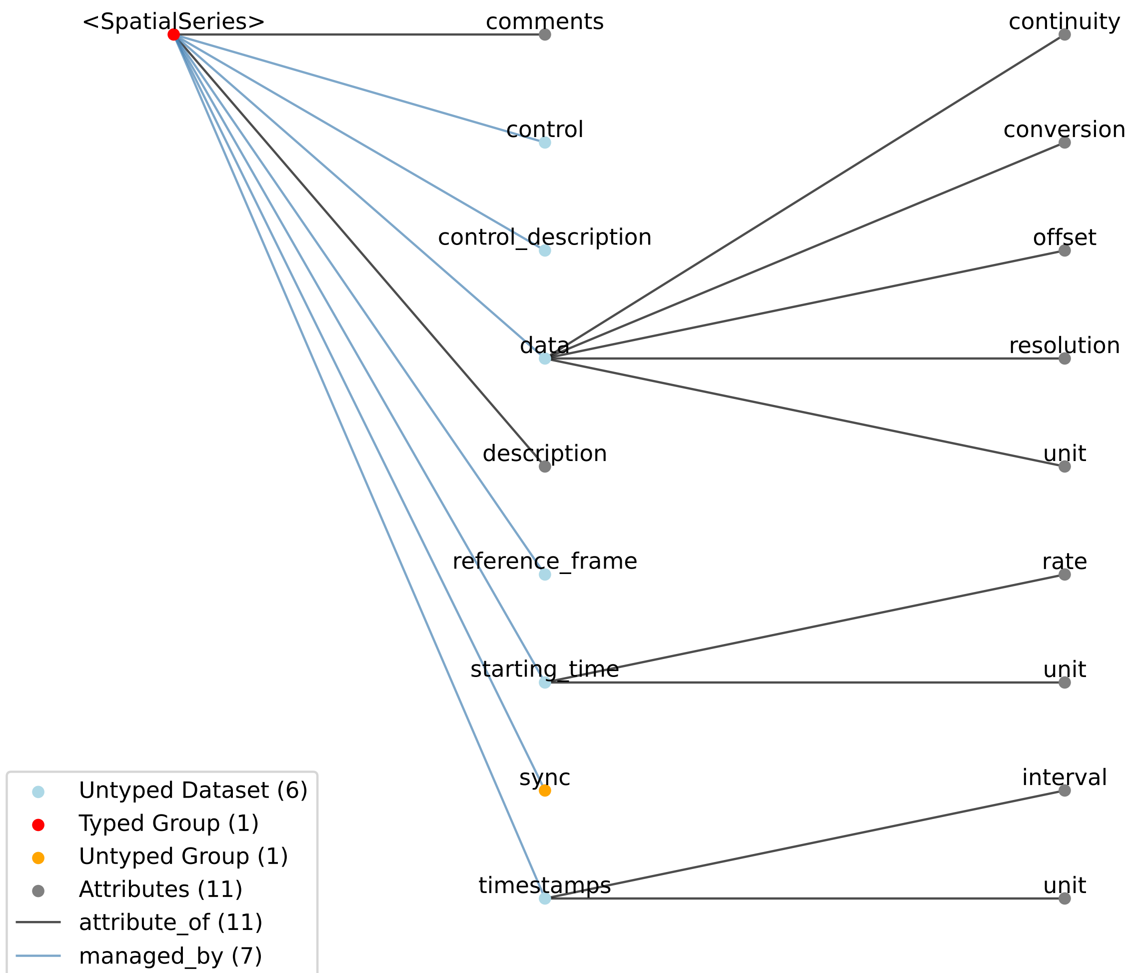
continues on next page

Table 4.53 – continued from previous page

Id	Type	Description
——unit	Attribute	Unit of measurement for timestamps, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—control	Dataset	Numerical labels that apply to each time point in data for the purpose of querying and slicing data by these values. If present, the length of this array should be the same size as the first dimension of data. • **Quantity:** 0 or 1 • **Data Type:** uint8 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** control
—control_description	Dataset	Description of each control value. Must be present if control is present. If present, control_description[0] should describe time points where control == 0. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** ['num_control_values'] • **Shape:** [None] • **Name:** control_description

Table 4.54: Groups contained in <SpatialSeries>

Id	Type	Description
<SpatialSeries>	Group	Top level Group for <SpatialSeries> • **Neurodata Type:** *SpatialSeries* • **Extends:** *TimeSeries*
—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. • **Quantity:** 0 or 1 • **Name:** sync


<SpatialSeries>
comments
control
control_description
data
description
reference_frame
starting_time
sync
timestamps
continuity
conversion
offset
resolution
unit
rate
unit
interval
unit
Untyped Dataset (6)
Typed Group (1)
Untyped Group (1)
Attributes (11)
attribute_of (11)
managed_by (7)

4.7.1.1 Groups: sync

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.

- **Quantity:** 0 or 1
- **Name:** sync

4.7.2 BehavioralEpochs

Overview: TimeSeries for storing behavioral 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` extends `NWBDataInterface` and includes all elements of *NWBDataInterface* with the following additions or changes.

- **Extends:** *NWBDataInterface*
- **Primitive Type:** Group
- **Default Name:** BehavioralEpochs
- **Inherits from:** *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.behavior.yaml
- **Source Specification:** see Section 5.8.2

Table 4.55: Groups contained in <BehavioralEpochs>

Id	Type	Description
<BehavioralEpochs>	Group	Top level Group for <BehavioralEpochs> • **Neurodata Type:** *BehavioralEpochs* • **Extends:** *NWBDataInterface* • **Default Name:** BehavioralEpochs
—<*IntervalSeries*>	Group	IntervalSeries object containing start and stop times of epochs. • **Extends:** *IntervalSeries* • **Quantity:** 0 or more

4.7.2.1 Groups: <IntervalSeries>

IntervalSeries object containing start and stop times of epochs.

- **Extends:** *IntervalSeries*
- **Quantity:** 0 or more

4.7.3 BehavioralEvents

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

`BehavioralEvents` extends `NWBDataInterface` and includes all elements of *NWBDataInterface* with the following additions or changes.

- **Extends:** *NWBDataInterface*
- **Primitive Type:** Group
- **Default Name:** BehavioralEvents
- **Inherits from:** *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.behavior.yaml
- **Source Specification:** see Section 5.8.3

Table 4.56: Groups contained in <BehavioralEvents>

Id	Type	Description
<BehavioralEvents>	Group	Top level Group for <BehavioralEvents> • **Neurodata Type:** *BehavioralEvents* • **Extends:** *NWBDataInterface* • **Default Name:** BehavioralEvents
—<*TimeSeries*>	Group	TimeSeries object containing behavioral events. • **Extends:** *TimeSeries* • **Quantity:** 0 or more

4.7.3.1 Groups: <TimeSeries>

TimeSeries object containing behavioral events.

- **Extends:** *TimeSeries*
- **Quantity:** 0 or more

4.7.4 BehavioralTimeSeries

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

`BehavioralTimeSeries` extends `NWBDataInterface` and includes all elements of *NWBDataInterface* with the following additions or changes.

- **Extends:** *NWBDataInterface*
- **Primitive Type:** Group
- **Default Name:** BehavioralTimeSeries
- **Inherits from:** *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.behavior.yaml
- **Source Specification:** see Section 5.8.4

Table 4.57: Groups contained in <BehavioralTimeSeries>

Id	Type	Description
<BehavioralTimeSeries>	Group	Top level Group for <BehavioralTimeSeries> • **Neurodata Type:** *BehavioralTimeSeries* • **Extends:** *NWBDataInterface* • **Default Name:** BehavioralTimeSeries
—<*TimeSeries*>	Group	TimeSeries object containing continuous behavioral data. • **Extends:** *TimeSeries* • **Quantity:** 0 or more

4.7.4.1 Groups: <TimeSeries>

TimeSeries object containing continuous behavioral data.

- **Extends:** *TimeSeries*
- **Quantity:** 0 or more

4.7.5 PupilTracking

Overview: Eye-tracking data, representing pupil size.

`PupilTracking` extends `NWBDataInterface` and includes all elements of *NWBDataInterface* with the following additions or changes.

- **Extends:** *NWBDataInterface*
- **Primitive Type:** Group
- **Default Name:** PupilTracking
- **Inherits from:** *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.behavior.yaml
- **Source Specification:** see Section 5.8.5

Table 4.58: Groups contained in <PupilTracking>

Id	Type	Description
<PupilTracking>	Group	Top level Group for <PupilTracking> • **Neurodata Type:** *PupilTracking* • **Extends:** *NWBDataInterface* • **Default Name:** PupilTracking
—<*TimeSeries*>	Group	TimeSeries object containing time series data on pupil size. • **Extends:** *TimeSeries* • **Quantity:** 1 or more

4.7.5.1 Groups: <TimeSeries>

TimeSeries object containing time series data on pupil size.

- **Extends:** *TimeSeries*
- **Quantity:** 1 or more

4.7.6 EyeTracking

Overview: Eye-tracking data, representing direction of gaze.

`EyeTracking` extends `NWBDataInterface` and includes all elements of *NWBDataInterface* with the following additions or changes.

- **Extends:** *NWBDataInterface*
- **Primitive Type:** Group
- **Default Name:** EyeTracking
- **Inherits from:** *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.behavior.yaml
- **Source Specification:** see Section 5.8.6

Table 4.59: Groups contained in <EyeTracking>

Id	Type	Description
<EyeTracking>	Group	Top level Group for <EyeTracking> • **Neurodata Type:** *EyeTracking* • **Extends:** *NWBDataInterface* • **Default Name:** EyeTracking
—<*SpatialSeries*>	Group	SpatialSeries object containing data measuring direction of gaze. • **Extends:** *SpatialSeries* • **Quantity:** 0 or more

4.7.6.1 Groups: <SpatialSeries>

SpatialSeries object containing data measuring direction of gaze.

- **Extends:** *SpatialSeries*
- **Quantity:** 0 or more

4.7.7 CompassDirection

Overview: 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` extends `NWBDataInterface` and includes all elements of *NWBDataInterface* with the following additions or changes.

- **Extends:** *NWBDataInterface*
- **Primitive Type:** Group
- **Default Name:** CompassDirection
- **Inherits from:** *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.behavior.yaml
- **Source Specification:** see Section 5.8.7

Table 4.60: Groups contained in <CompassDirection>

Id	Type	Description
<CompassDirection>	Group	Top level Group for <CompassDirection> • **Neurodata Type:** *CompassDirection* • **Extends:** *NWBDataInterface* • **Default Name:** CompassDirection
—<*SpatialSeries*>	Group	SpatialSeries object containing direction of gaze travel. • **Extends:** *SpatialSeries* • **Quantity:** 0 or more

4.7.7.1 Groups: <SpatialSeries>

SpatialSeries object containing direction of gaze travel.

- **Extends:** *SpatialSeries*
- **Quantity:** 0 or more

4.7.8 Position

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

`Position` extends `NWBDataInterface` and includes all elements of *NWBDataInterface* with the following additions or changes.

- **Extends:** *NWBDataInterface*
- **Primitive Type:** Group
- **Default Name:** Position
- **Inherits from:** *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.behavior.yaml
- **Source Specification:** see Section 5.8.8

Table 4.61: Groups contained in <Position>

Id	Type	Description
<Position>	Group	Top level Group for <Position> • **Neurodata Type:** *Position* • **Extends:** *NWBDataInterface* • **Default Name:** Position
—<*SpatialSeries*>	Group	SpatialSeries object containing position data. • **Extends:** *SpatialSeries* • **Quantity:** 1 or more

4.7.8.1 Groups: <SpatialSeries>

SpatialSeries object containing position data.

- **Extends:** *SpatialSeries*
- **Quantity:** 1 or more

4.8 Extracellular electrophysiology

This source module contains neurodata_types for extracellular electrophysiology data.

4.8.1 ElectricalSeries

Overview: A time series of acquired voltage data from extracellular recordings. The data field is an int or float array storing data in volts. The first dimension should always represent time. The second dimension, if present, should represent channels.

`ElectricalSeries` extends `TimeSeries` and includes all elements of *TimeSeries* with the following additions or changes.

- **Extends:** *TimeSeries*
- **Primitive Type:** Group
- **Inherits from:** *TimeSeries*, *NWBDataInterface*, *NWBContainer*, Container
- **Subtypes:** *SpikeEventSeries*
- **Source filename:** nwb.ecephys.yaml
- **Source Specification:** see Section 5.9.1

Table 4.62: Datasets, Links, and Attributes contained in <ElectricalSeries>

Id	Type	Description
<ElectricalSeries>	Group	Top level Group for <ElectricalSeries> • **Neurodata Type:** *ElectricalSeries* • **Extends:** *TimeSeries*
—filtering	Attribute	Filtering applied to all channels of the data. For example, if this ElectricalSeries represents high-pass-filtered data (also known as AP Band), then this value could be “High-pass 4-pole Bessel filter at 500 Hz”. If this ElectricalSeries represents low-pass-filtered LFP data and the type of filter is unknown, then this value could be “Low-pass filter at 300 Hz”. If a non-standard filter type is used, provide as much detail about the filter properties as possible. • **Data Type:** text • **Required:** False • **Name:** filtering
—description	Attribute	Description of the time series. • **Data Type:** text • **Required:** False • **Default Value:** no description • **Name:** description

continues on next page

Table 4.62 – continued from previous page

Id	Type	Description
—comments	Attribute	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. • **Data Type:** text • **Required:** False • **Default Value:** no comments • **Name:** comments
—data	Dataset	Recorded voltage data. • **Data Type:** numeric • **Dimensions:** [['num_times'], ['num_times', 'num_channels'], ['num_times', 'num_channels', 'num_samples']] • **Shape:** [[None], [None, None], [None, None, None]] • **Name:** data
——unit	Attribute	Base unit of measurement for working with the data. This value is fixed to 'volts'. Actual stored values are not necessarily stored in these units. To access the data in these units, multiply 'data' by 'conversion', followed by 'channel_conversion' (if present), and then add 'offset'. • **Data Type:** text • **Value:** volts • **Name:** unit
——conversion	Attribute	Scalar to multiply each element in data to convert it to the specified 'unit'. If the data are stored in acquisition system units or other units that require a conversion to be interpretable, multiply the data by 'conversion' to convert the data to the specified 'unit'. e.g. if the data acquisition system stores values in this object as signed 16-bit integers (int16 range -32,768 to 32,767) that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition values to recorded volts is 2.5/32768/8000 = 9.5367e-9. • **Data Type:** float32 • **Required:** False • **Default Value:** 1.0 • **Name:** conversion
——offset	Attribute	Scalar to add to the data after scaling by 'conversion' to finalize its coercion to the specified 'unit'. Two common examples of this include (a) data stored in an unsigned type that requires a shift after scaling to re-center the data, and (b) specialized recording devices that naturally cause a scalar offset with respect to the true units. • **Data Type:** float32 • **Required:** False • **Default Value:** 0.0 • **Name:** offset

continues on next page

Table 4.62 – continued from previous page

Id	Type	Description
——resolution	Attribute	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 -1.0. • **Data Type:** float32 • **Required:** False • **Default Value:** -1.0 • **Name:** resolution
——continuity	Attribute	Optionally describe the continuity of the data. Can be "continuous", "instantaneous", or "step". For example, a voltage trace would be "continuous", because samples are recorded from a continuous process. An array of lick times would be "instantaneous", because the data represents distinct moments in time. Times of image presentations would be "step" because the picture remains the same until the next timepoint. This field is optional, but is useful in providing information about the underlying data. It may inform the way this data is interpreted, the way it is visualized, and what analysis methods are applicable. • **Data Type:** text • **Required:** False • **Name:** continuity
—electrodes	Dataset	DynamicTableRegion pointer to the electrodes that this time series was generated from. • **Extends:** DynamicTableRegion • **Name:** electrodes
—channel_conversion	Dataset	Channel-specific conversion factor. Multiply the data in the 'data' dataset by these values along the channel axis (as indicated by axis attribute) AND by the global conversion factor in the 'conversion' attribute of 'data' to get the data values in Volts, i.e, data in Volts = data * data.conversion * channel_conversion. This approach allows for both global and per-channel data conversion factors needed to support the storage of electrical recordings as native values generated by data acquisition systems. If this dataset is not present, then there is no channel-specific conversion factor, i.e. it is 1 for all channels. • **Quantity:** 0 or 1 • **Data Type:** float32 • **Dimensions:** ['num_channels'] • **Shape:** [None] • **Name:** channel_conversion
——axis	Attribute	The zero-indexed axis of the 'data' dataset that the channel-specific conversion factor corresponds to. This value is fixed to 1. • **Data Type:** int32 • **Value:** 1 • **Name:** axis

continues on next page

Table 4.62 – continued from previous page

Id	Type	Description
—starting_time	Dataset	Timestamp of the first sample in seconds. When timestamps are uniformly spaced, the timestamp of the first sample can be specified and all subsequent ones calculated from the sampling rate attribute. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Name:** starting_time
——rate	Attribute	Sampling rate, in Hz. • **Data Type:** float32 • **Name:** rate
——unit	Attribute	Unit of measurement for time, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—timestamps	Dataset	Timestamps for samples stored in data, in seconds, relative to the common experiment master-clock stored in NWBFile.timestamps_reference_time. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** timestamps
——interval	Attribute	Value is '1' • **Data Type:** int32 • **Value:** 1 • **Name:** interval
——unit	Attribute	Unit of measurement for timestamps, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—control	Dataset	Numerical labels that apply to each time point in data for the purpose of querying and slicing data by these values. If present, the length of this array should be the same size as the first dimension of data. • **Quantity:** 0 or 1 • **Data Type:** uint8 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** control

continues on next page

Table 4.62 – continued from previous page

Id	Type	Description
—control_description	Dataset	Description of each control value. Must be present if control is present. If present, control_description[0] should describe time points where control == 0. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** ['num_control_values'] • **Shape:** [None] • **Name:** control_description

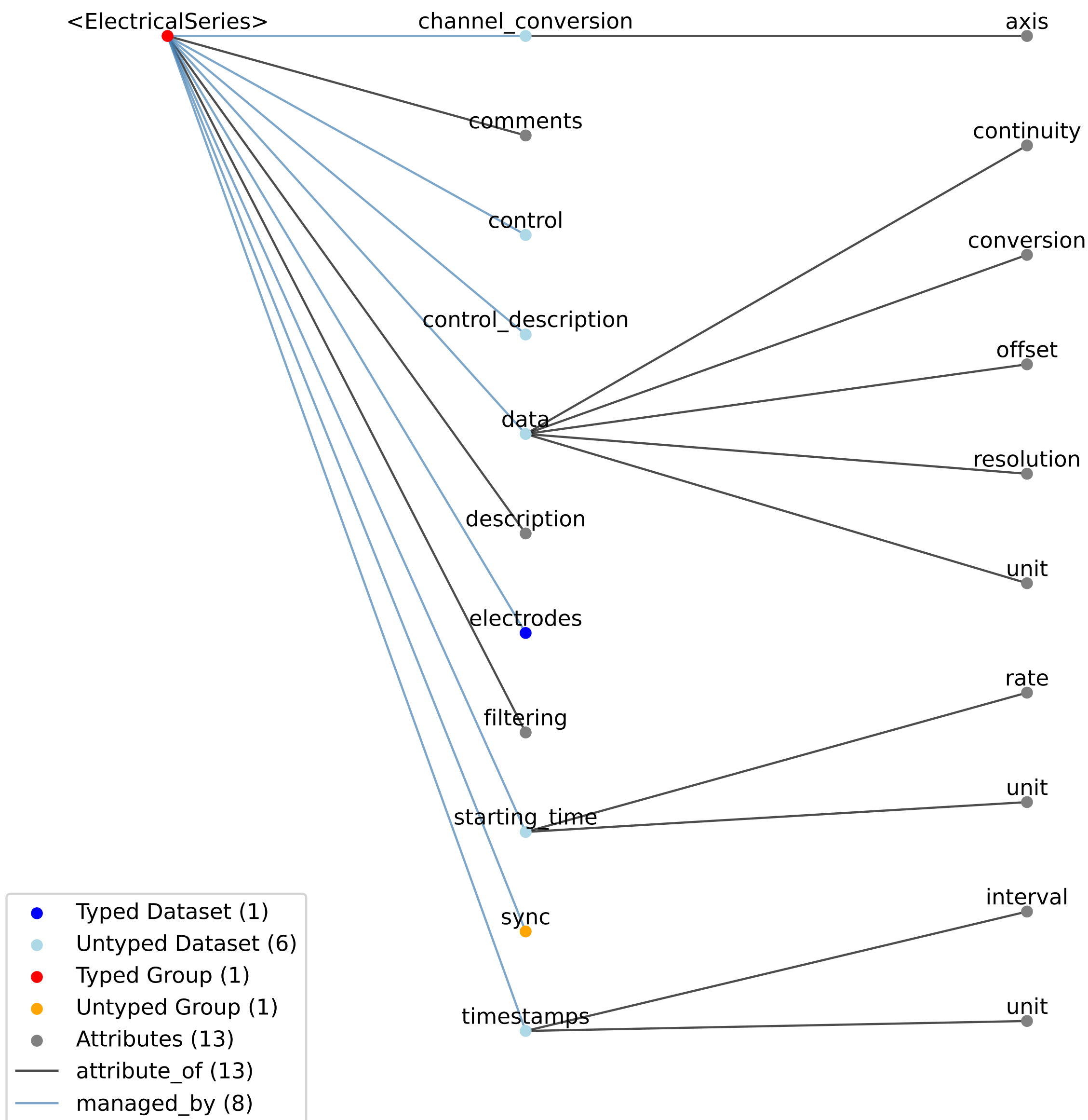
Table 4.63: Groups contained in <ElectricalSeries>

Id	Type	Description
<ElectricalSeries>	Group	Top level Group for <ElectricalSeries> • **Neurodata Type:** *ElectricalSeries* • **Extends:** *TimeSeries*
—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. • **Quantity:** 0 or 1 • **Name:** sync

4.8.1.1 Groups: sync

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.

- **Quantity:** 0 or 1
- **Name:** sync



<ElectricalSeries>
channel_conversion
axis
comments
continuity
control
conversion
control_description
offset
data
resolution
description
unit
electrodes
rate
filtering
unit
starting_time
interval
sync
timestamps
unit
Typed Dataset (1)
Untyped Dataset (6)
Typed Group (1)
Untyped Group (1)
Attributes (13)
attribute_of (13)
managed_by (8)

4.8.2 SpikeEventSeries

Overview: Stores snapshots/snippets of recorded spike events (i.e., threshold crossings). This may also be raw data, as reported by ephys hardware. If so, the TimeSeries::description field should describe 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` extends `ElectricalSeries` and includes all elements of *ElectricalSeries* with the following additions or changes.

- **Extends:** *ElectricalSeries*
- **Primitive Type:** Group
- **Inherits from:** *ElectricalSeries*, *TimeSeries*, *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.ecephys.yaml
- **Source Specification:** see Section 5.9.2

Table 4.64: Datasets, Links, and Attributes contained in <SpikeEventSeries>

Id	Type	Description
<SpikeEventSeries>	Group	Top level Group for <SpikeEventSeries> • **Neurodata Type:** *SpikeEventSeries* • **Extends:** *ElectricalSeries*
—filtering	Attribute	Filtering applied to all channels of the data. For example, if this ElectricalSeries represents high-pass-filtered data (also known as AP Band), then this value could be "High-pass 4-pole Bessel filter at 500 Hz". If this ElectricalSeries represents low-pass-filtered LFP data and the type of filter is unknown, then this value could be "Low-pass filter at 300 Hz". If a non-standard filter type is used, provide as much detail about the filter properties as possible. • **Data Type:** text • **Required:** False • **Name:** filtering
—description	Attribute	Description of the time series. • **Data Type:** text • **Required:** False • **Default Value:** no description • **Name:** description
—comments	Attribute	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. • **Data Type:** text • **Required:** False • **Default Value:** no comments • **Name:** comments

continues on next page

Table 4.64 – continued from previous page

Id	Type	Description
—data	Dataset	Spike waveforms. • **Data Type:** numeric • **Dimensions:** [['num_events', 'num_samples'], ['num_events', 'num_channels', 'num_samples']] • **Shape:** [[None, None], [None, None, None]] • **Name:** data
——unit	Attribute	Unit of measurement for waveforms, which is fixed to 'volts'. • **Data Type:** text • **Value:** volts • **Name:** unit
——conversion	Attribute	Scalar to multiply each element in data to convert it to the specified 'unit'. If the data are stored in acquisition system units or other units that require a conversion to be interpretable, multiply the data by 'conversion' to convert the data to the specified 'unit'. e.g. if the data acquisition system stores values in this object as signed 16-bit integers (int16 range -32,768 to 32,767) that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition values to recorded volts is 2.5/32768/8000 = 9.5367e-9. • **Data Type:** float32 • **Required:** False • **Default Value:** 1.0 • **Name:** conversion
——offset	Attribute	Scalar to add to the data after scaling by 'conversion' to finalize its coercion to the specified 'unit'. Two common examples of this include (a) data stored in an unsigned type that requires a shift after scaling to re-center the data, and (b) specialized recording devices that naturally cause a scalar offset with respect to the true units. • **Data Type:** float32 • **Required:** False • **Default Value:** 0.0 • **Name:** offset
——resolution	Attribute	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 -1.0. • **Data Type:** float32 • **Required:** False • **Default Value:** -1.0 • **Name:** resolution

continues on next page

Table 4.64 – continued from previous page

Id	Type	Description
——continuity	Attribute	Optionally describe the continuity of the data. Can be "continuous", "instantaneous", or "step". For example, a voltage trace would be "continuous", because samples are recorded from a continuous process. An array of lick times would be "instantaneous", because the data represents distinct moments in time. Times of image presentations would be "step" because the picture remains the same until the next timepoint. This field is optional, but is useful in providing information about the underlying data. It may inform the way this data is interpreted, the way it is visualized, and what analysis methods are applicable. • **Data Type:** text • **Required:** False • **Name:** continuity
—timestamps	Dataset	Timestamps for samples stored in data, in seconds, relative to the common experiment master-clock stored in NWBFile.timestamps_reference_time. Timestamps are required for the events. Unlike for TimeSeries, timestamps are required for SpikeEventSeries and are thus re-specified here. • **Data Type:** float64 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** timestamps
——interval	Attribute	Value is '1' • **Data Type:** int32 • **Value:** 1 • **Name:** interval
——unit	Attribute	Unit of measurement for timestamps, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—electrodes	Dataset	DynamicTableRegion pointer to the electrodes that this time series was generated from. • **Extends:** DynamicTableRegion • **Name:** electrodes

continues on next page

Table 4.64 – continued from previous page

Id	Type	Description
—channel_conversion	Dataset	Channel-specific conversion factor. Multiply the data in the 'data' dataset by these values along the channel axis (as indicated by axis attribute) AND by the global conversion factor in the 'conversion' attribute of 'data' to get the data values in Volts, i.e, data in Volts = data * data.conversion * channel_conversion. This approach allows for both global and per-channel data conversion factors needed to support the storage of electrical recordings as native values generated by data acquisition systems. If this dataset is not present, then there is no channel-specific conversion factor, i.e. it is 1 for all channels. • **Quantity:** 0 or 1 • **Data Type:** float32 • **Dimensions:** ['num_channels'] • **Shape:** [None] • **Name:** channel_conversion
——axis	Attribute	The zero-indexed axis of the 'data' dataset that the channel-specific conversion factor corresponds to. This value is fixed to 1. • **Data Type:** int32 • **Value:** 1 • **Name:** axis
—starting_time	Dataset	Timestamp of the first sample in seconds. When timestamps are uniformly spaced, the timestamp of the first sample can be specified and all subsequent ones calculated from the sampling rate attribute. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Name:** starting_time
——rate	Attribute	Sampling rate, in Hz. • **Data Type:** float32 • **Name:** rate
——unit	Attribute	Unit of measurement for time, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—control	Dataset	Numerical labels that apply to each time point in data for the purpose of querying and slicing data by these values. If present, the length of this array should be the same size as the first dimension of data. • **Quantity:** 0 or 1 • **Data Type:** uint8 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** control

continues on next page

Table 4.64 – continued from previous page

Id	Type	Description
—control_description	Dataset	Description of each control value. Must be present if control is present. If present, control_description[0] should describe time points where control == 0. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** ['num_control_values'] • **Shape:** [None] • **Name:** control_description

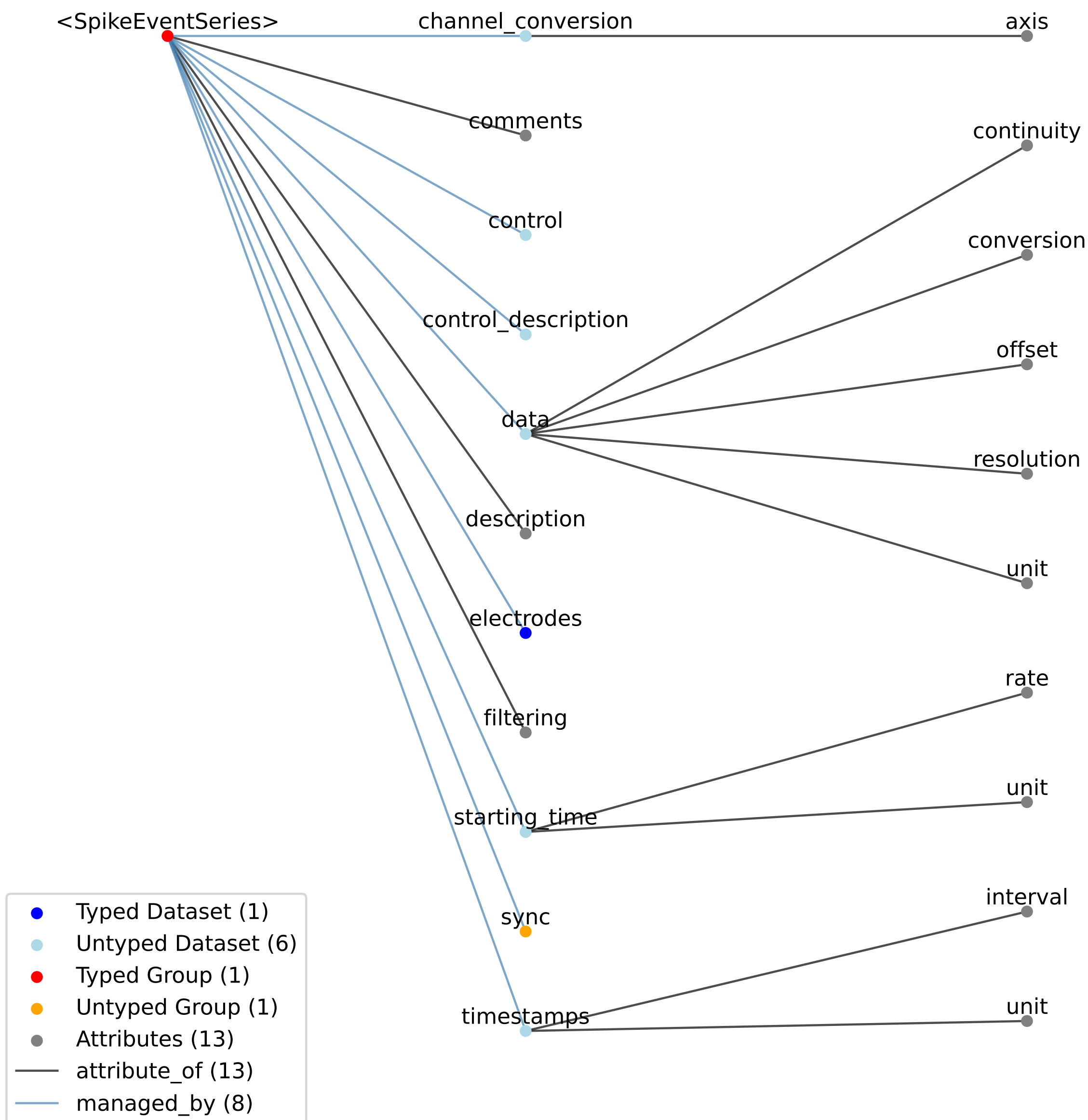
Table 4.65: Groups contained in <SpikeEventSeries>

Id	Type	Description
<SpikeEventSeries>	Group	Top level Group for <SpikeEventSeries> • **Neurodata Type:** *SpikeEventSeries* • **Extends:** *ElectricalSeries*
—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. • **Quantity:** 0 or 1 • **Name:** sync

4.8.2.1 Groups: sync

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.

- **Quantity:** 0 or 1
- **Name:** sync



<SpikeEventSeries>
channel_conversion
axis
comments
continuity
control
conversion
control_description
offset
data
resolution
description
unit
electrodes
rate
filtering
unit
starting_time
interval
sync
timestamps
unit
Typed Dataset (1)
Untyped Dataset (6)
Typed Group (1)
Untyped Group (1)
Attributes (13)
attribute_of (13)
managed_by (8)

4.8.3 FeatureExtraction

Overview: Features, such as PC1 and PC2, that are extracted from signals stored in a SpikeEventSeries or other source.

`FeatureExtraction` extends `NWBDataInterface` and includes all elements of *NWBDataInterface* with the following additions or changes.

- **Extends:** *NWBDataInterface*
- **Primitive Type:** Group
- **Default Name:** FeatureExtraction
- **Inherits from:** *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.ecephys.yaml
- **Source Specification:** see Section 5.9.3

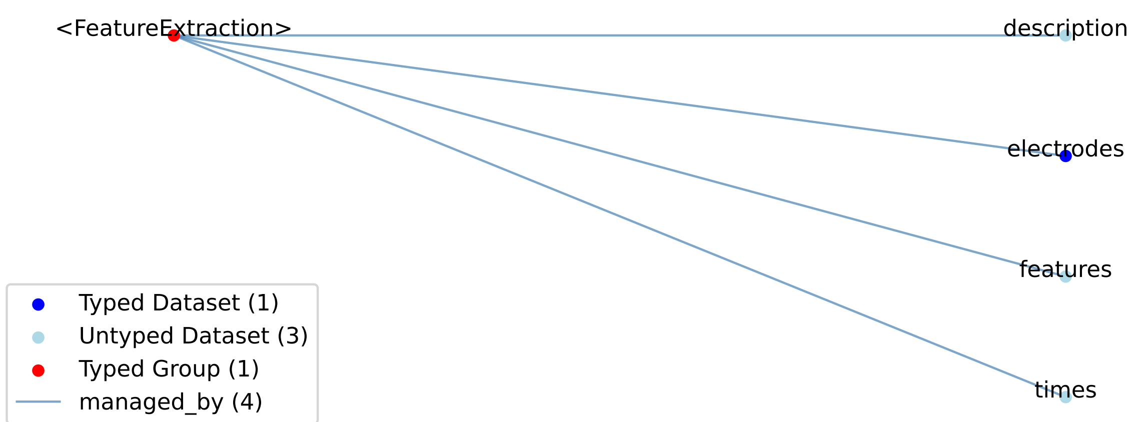


Table 4.66: Datasets, Links, and Attributes contained in <FeatureExtraction>

Id	Type	Description
<FeatureExtraction>	Group	Top level Group for <FeatureExtraction> • **Neurodata Type:** *FeatureExtraction* • **Extends:** *NWBDataInterface* • **Default Name:** FeatureExtraction
—description	Dataset	Description of features (eg, ''PC1'') for each of the extracted features. • **Data Type:** text • **Dimensions:** ['num_features'] • **Shape:** [None] • **Name:** description
—features	Dataset	Multi-dimensional array of features extracted from each event. • **Data Type:** float32 • **Dimensions:** ['num_events', 'num_channels', 'num_features'] • **Shape:** [None, None, None] • **Name:** features

continues on next page

Table 4.66 – continued from previous page

Id	Type	Description
—times	Dataset	Times of events that features correspond to (can be a link). • **Data Type:** float64 • **Dimensions:** ['num_events'] • **Shape:** [None] • **Name:** times
—electrodes	Dataset	DynamicTableRegion pointer to the electrodes that this time series was generated from. • **Extends:** DynamicTableRegion • **Name:** electrodes

4.8.4 EventDetection

Overview: Detected spike events from voltage trace(s).

`EventDetection` extends `NWBDataInterface` and includes all elements of *NWBDataInterface* with the following additions or changes.

- **Extends:** *NWBDataInterface*
- **Primitive Type:** Group
- **Default Name:** EventDetection
- **Inherits from:** *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.ecephys.yaml
- **Source Specification:** see Section 5.9.4

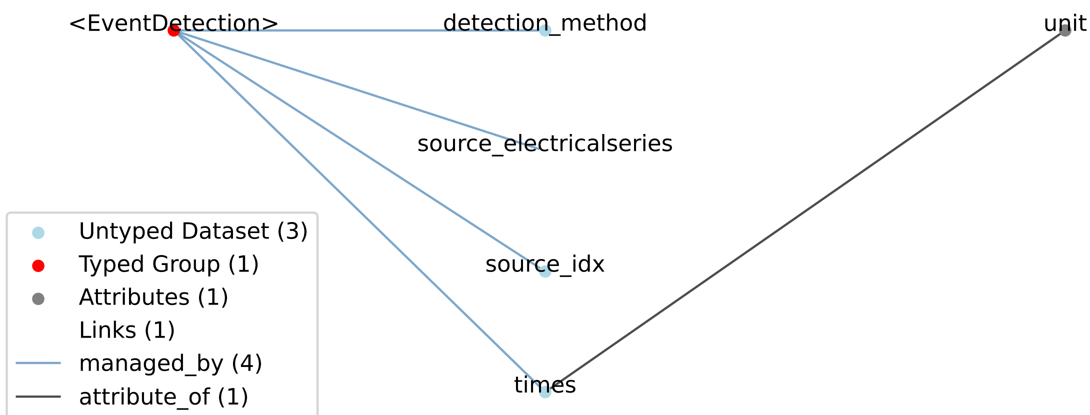


Table 4.67: Datasets, Links, and Attributes contained in <EventDetection>

Id	Type	Description
<EventDetection>	Group	Top level Group for <EventDetection> • **Neurodata Type:** *EventDetection* • **Extends:** *NWBDataInterface* • **Default Name:** EventDetection
—detection_method	Dataset	Description of how events were detected, such as voltage threshold, or dV/dT threshold, as well as relevant values. • **Data Type:** text • **Name:** detection_method

continues on next page

Table 4.67 – continued from previous page

Id	Type	Description
—source_idx	Dataset	Indices (zero-based) into source ElectricalSeries::data array corresponding to time of event. ‘’description” should define what is meant by time of event (e.g., .25 ms before action potential peak, zero-crossing time, etc). The index points to each event from the raw data. • **Data Type:** int32 • **Dimensions:** [‘num_events’] • **Shape:** [None] • **Name:** source_idx
—times	Dataset	Timestamps of events, in seconds. • **Data Type:** float64 • **Dimensions:** [‘num_events’] • **Shape:** [None] • **Name:** times
——unit	Attribute	Unit of measurement for event times, which is fixed to ‘seconds’. • **Data Type:** text • **Value:** seconds • **Name:** unit
—source_electricalseries	Link	Link to the 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 include that information here. • **Target Type** *ElectricalSeries* • **Name:** source_electricalseries

Table 4.68: Groups contained in <EventDetection>

Id	Type	Description
<EventDetection>	Group	Top level Group for <EventDetection> • **Neurodata Type:** *EventDetection* • **Extends:** *NWBDataInterface* • **Default Name:** EventDetection
—source_electricalseries	Link	Link to the 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 include that information here. • **Target Type** *ElectricalSeries* • **Name:** source_electricalseries

4.8.5 EventWaveform

Overview: 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` extends `NWBDataInterface` and includes all elements of *NWBDataInterface* with the following additions or changes.

- **Extends:** *NWBDataInterface*
- **Primitive Type:** Group
- **Default Name:** EventWaveform
- **Inherits from:** *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.ecephys.yaml
- **Source Specification:** see Section 5.9.5

Table 4.69: Groups contained in <EventWaveform>

Id	Type	Description
<EventWaveform>	Group	Top level Group for <EventWaveform> • **Neurodata Type:** *EventWaveform* • **Extends:** *NWBDataInterface* • **Default Name:** EventWaveform
—<*SpikeEventSeries*>	Group	SpikeEventSeries object(s) containing detected spike event waveforms. • **Extends:** *SpikeEventSeries* • **Quantity:** 0 or more

4.8.5.1 Groups: <SpikeEventSeries>

SpikeEventSeries object(s) containing detected spike event waveforms.

- **Extends:** *SpikeEventSeries*
- **Quantity:** 0 or more

4.8.6 FilteredEphys

Overview: Electrophysiology 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. Filter properties should be noted in the ElectricalSeries 'filtering' attribute.

`FilteredEphys` extends `NWBDataInterface` and includes all elements of *NWBDataInterface* with the following additions or changes.

- **Extends:** *NWBDataInterface*
- **Primitive Type:** Group
- **Default Name:** FilteredEphys
- **Inherits from:** *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.ecephys.yaml
- **Source Specification:** see Section 5.9.6

Table 4.70: Groups contained in <FilteredEphys>

Id	Type	Description
<FilteredEphys>	Group	Top level Group for <FilteredEphys> • **Neurodata Type:** *FilteredEphys* • **Extends:** *NWBDataInterface* • **Default Name:** FilteredEphys
—<*ElectricalSeries*>	Group	ElectricalSeries object(s) containing filtered electrophysiology data. • **Extends:** *ElectricalSeries* • **Quantity:** 1 or more

4.8.6.1 Groups: <ElectricalSeries>

ElectricalSeries object(s) containing filtered electrophysiology data.

- **Extends:** *ElectricalSeries*
- **Quantity:** 1 or more

4.8.7 LFP

Overview: 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 'filtering' attribute.

`LFP` extends `NWBDataInterface` and includes all elements of *NWBDataInterface* with the following additions or changes.

- **Extends:** *NWBDataInterface*
- **Primitive Type:** Group
- **Default Name:** LFP
- **Inherits from:** *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.ecephys.yaml
- **Source Specification:** see Section 5.9.7

Table 4.71: Groups contained in <LFP>

Id	Type	Description
<LFP>	Group	Top level Group for <LFP> • **Neurodata Type:** *LFP* • **Extends:** *NWBDataInterface* • **Default Name:** LFP
—<*ElectricalSeries*>	Group	ElectricalSeries object(s) containing LFP data for one or more channels. • **Extends:** *ElectricalSeries* • **Quantity:** 1 or more

4.8.7.1 Groups: <ElectricalSeries>

ElectricalSeries object(s) containing LFP data for one or more channels.

- **Extends:** *ElectricalSeries*
- **Quantity:** 1 or more

4.8.8 ElectrodeGroup

Overview: A physical grouping of electrodes, e.g. a shank of an array.

`ElectrodeGroup` extends `NWBContainer` and includes all elements of *NWBContainer* with the following additions or changes.

- **Extends:** *NWBContainer*
- **Primitive Type:** Group
- **Inherits from:** *NWBContainer*, Container
- **Source filename:** nwb.ecephys.yaml
- **Source Specification:** see Section 5.9.8

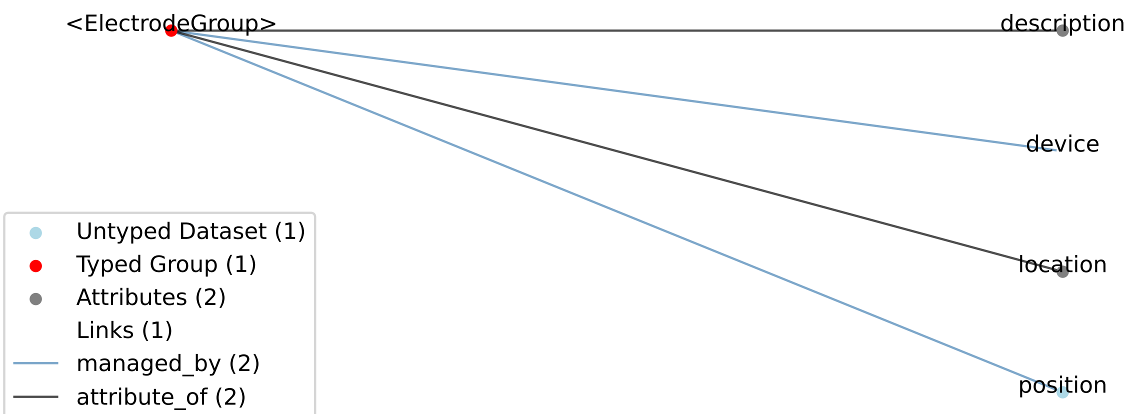


Table 4.72: Datasets, Links, and Attributes contained in <ElectrodeGroup>

Id	Type	Description
<ElectrodeGroup>	Group	Top level Group for <ElectrodeGroup> • **Neurodata Type:** *ElectrodeGroup* • **Extends:** *NWBContainer*
—description	Attribute	Description of this electrode group. • **Data Type:** text • **Name:** description
—location	Attribute	Location of electrode group. Specify the area, layer, comments on estimation of area/layer, etc. Use standard atlas names for anatomical regions when possible. • **Data Type:** text • **Name:** location

continues on next page

Table 4.72 – continued from previous page

Id	Type	Description
—position	Dataset	stereotaxic or common framework coordinates • **Quantity:** 0 or 1 • **Data Type: Compound data type with the following elements:** – **x:** x coordinate (*dtype*= float32) – **y:** y coordinate (*dtype*= float32) – **z:** z coordinate (*dtype*= float32) • **Name:** position
—device	Link	Link to the device that was used to record from this electrode group. • **Target Type** *Device* • **Name:** device

Table 4.73: Groups contained in <ElectrodeGroup>

Id	Type	Description
<ElectrodeGroup>	Group	Top level Group for <ElectrodeGroup> • **Neurodata Type:** *ElectrodeGroup* • **Extends:** *NWBContainer*
—device	Link	Link to the device that was used to record from this electrode group. • **Target Type** *Device* • **Name:** device

4.8.9 ClusterWaveforms

Overview: DEPRECATED 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` extends `NWBDataInterface` and includes all elements of *NWBDataInterface* with the following additions or changes.

- **Extends:** *NWBDataInterface*
- **Primitive Type:** Group
- **Default Name:** ClusterWaveforms
- **Inherits from:** *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.ecephys.yaml
- **Source Specification:** see Section 5.9.9

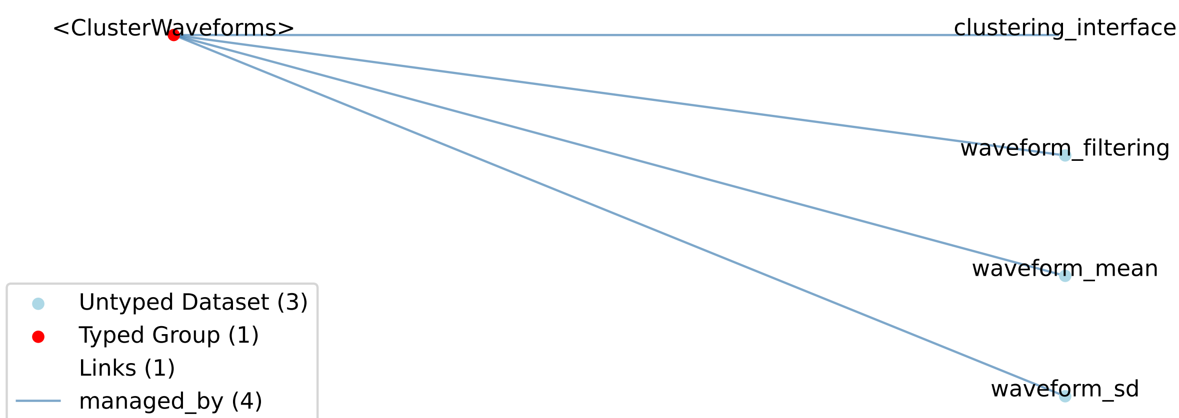


Table 4.74: Datasets, Links, and Attributes contained in <ClusterWaveforms>

Id	Type	Description
<ClusterWaveforms>	Group	Top level Group for <ClusterWaveforms> • **Neurodata Type:** *ClusterWaveforms* • **Extends:** *NWBDataInterface* • **Default Name:** ClusterWaveforms
—waveform_filtering	Dataset	Filtering applied to data before generating mean/sd • **Data Type:** text • **Name:** waveform_filtering

continues on next page

Table 4.74 – continued from previous page

Id	Type	Description
—waveform_mean	Dataset	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) • **Data Type:** float32 • **Dimensions:** ['num_clusters', 'num_samples'] • **Shape:** [None, None] • **Name:** waveform_mean
—waveform_sd	Dataset	Stdev of waveforms for each cluster, using the same indices as in mean • **Data Type:** float32 • **Dimensions:** ['num_clusters', 'num_samples'] • **Shape:** [None, None] • **Name:** waveform_sd
—clustering_interface	Link	Link to Clustering interface that was the source of the clustered data • **Target Type** *Clustering* • **Name:** clustering_interface

Table 4.75: Groups contained in <ClusterWaveforms>

Id	Type	Description
<ClusterWaveforms>	Group	Top level Group for <ClusterWaveforms> • **Neurodata Type:** *ClusterWaveforms* • **Extends:** *NWBDataInterface* • **Default Name:** ClusterWaveforms
—clustering_interface	Link	Link to Clustering interface that was the source of the clustered data • **Target Type** *Clustering* • **Name:** clustering_interface

4.8.10 Clustering

Overview: DEPRECATED Clustered spike data, whether from automatic clustering tools (e.g., klustakwik) or as a result of manual sorting.

`Clustering` extends `NWBDataInterface` and includes all elements of *NWBDataInterface* with the following additions or changes.

- **Extends:** *NWBDataInterface*
- **Primitive Type:** Group
- **Default Name:** Clustering
- **Inherits from:** *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.ecephys.yaml
- **Source Specification:** see Section 5.9.10

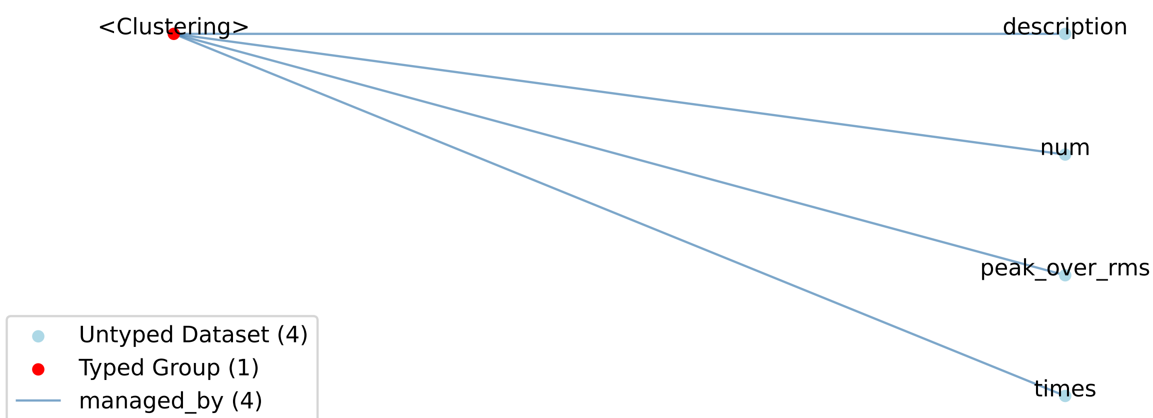


Table 4.76: Datasets, Links, and Attributes contained in <Clustering>

Id	Type	Description
<Clustering>	Group	Top level Group for <Clustering> • **Neurodata Type:** *Clustering* • **Extends:** *NWBDataInterface* • **Default Name:** Clustering
—description	Dataset	Description of clusters or clustering, (e.g. cluster 0 is noise, clusters curated using Klusters, etc) • **Data Type:** text • **Name:** description
—num	Dataset	Cluster number of each event • **Data Type:** int32 • **Dimensions:** ['num_events'] • **Shape:** [None] • **Name:** num

continues on next page

Table 4.76 – continued from previous page

Id	Type	Description
—peak_over_rms	Dataset	Maximum ratio of waveform peak to RMS on any channel in the cluster (provides a basic clustering metric). • **Data Type:** float32 • **Dimensions:** ['num_clusters'] • **Shape:** [None] • **Name:** peak_over_rms
—times	Dataset	Times of clustered events, in seconds. This may be a link to times field in associated FeatureExtraction module. • **Data Type:** float64 • **Dimensions:** ['num_events'] • **Shape:** [None] • **Name:** times

4.9 Intracellular electrophysiology

This source module contains neurodata_types for intracellular electrophysiology data.

4.9.1 PatchClampSeries

Overview: An abstract base class for patch-clamp data - stimulus or response, current or voltage.

`PatchClampSeries` extends `TimeSeries` and includes all elements of *TimeSeries* with the following additions or changes.

- **Extends:** *TimeSeries*
- **Primitive Type:** Group
- **Inherits from:** *TimeSeries*, *NWBDataInterface*, *NWBContainer*, Container
- **Subtypes:** *VoltageClampStimulusSeries*, *CurrentClampSeries*, *IZeroClampSeries*, *CurrentClampStimulusSeries*, *VoltageClampSeries*
- **Source filename:** nwb.icephys.yaml
- **Source Specification:** see Section 5.10.1

Table 4.77: Datasets, Links, and Attributes contained in <PatchClampSeries>

Id	Type	Description
<PatchClampSeries>	Group	Top level Group for <PatchClampSeries> • **Neurodata Type:** *PatchClampSeries* • **Extends:** *TimeSeries*
—stimulus_description	Attribute	Protocol/stimulus name for this patch-clamp dataset. • **Data Type:** text • **Name:** stimulus_description
—sweep_number	Attribute	Sweep number, allows to group different PatchClampSeries together. • **Data Type:** uint32 • **Required:** False • **Name:** sweep_number
—description	Attribute	Description of the time series. • **Data Type:** text • **Required:** False • **Default Value:** no description • **Name:** description

continues on next page

Table 4.77 – continued from previous page

Id	Type	Description
—comments	Attribute	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. • **Data Type:** text • **Required:** False • **Default Value:** no comments • **Name:** comments
—data	Dataset	Recorded voltage or current. • **Data Type:** numeric • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** data
——unit	Attribute	Base unit of measurement for working with the data. Actual stored values are not necessarily stored in these units. To access the data in these units, multiply 'data' by 'conversion' and add 'offset'. • **Data Type:** text • **Name:** unit
——conversion	Attribute	Scalar to multiply each element in data to convert it to the specified 'unit'. If the data are stored in acquisition system units or other units that require a conversion to be interpretable, multiply the data by 'conversion' to convert the data to the specified 'unit'. e.g. if the data acquisition system stores values in this object as signed 16-bit integers (int16 range -32,768 to 32,767) that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition values to recorded volts is 2.5/32768/8000 = 9.5367e-9. • **Data Type:** float32 • **Required:** False • **Default Value:** 1.0 • **Name:** conversion
——offset	Attribute	Scalar to add to the data after scaling by 'conversion' to finalize its coercion to the specified 'unit'. Two common examples of this include (a) data stored in an unsigned type that requires a shift after scaling to re-center the data, and (b) specialized recording devices that naturally cause a scalar offset with respect to the true units. • **Data Type:** float32 • **Required:** False • **Default Value:** 0.0 • **Name:** offset

continues on next page

Table 4.77 – continued from previous page

Id	Type	Description
——resolution	Attribute	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 -1.0. • **Data Type:** float32 • **Required:** False • **Default Value:** -1.0 • **Name:** resolution
——continuity	Attribute	Optionally describe the continuity of the data. Can be "continuous", "instantaneous", or "step". For example, a voltage trace would be "continuous", because samples are recorded from a continuous process. An array of lick times would be "instantaneous", because the data represents distinct moments in time. Times of image presentations would be "step" because the picture remains the same until the next timepoint. This field is optional, but is useful in providing information about the underlying data. It may inform the way this data is interpreted, the way it is visualized, and what analysis methods are applicable. • **Data Type:** text • **Required:** False • **Name:** continuity
—gain	Dataset	Gain of the recording, in units Volt/Amp (v-clamp) or Volt/Volt (c-clamp). • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** gain
—starting_time	Dataset	Timestamp of the first sample in seconds. When timestamps are uniformly spaced, the timestamp of the first sample can be specified and all subsequent ones calculated from the sampling rate attribute. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Name:** starting_time
——rate	Attribute	Sampling rate, in Hz. • **Data Type:** float32 • **Name:** rate
——unit	Attribute	Unit of measurement for time, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit

continues on next page

Table 4.77 – continued from previous page

Id	Type	Description
—timestamps	Dataset	Timestamps for samples stored in data, in seconds, relative to the common experiment master-clock stored in NWBFile.timestamps_reference_time. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** timestamps
——interval	Attribute	Value is '1' • **Data Type:** int32 • **Value:** 1 • **Name:** interval
——unit	Attribute	Unit of measurement for timestamps, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—control	Dataset	Numerical labels that apply to each time point in data for the purpose of querying and slicing data by these values. If present, the length of this array should be the same size as the first dimension of data. • **Quantity:** 0 or 1 • **Data Type:** uint8 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** control
—control_description	Dataset	Description of each control value. Must be present if control is present. If present, control_description[0] should describe time points where control == 0. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** ['num_control_values'] • **Shape:** [None] • **Name:** control_description
—electrode	Link	Link to IntracellularElectrode object that describes the electrode that was used to apply or record this data. • **Target Type** *IntracellularElectrode* • **Name:** electrode

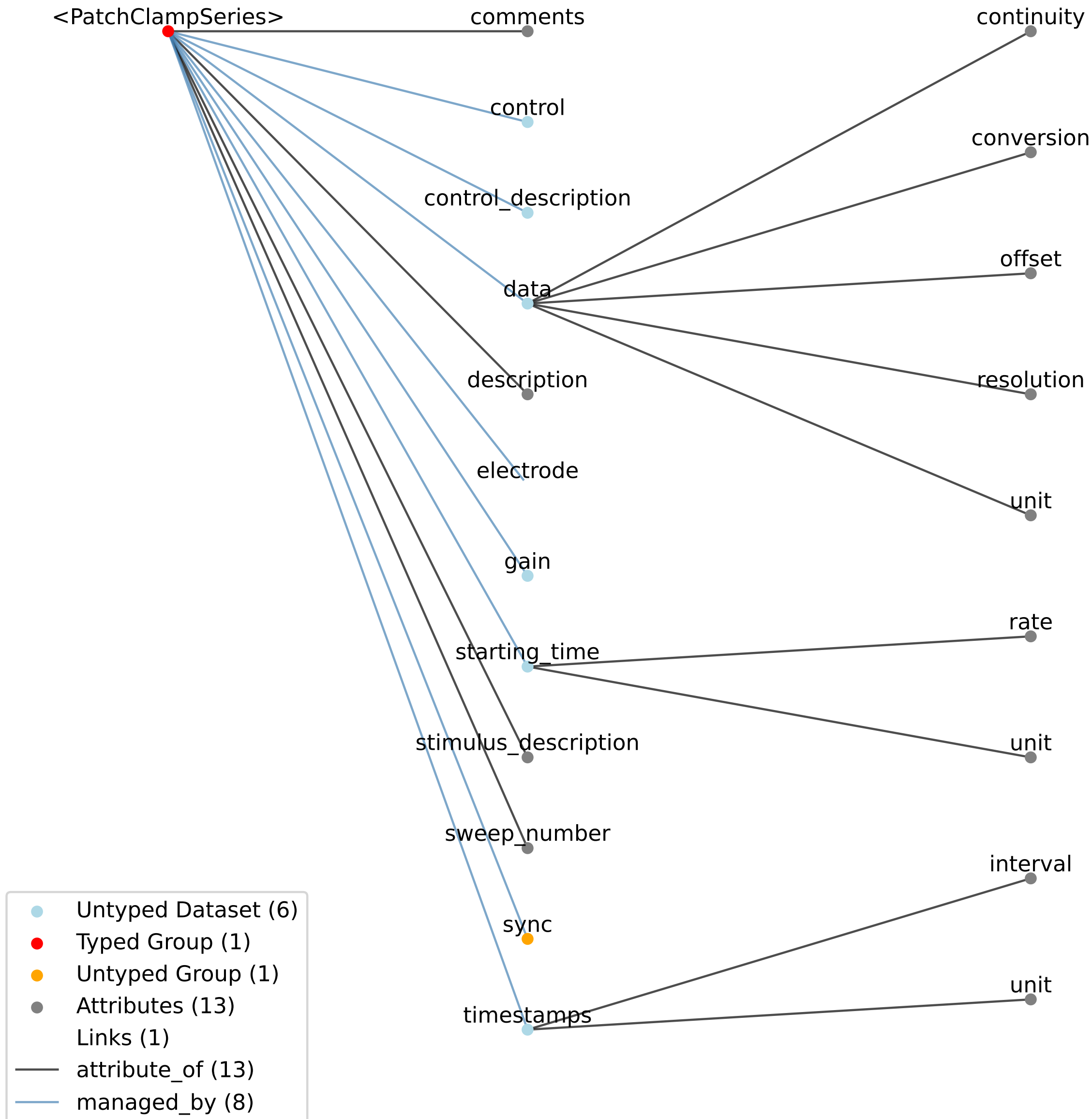

<PatchClampSeries>
comments
control
control_description
data
description
electrode
gain
starting_time
stimulus_description
sweep_number
sync
timestamps
continuity
conversion
offset
resolution
unit
rate
unit
interval
unit
Untyped Dataset (6)
Typed Group (1)
Untyped Group (1)
Attributes (13)
Links (1)
attribute_of (13)
managed_by (8)

Table 4.78: Groups contained in <PatchClampSeries>

Id	Type	Description
<PatchClampSeries>	Group	Top level Group for <PatchClampSeries> • **Neurodata Type:** *PatchClampSeries* • **Extends:** *TimeSeries*
—electrode	Link	Link to IntracellularElectrode object that describes the electrode that was used to apply or record this data. • **Target Type** *IntracellularElectrode* • **Name:** electrode
—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. • **Quantity:** 0 or 1 • **Name:** sync

4.9.1.1 Groups: sync

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.

- **Quantity:** 0 or 1
- **Name:** sync

4.9.2 CurrentClampSeries

Overview: Voltage data from an intracellular current-clamp recording. A corresponding CurrentClampStimulusSeries (stored separately as a stimulus) is used to store the current injected.

`CurrentClampSeries` extends `PatchClampSeries` and includes all elements of *PatchClampSeries* with the following additions or changes.

- **Extends:** *PatchClampSeries*
- **Primitive Type:** Group
- **Inherits from:** *PatchClampSeries*, *TimeSeries*, *NWBDataInterface*, *NWBContainer*, Container
- **Subtypes:** *IZeroClampSeries*
- **Source filename:** nwb.icephys.yaml
- **Source Specification:** see Section 5.10.2

Table 4.79: Datasets, Links, and Attributes contained in <CurrentClampSeries>

Id	Type	Description
<CurrentClampSeries>	Group	Top level Group for <CurrentClampSeries> • **Neurodata Type:** *CurrentClampSeries* • **Extends:** *PatchClampSeries*
—stimulus_description	Attribute	Protocol/stimulus name for this patch-clamp dataset. • **Data Type:** text • **Name:** stimulus_description
—sweep_number	Attribute	Sweep number, allows to group different PatchClampSeries together. • **Data Type:** uint32 • **Required:** False • **Name:** sweep_number
—description	Attribute	Description of the time series. • **Data Type:** text • **Required:** False • **Default Value:** no description • **Name:** description
—comments	Attribute	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. • **Data Type:** text • **Required:** False • **Default Value:** no comments • **Name:** comments
—data	Dataset	Recorded voltage. • **Name:** data

continues on next page

Table 4.79 – continued from previous page

Id	Type	Description
——unit	Attribute	Base unit of measurement for working with the data. which is fixed to 'volts'. Actual stored values are not necessarily stored in these units. To access the data in these units, multiply 'data' by 'conversion' and add 'offset'. • **Data Type:** text • **Value:** volts • **Name:** unit
——conversion	Attribute	Scalar to multiply each element in data to convert it to the specified 'unit'. If the data are stored in acquisition system units or other units that require a conversion to be interpretable, multiply the data by 'conversion' to convert the data to the specified 'unit'. e.g. if the data acquisition system stores values in this object as signed 16-bit integers (int16 range -32,768 to 32,767) that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition values to recorded volts is 2.5/32768/8000 = 9.5367e-9. • **Data Type:** float32 • **Required:** False • **Default Value:** 1.0 • **Name:** conversion
——offset	Attribute	Scalar to add to the data after scaling by 'conversion' to finalize its coercion to the specified 'unit'. Two common examples of this include (a) data stored in an unsigned type that requires a shift after scaling to re-center the data, and (b) specialized recording devices that naturally cause a scalar offset with respect to the true units. • **Data Type:** float32 • **Required:** False • **Default Value:** 0.0 • **Name:** offset
——resolution	Attribute	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 -1.0. • **Data Type:** float32 • **Required:** False • **Default Value:** -1.0 • **Name:** resolution

continues on next page

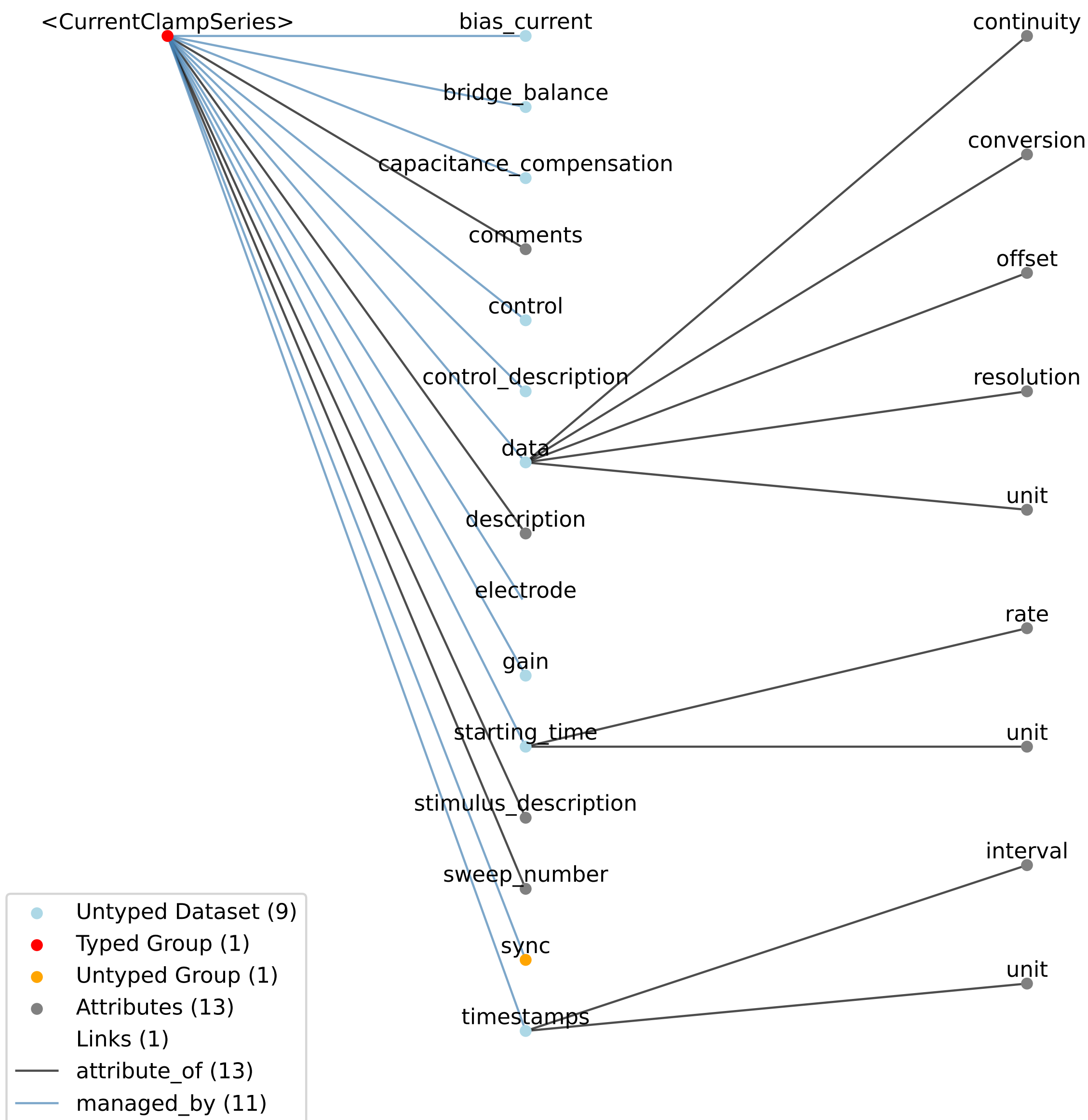
Table 4.79 – continued from previous page

Id	Type	Description
——continuity	Attribute	Optionally describe the continuity of the data. Can be "continuous", "instantaneous", or "step". For example, a voltage trace would be "continuous", because samples are recorded from a continuous process. An array of lick times would be "instantaneous", because the data represents distinct moments in time. Times of image presentations would be "step" because the picture remains the same until the next timepoint. This field is optional, but is useful in providing information about the underlying data. It may inform the way this data is interpreted, the way it is visualized, and what analysis methods are applicable. • **Data Type:** text • **Required:** False • **Name:** continuity
—bias_current	Dataset	Bias current, in amps. • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** bias_current
—bridge_balance	Dataset	Bridge balance, in ohms. • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** bridge_balance
—capacitance_compensation	Dataset	Capacitance compensation, in farads. • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** capacitance_compensation
—gain	Dataset	Gain of the recording, in units Volt/Amp (v-clamp) or Volt/Volt (c-clamp). • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** gain
—starting_time	Dataset	Timestamp of the first sample in seconds. When timestamps are uniformly spaced, the timestamp of the first sample can be specified and all subsequent ones calculated from the sampling rate attribute. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Name:** starting_time
——rate	Attribute	Sampling rate, in Hz. • **Data Type:** float32 • **Name:** rate
——unit	Attribute	Unit of measurement for time, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit

continues on next page

Table 4.79 – continued from previous page

Id	Type	Description
—timestamps	Dataset	Timestamps for samples stored in data, in seconds, relative to the common experiment master-clock stored in NWBFile.timestamps_reference_time. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Dimensions:** [‘num_times’] • **Shape:** [None] • **Name:** timestamps
——interval	Attribute	Value is ‘1’ • **Data Type:** int32 • **Value:** 1 • **Name:** interval
——unit	Attribute	Unit of measurement for timestamps, which is fixed to ‘seconds’. • **Data Type:** text • **Value:** seconds • **Name:** unit
—control	Dataset	Numerical labels that apply to each time point in data for the purpose of querying and slicing data by these values. If present, the length of this array should be the same size as the first dimension of data. • **Quantity:** 0 or 1 • **Data Type:** uint8 • **Dimensions:** [‘num_times’] • **Shape:** [None] • **Name:** control
—control_description	Dataset	Description of each control value. Must be present if control is present. If present, control_description[0] should describe time points where control == 0. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** [‘num_control_values’] • **Shape:** [None] • **Name:** control_description
—electrode	Link	Link to IntracellularElectrode object that describes the electrode that was used to apply or record this data. • **Target Type** *IntracellularElectrode* • **Name:** electrode



<CurrentClampSeries>
bias_current
bridge_balance
capacitance_compensation
comments
control
control_description
data
description
electrode
gain
starting_time
stimulus_description
sweep_number
sync
timestamps
continuity
conversion
offset
resolution
unit
rate
unit
interval
unit
Untyped Dataset (9)
Typed Group (1)
Untyped Group (1)
Attributes (13)
Links (1)
attribute_of (13)
managed_by (11)

Table 4.80: Groups contained in <CurrentClampSeries>

Id	Type	Description
<CurrentClampSeries>	Group	Top level Group for <CurrentClampSeries> • **Neurodata Type:** *CurrentClampSeries* • **Extends:** *PatchClampSeries*
—electrode	Link	Link to IntracellularElectrode object that describes the electrode that was used to apply or record this data. • **Target Type** *IntracellularElectrode* • **Name:** electrode
—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. • **Quantity:** 0 or 1 • **Name:** sync

4.9.2.1 Groups: sync

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.

- **Quantity:** 0 or 1
- **Name:** sync

4.9.3 IZeroClampSeries

Overview: Voltage data from an intracellular recording 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` extends `CurrentClampSeries` and includes all elements of *CurrentClampSeries* with the following additions or changes.

- **Extends:** *CurrentClampSeries*
- **Primitive Type:** Group
- **Inherits from:** *CurrentClampSeries*, *PatchClampSeries*, *TimeSeries*, *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.icephys.yaml
- **Source Specification:** see Section 5.10.3

Table 4.81: Datasets, Links, and Attributes contained in <IZeroClampSeries>

Id	Type	Description
<IZeroClampSeries>	Group	Top level Group for <IZeroClampSeries> • **Neurodata Type:** *IZeroClampSeries* • **Extends:** *CurrentClampSeries*
—stimulus_description	Attribute	An IZeroClampSeries has no stimulus, so this attribute is automatically set to “N/A” • **Data Type:** text • **Value:** N/A • **Name:** stimulus_description
—sweep_number	Attribute	Sweep number, allows to group different PatchClampSeries together. • **Data Type:** uint32 • **Required:** False • **Name:** sweep_number
—description	Attribute	Description of the time series. • **Data Type:** text • **Required:** False • **Default Value:** no description • **Name:** description
—comments	Attribute	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. • **Data Type:** text • **Required:** False • **Default Value:** no comments • **Name:** comments

continues on next page

Table 4.81 – continued from previous page

Id	Type	Description
—bias_current	Dataset	Bias current, in amps, fixed to 0.0. • **Data Type:** float32 • **Name:** bias_current
—bridge_balance	Dataset	Bridge balance, in ohms, fixed to 0.0. • **Data Type:** float32 • **Name:** bridge_balance
—capaci-tance_compensation	Dataset	Capacitance compensation, in farads, fixed to 0.0. • **Data Type:** float32 • **Name:** capacitance_compensation
—data	Dataset	Recorded voltage. • **Name:** data
——unit	At-tribute	Base unit of measurement for working with the data. which is fixed to 'volts'. Actual stored values are not necessarily stored in these units. To access the data in these units, multiply 'data' by 'conversion' and add 'offset'. • **Data Type:** text • **Value:** volts • **Name:** unit
——conversion	At-tribute	Scalar to multiply each element in data to convert it to the specified 'unit'. If the data are stored in acquisition system units or other units that require a conversion to be interpretable, multiply the data by 'conversion' to convert the data to the specified 'unit'. e.g. if the data acquisition system stores values in this object as signed 16-bit integers (int16 range -32,768 to 32,767) that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition values to recorded volts is 2.5/32768/8000 = 9.5367e-9. • **Data Type:** float32 • **Required:** False • **Default Value:** 1.0 • **Name:** conversion
——offset	At-tribute	Scalar to add to the data after scaling by 'conversion' to finalize its coercion to the specified 'unit'. Two common examples of this include (a) data stored in an unsigned type that requires a shift after scaling to re-center the data, and (b) specialized recording devices that naturally cause a scalar offset with respect to the true units. • **Data Type:** float32 • **Required:** False • **Default Value:** 0.0 • **Name:** offset

continues on next page

Table 4.81 – continued from previous page

Id	Type	Description
——resolution	Attribute	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 -1.0. • **Data Type:** float32 • **Required:** False • **Default Value:** -1.0 • **Name:** resolution
——continuity	Attribute	Optionally describe the continuity of the data. Can be "continuous", "instantaneous", or "step". For example, a voltage trace would be "continuous", because samples are recorded from a continuous process. An array of lick times would be "instantaneous", because the data represents distinct moments in time. Times of image presentations would be "step" because the picture remains the same until the next timepoint. This field is optional, but is useful in providing information about the underlying data. It may inform the way this data is interpreted, the way it is visualized, and what analysis methods are applicable. • **Data Type:** text • **Required:** False • **Name:** continuity
—gain	Dataset	Gain of the recording, in units Volt/Amp (v-clamp) or Volt/Volt (c-clamp). • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** gain
—starting_time	Dataset	Timestamp of the first sample in seconds. When timestamps are uniformly spaced, the timestamp of the first sample can be specified and all subsequent ones calculated from the sampling rate attribute. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Name:** starting_time
——rate	Attribute	Sampling rate, in Hz. • **Data Type:** float32 • **Name:** rate
——unit	Attribute	Unit of measurement for time, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit

continues on next page

Table 4.81 – continued from previous page

Id	Type	Description
—timestamps	Dataset	Timestamps for samples stored in data, in seconds, relative to the common experiment master-clock stored in NWBFile.timestamps_reference_time. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** timestamps
——interval	Attribute	Value is '1' • **Data Type:** int32 • **Value:** 1 • **Name:** interval
——unit	Attribute	Unit of measurement for timestamps, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—control	Dataset	Numerical labels that apply to each time point in data for the purpose of querying and slicing data by these values. If present, the length of this array should be the same size as the first dimension of data. • **Quantity:** 0 or 1 • **Data Type:** uint8 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** control
—control_description	Dataset	Description of each control value. Must be present if control is present. If present, control_description[0] should describe time points where control == 0. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** ['num_control_values'] • **Shape:** [None] • **Name:** control_description
—electrode	Link	Link to IntracellularElectrode object that describes the electrode that was used to apply or record this data. • **Target Type** *IntracellularElectrode* • **Name:** electrode

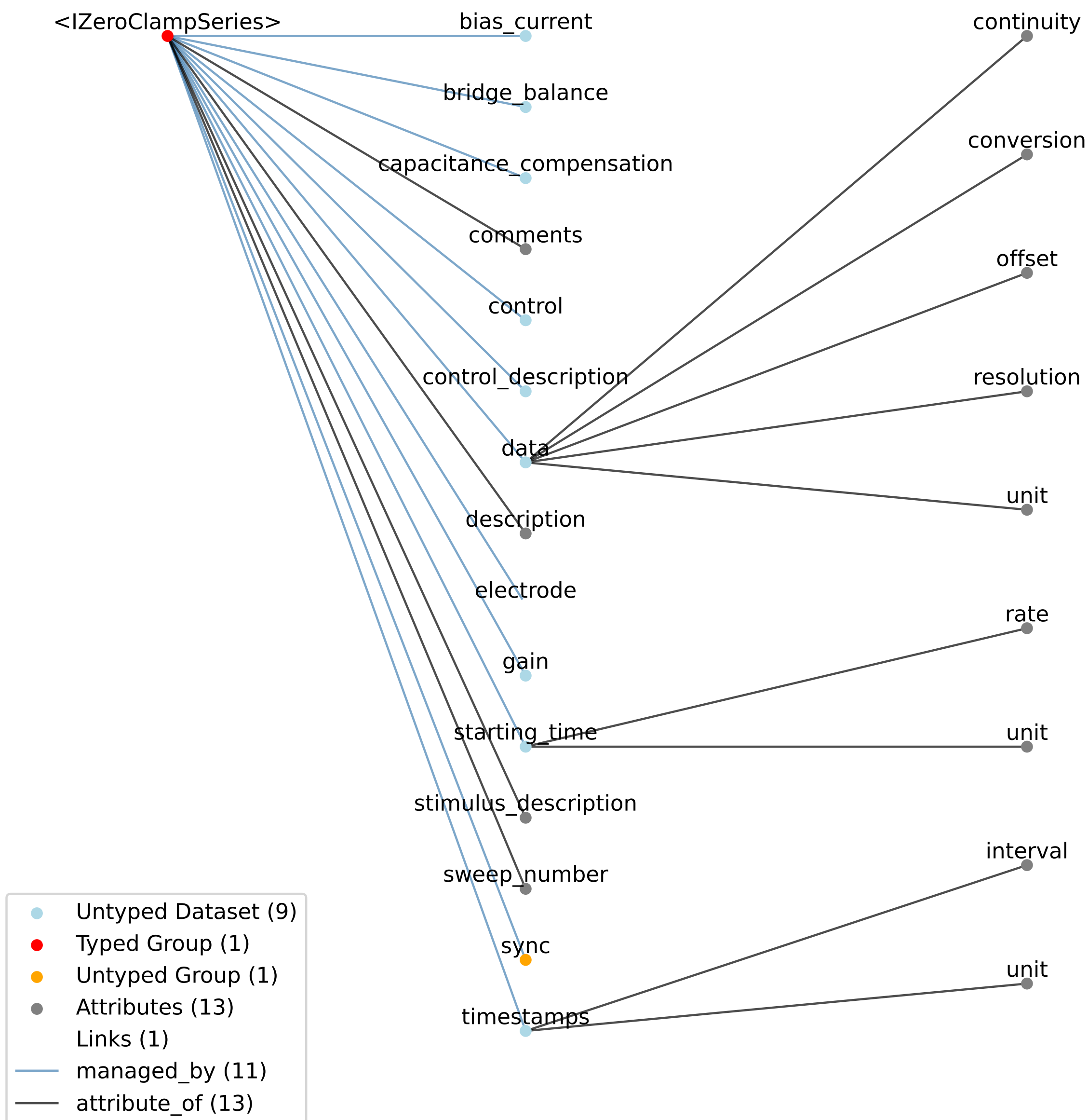

<IZeroClampSeries>
bias_current
bridge_balance
capacitance_compensation
comments
control
control_description
data
description
electrode
gain
starting_time
stimulus_description
sweep_number
sync
timestamps
continuity
conversion
offset
resolution
unit
rate
unit
interval
unit
Untyped Dataset (9)
Typed Group (1)
Untyped Group (1)
Attributes (13)
Links (1)
managed_by (11)
attribute_of (13)

Table 4.82: Groups contained in <IZeroClampSeries>

Id	Type	Description
<IZeroClampSeries>	Group	Top level Group for <IZeroClampSeries> • **Neurodata Type:** *IZeroClampSeries* • **Extends:** *CurrentClampSeries*
—electrode	Link	Link to IntracellularElectrode object that describes the electrode that was used to apply or record this data. • **Target Type** *IntracellularElectrode* • **Name:** electrode
—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. • **Quantity:** 0 or 1 • **Name:** sync

4.9.3.1 Groups: sync

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.

- **Quantity:** 0 or 1
- **Name:** sync

4.9.4 CurrentClampStimulusSeries

Overview: Stimulus current applied during current clamp recording.

`CurrentClampStimulusSeries` extends `PatchClampSeries` and includes all elements of *PatchClampSeries* with the following additions or changes.

- **Extends:** *PatchClampSeries*
- **Primitive Type:** Group
- **Inherits from:** *PatchClampSeries*, *TimeSeries*, *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.icephys.yaml
- **Source Specification:** see Section 5.10.4

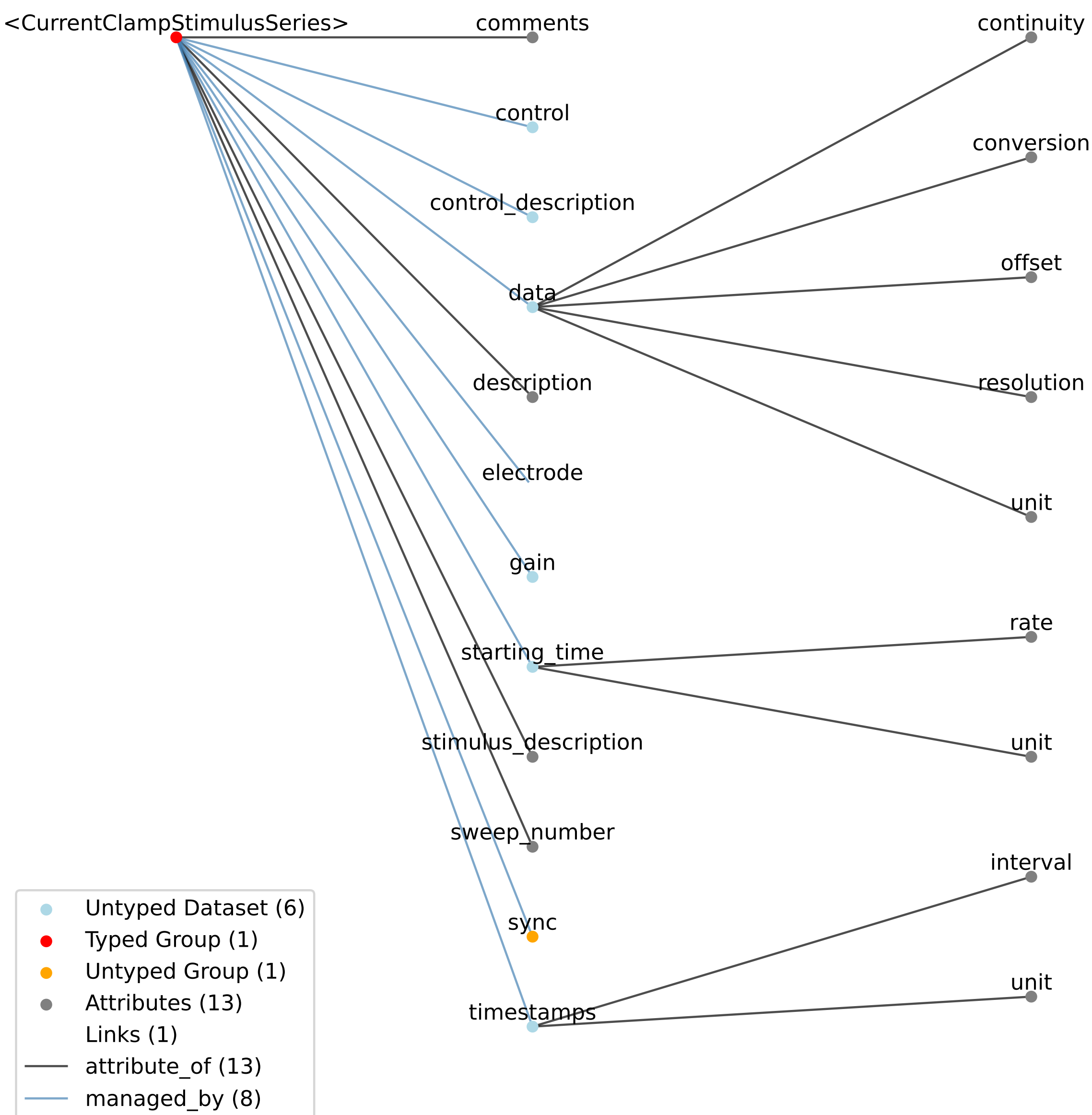


Table 4.83: Datasets, Links, and Attributes contained in <CurrentClampStimulusSeries>

Id	Type	Description
<CurrentClampStimulusSeries>	Group	Top level Group for <CurrentClampStimulusSeries> • **Neurodata Type:** *CurrentClampStimulusSeries* • **Extends:** *PatchClampSeries*
—stimulus_description	Attribute	Protocol/stimulus name for this patch-clamp dataset. • **Data Type:** text • **Name:** stimulus_description
—sweep_number	Attribute	Sweep number, allows to group different PatchClampSeries together. • **Data Type:** uint32 • **Required:** False • **Name:** sweep_number
—description	Attribute	Description of the time series. • **Data Type:** text • **Required:** False • **Default Value:** no description • **Name:** description
—comments	Attribute	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. • **Data Type:** text • **Required:** False • **Default Value:** no comments • **Name:** comments
—data	Dataset	Stimulus current applied. • **Name:** data
——unit	Attribute	Base unit of measurement for working with the data. which is fixed to 'amperes'. Actual stored values are not necessarily stored in these units. To access the data in these units, multiply 'data' by 'conversion' and add 'offset'. • **Data Type:** text • **Value:** amperes • **Name:** unit

continues on next page

Table 4.83 – continued from previous page

Id	Type	Description
——conversion	Attribute	Scalar to multiply each element in data to convert it to the specified 'unit'. If the data are stored in acquisition system units or other units that require a conversion to be interpretable, multiply the data by 'conversion' to convert the data to the specified 'unit'. e.g. if the data acquisition system stores values in this object as signed 16-bit integers (int16 range -32,768 to 32,767) that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition values to recorded volts is 2.5/32768/8000 = 9.5367e-9. • **Data Type:** float32 • **Required:** False • **Default Value:** 1.0 • **Name:** conversion
——offset	Attribute	Scalar to add to the data after scaling by 'conversion' to finalize its coercion to the specified 'unit'. Two common examples of this include (a) data stored in an unsigned type that requires a shift after scaling to re-center the data, and (b) specialized recording devices that naturally cause a scalar offset with respect to the true units. • **Data Type:** float32 • **Required:** False • **Default Value:** 0.0 • **Name:** offset
——resolution	Attribute	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 -1.0. • **Data Type:** float32 • **Required:** False • **Default Value:** -1.0 • **Name:** resolution
——continuity	Attribute	Optionally describe the continuity of the data. Can be "continuous", "instantaneous", or "step". For example, a voltage trace would be "continuous", because samples are recorded from a continuous process. An array of lick times would be "instantaneous", because the data represents distinct moments in time. Times of image presentations would be "step" because the picture remains the same until the next timepoint. This field is optional, but is useful in providing information about the underlying data. It may inform the way this data is interpreted, the way it is visualized, and what analysis methods are applicable. • **Data Type:** text • **Required:** False • **Name:** continuity

continues on next page

Table 4.83 – continued from previous page

Id	Type	Description
—gain	Dataset	Gain of the recording, in units Volt/Amp (v-clamp) or Volt/Volt (c-clamp). • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** gain
—starting_time	Dataset	Timestamp of the first sample in seconds. When timestamps are uniformly spaced, the timestamp of the first sample can be specified and all subsequent ones calculated from the sampling rate attribute. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Name:** starting_time
——rate	Attribute	Sampling rate, in Hz. • **Data Type:** float32 • **Name:** rate
——unit	Attribute	Unit of measurement for time, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—timestamps	Dataset	Timestamps for samples stored in data, in seconds, relative to the common experiment master-clock stored in NWBFile.timestamps_reference_time. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** timestamps
——interval	Attribute	Value is '1' • **Data Type:** int32 • **Value:** 1 • **Name:** interval
——unit	Attribute	Unit of measurement for timestamps, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—control	Dataset	Numerical labels that apply to each time point in data for the purpose of querying and slicing data by these values. If present, the length of this array should be the same size as the first dimension of data. • **Quantity:** 0 or 1 • **Data Type:** uint8 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** control

continues on next page

Table 4.83 – continued from previous page

Id	Type	Description
—control_description	Dataset	Description of each control value. Must be present if control is present. If present, control_description[0] should describe time points where control == 0. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** ['num_control_values'] • **Shape:** [None] • **Name:** control_description
—electrode	Link	Link to IntracellularElectrode object that describes the electrode that was used to apply or record this data. • **Target Type** *IntracellularElectrode* • **Name:** electrode

Table 4.84: Groups contained in <CurrentClampStimulusSeries>

Id	Type	Description
<CurrentClampStimu-lusSeries>	Group	Top level Group for <CurrentClampStimulusSeries> • **Neurodata Type:** *CurrentClampStimulusSeries* • **Extends:** *PatchClampSeries*
—electrode	Link	Link to IntracellularElectrode object that describes the electrode that was used to apply or record this data. • **Target Type** *IntracellularElectrode* • **Name:** electrode
—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. • **Quantity:** 0 or 1 • **Name:** sync

4.9.4.1 Groups: sync

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.

- **Quantity:** 0 or 1
- **Name:** sync

4.9.5 VoltageClampSeries

Overview: Current data from an intracellular voltage-clamp recording. A corresponding VoltageClampStimulusSeries (stored separately as a stimulus) is used to store the voltage injected.

`VoltageClampSeries` extends `PatchClampSeries` and includes all elements of *PatchClampSeries* with the following additions or changes.

- **Extends:** *PatchClampSeries*
- **Primitive Type:** Group
- **Inherits from:** *PatchClampSeries*, *TimeSeries*, *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.icephys.yaml
- **Source Specification:** see Section 5.10.5

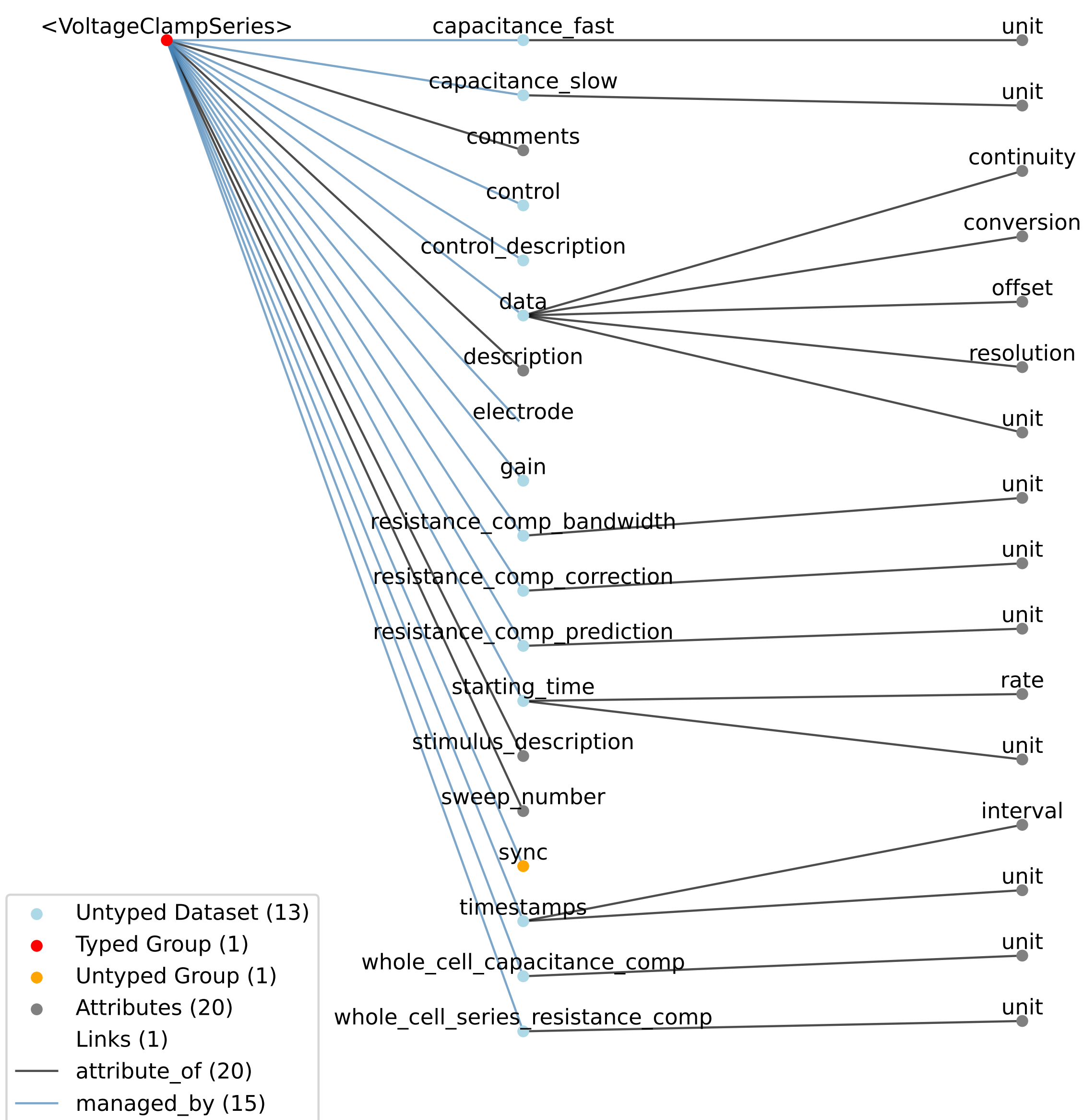


Table 4.85: Datasets, Links, and Attributes contained in <VoltageClampSeries>

Id	Type	Description
<VoltageClampSeries>	Group	Top level Group for <VoltageClampSeries> • **Neurodata Type:** *VoltageClampSeries* • **Extends:** *PatchClampSeries*
—stimulus_description	Attribute	Protocol/stimulus name for this patch-clamp dataset. • **Data Type:** text • **Name:** stimulus_description
—sweep_number	Attribute	Sweep number, allows to group different PatchClampSeries together. • **Data Type:** uint32 • **Required:** False • **Name:** sweep_number
—description	Attribute	Description of the time series. • **Data Type:** text • **Required:** False • **Default Value:** no description • **Name:** description
—comments	Attribute	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. • **Data Type:** text • **Required:** False • **Default Value:** no comments • **Name:** comments
—data	Dataset	Recorded current. • **Name:** data
——unit	Attribute	Base unit of measurement for working with the data. which is fixed to 'amperes'. Actual stored values are not necessarily stored in these units. To access the data in these units, multiply 'data' by 'conversion' and add 'offset'. • **Data Type:** text • **Value:** amperes • **Name:** unit

continues on next page

Table 4.85 – continued from previous page

Id	Type	Description
——conversion	Attribute	Scalar to multiply each element in data to convert it to the specified 'unit'. If the data are stored in acquisition system units or other units that require a conversion to be interpretable, multiply the data by 'conversion' to convert the data to the specified 'unit'. e.g. if the data acquisition system stores values in this object as signed 16-bit integers (int16 range -32,768 to 32,767) that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition values to recorded volts is 2.5/32768/8000 = 9.5367e-9. • **Data Type:** float32 • **Required:** False • **Default Value:** 1.0 • **Name:** conversion
——offset	Attribute	Scalar to add to the data after scaling by 'conversion' to finalize its coercion to the specified 'unit'. Two common examples of this include (a) data stored in an unsigned type that requires a shift after scaling to re-center the data, and (b) specialized recording devices that naturally cause a scalar offset with respect to the true units. • **Data Type:** float32 • **Required:** False • **Default Value:** 0.0 • **Name:** offset
——resolution	Attribute	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 -1.0. • **Data Type:** float32 • **Required:** False • **Default Value:** -1.0 • **Name:** resolution
——continuity	Attribute	Optionally describe the continuity of the data. Can be "continuous", "instantaneous", or "step". For example, a voltage trace would be "continuous", because samples are recorded from a continuous process. An array of lick times would be "instantaneous", because the data represents distinct moments in time. Times of image presentations would be "step" because the picture remains the same until the next timepoint. This field is optional, but is useful in providing information about the underlying data. It may inform the way this data is interpreted, the way it is visualized, and what analysis methods are applicable. • **Data Type:** text • **Required:** False • **Name:** continuity

continues on next page

Table 4.85 – continued from previous page

Id	Type	Description
—capacitance_fast	Dataset	Fast capacitance, in farads. • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** capacitance_fast
——unit	Attribute	Unit of measurement for capacitance_fast, which is fixed to 'farads'. • **Data Type:** text • **Value:** farads • **Name:** unit
—capacitance_slow	Dataset	Slow capacitance, in farads. • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** capacitance_slow
——unit	Attribute	Unit of measurement for capacitance_fast, which is fixed to 'farads'. • **Data Type:** text • **Value:** farads • **Name:** unit
—resistance_comp_bandwidth	Dataset	Resistance compensation bandwidth, in hertz. • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** resistance_comp_bandwidth
——unit	Attribute	Unit of measurement for resistance_comp_bandwidth, which is fixed to 'hertz'. • **Data Type:** text • **Value:** hertz • **Name:** unit
—resistance_comp_correction	Dataset	Resistance compensation correction, in percent. • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** resistance_comp_correction
——unit	Attribute	Unit of measurement for resistance_comp_correction, which is fixed to 'percent'. • **Data Type:** text • **Value:** percent • **Name:** unit
—resistance_comp_prediction	Dataset	Resistance compensation prediction, in percent. • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** resistance_comp_prediction

continues on next page

Table 4.85 – continued from previous page

Id	Type	Description
——unit	Attribute	Unit of measurement for resistance_comp_prediction, which is fixed to 'percent'. • **Data Type:** text • **Value:** percent • **Name:** unit
—whole_cell_capacitance_co	Dataset	Whole cell capacitance compensation, in farads. • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** whole_cell_capacitance_comp
——unit	Attribute	Unit of measurement for whole_cell_capacitance_comp, which is fixed to 'farads'. • **Data Type:** text • **Value:** farads • **Name:** unit
—whole_cell_series_resistanc	Dataset	Whole cell series resistance compensation, in ohms. • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** whole_cell_series_resistance_comp
——unit	Attribute	Unit of measurement for whole_cell_series_resistance_comp, which is fixed to 'ohms'. • **Data Type:** text • **Value:** ohms • **Name:** unit
—gain	Dataset	Gain of the recording, in units Volt/Amp (v-clamp) or Volt/Volt (c-clamp). • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** gain
—starting_time	Dataset	Timestamp of the first sample in seconds. When timestamps are uniformly spaced, the timestamp of the first sample can be specified and all subsequent ones calculated from the sampling rate attribute. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Name:** starting_time
——rate	Attribute	Sampling rate, in Hz. • **Data Type:** float32 • **Name:** rate
——unit	Attribute	Unit of measurement for time, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit

continues on next page

Table 4.85 – continued from previous page

Id	Type	Description
—timestamps	Dataset	Timestamps for samples stored in data, in seconds, relative to the common experiment master-clock stored in NWBFile.timestamps_reference_time. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** timestamps
——interval	Attribute	Value is '1' • **Data Type:** int32 • **Value:** 1 • **Name:** interval
——unit	Attribute	Unit of measurement for timestamps, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—control	Dataset	Numerical labels that apply to each time point in data for the purpose of querying and slicing data by these values. If present, the length of this array should be the same size as the first dimension of data. • **Quantity:** 0 or 1 • **Data Type:** uint8 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** control
—control_description	Dataset	Description of each control value. Must be present if control is present. If present, control_description[0] should describe time points where control == 0. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** ['num_control_values'] • **Shape:** [None] • **Name:** control_description
—electrode	Link	Link to IntracellularElectrode object that describes the electrode that was used to apply or record this data. • **Target Type** *IntracellularElectrode* • **Name:** electrode

Table 4.86: Groups contained in <VoltageClampSeries>

Id	Type	Description
<VoltageClampSeries>	Group	Top level Group for <VoltageClampSeries> • **Neurodata Type:** *VoltageClampSeries* • **Extends:** *PatchClampSeries*
—electrode	Link	Link to IntracellularElectrode object that describes the electrode that was used to apply or record this data. • **Target Type** *IntracellularElectrode* • **Name:** electrode
—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. • **Quantity:** 0 or 1 • **Name:** sync

4.9.5.1 Groups: sync

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.

- **Quantity:** 0 or 1
- **Name:** sync

4.9.6 VoltageClampStimulusSeries

Overview: Stimulus voltage applied during a voltage clamp recording.

`VoltageClampStimulusSeries` extends `PatchClampSeries` and includes all elements of *PatchClampSeries* with the following additions or changes.

- **Extends:** *PatchClampSeries*
- **Primitive Type:** Group
- **Inherits from:** *PatchClampSeries*, *TimeSeries*, *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.icephys.yaml
- **Source Specification:** see Section 5.10.6

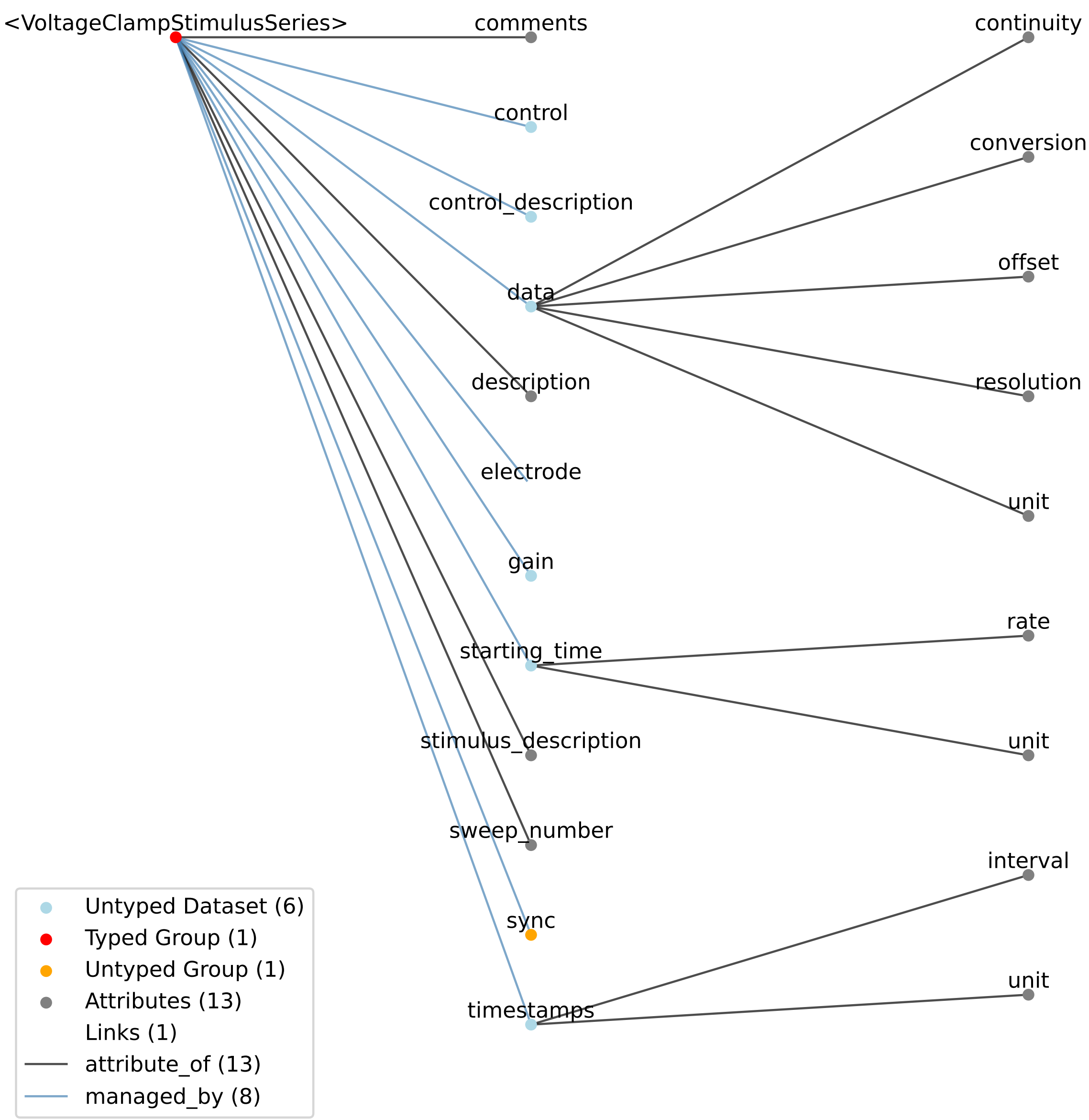


Table 4.87: Datasets, Links, and Attributes contained in <VoltageClampStimulusSeries>

Id	Type	Description
<VoltageClampStimulusSeries>	Group	Top level Group for <VoltageClampStimulusSeries> • **Neurodata Type:** *VoltageClampStimulusSeries* • **Extends:** *PatchClampSeries*
—stimulus_description	Attribute	Protocol/stimulus name for this patch-clamp dataset. • **Data Type:** text • **Name:** stimulus_description
—sweep_number	Attribute	Sweep number, allows to group different PatchClampSeries together. • **Data Type:** uint32 • **Required:** False • **Name:** sweep_number
—description	Attribute	Description of the time series. • **Data Type:** text • **Required:** False • **Default Value:** no description • **Name:** description
—comments	Attribute	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. • **Data Type:** text • **Required:** False • **Default Value:** no comments • **Name:** comments
—data	Dataset	Stimulus voltage applied. • **Name:** data
——unit	Attribute	Base unit of measurement for working with the data. which is fixed to 'volts'. Actual stored values are not necessarily stored in these units. To access the data in these units, multiply 'data' by 'conversion' and add 'offset'. • **Data Type:** text • **Value:** volts • **Name:** unit

continues on next page

Table 4.87 – continued from previous page

Id	Type	Description
——conversion	Attribute	Scalar to multiply each element in data to convert it to the specified 'unit'. If the data are stored in acquisition system units or other units that require a conversion to be interpretable, multiply the data by 'conversion' to convert the data to the specified 'unit'. e.g. if the data acquisition system stores values in this object as signed 16-bit integers (int16 range -32,768 to 32,767) that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition values to recorded volts is 2.5/32768/8000 = 9.5367e-9. • **Data Type:** float32 • **Required:** False • **Default Value:** 1.0 • **Name:** conversion
——offset	Attribute	Scalar to add to the data after scaling by 'conversion' to finalize its coercion to the specified 'unit'. Two common examples of this include (a) data stored in an unsigned type that requires a shift after scaling to re-center the data, and (b) specialized recording devices that naturally cause a scalar offset with respect to the true units. • **Data Type:** float32 • **Required:** False • **Default Value:** 0.0 • **Name:** offset
——resolution	Attribute	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 -1.0. • **Data Type:** float32 • **Required:** False • **Default Value:** -1.0 • **Name:** resolution
——continuity	Attribute	Optionally describe the continuity of the data. Can be "continuous", "instantaneous", or "step". For example, a voltage trace would be "continuous", because samples are recorded from a continuous process. An array of lick times would be "instantaneous", because the data represents distinct moments in time. Times of image presentations would be "step" because the picture remains the same until the next timepoint. This field is optional, but is useful in providing information about the underlying data. It may inform the way this data is interpreted, the way it is visualized, and what analysis methods are applicable. • **Data Type:** text • **Required:** False • **Name:** continuity

continues on next page

Table 4.87 – continued from previous page

Id	Type	Description
—gain	Dataset	Gain of the recording, in units Volt/Amp (v-clamp) or Volt/Volt (c-clamp). • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** gain
—starting_time	Dataset	Timestamp of the first sample in seconds. When timestamps are uniformly spaced, the timestamp of the first sample can be specified and all subsequent ones calculated from the sampling rate attribute. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Name:** starting_time
——rate	Attribute	Sampling rate, in Hz. • **Data Type:** float32 • **Name:** rate
——unit	Attribute	Unit of measurement for time, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—timestamps	Dataset	Timestamps for samples stored in data, in seconds, relative to the common experiment master-clock stored in NWBFile.timestamps_reference_time. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** timestamps
——interval	Attribute	Value is '1' • **Data Type:** int32 • **Value:** 1 • **Name:** interval
——unit	Attribute	Unit of measurement for timestamps, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—control	Dataset	Numerical labels that apply to each time point in data for the purpose of querying and slicing data by these values. If present, the length of this array should be the same size as the first dimension of data. • **Quantity:** 0 or 1 • **Data Type:** uint8 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** control

continues on next page

Table 4.87 – continued from previous page

Id	Type	Description
—control_description	Dataset	Description of each control value. Must be present if control is present. If present, control_description[0] should describe time points where control == 0. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** ['num_control_values'] • **Shape:** [None] • **Name:** control_description
—electrode	Link	Link to IntracellularElectrode object that describes the electrode that was used to apply or record this data. • **Target Type** *IntracellularElectrode* • **Name:** electrode

Table 4.88: Groups contained in <VoltageClampStimulusSeries>

Id	Type	Description
<VoltageClampStimulusSeries>	Group	Top level Group for <VoltageClampStimulusSeries> • **Neurodata Type:** *VoltageClampStimulusSeries* • **Extends:** *PatchClampSeries*
—electrode	Link	Link to IntracellularElectrode object that describes the electrode that was used to apply or record this data. • **Target Type** *IntracellularElectrode* • **Name:** electrode
—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. • **Quantity:** 0 or 1 • **Name:** sync

4.9.6.1 Groups: sync

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.

- **Quantity:** 0 or 1
- **Name:** sync

4.9.7 IntracellularElectrode

Overview: An intracellular electrode and its metadata.

`IntracellularElectrode` extends `NWBContainer` and includes all elements of *NWBContainer* with the following additions or changes.

- **Extends:** *NWBContainer*
- **Primitive Type:** Group
- **Inherits from:** *NWBContainer*, Container
- **Source filename:** nwb.icephys.yaml
- **Source Specification:** see Section 5.10.7

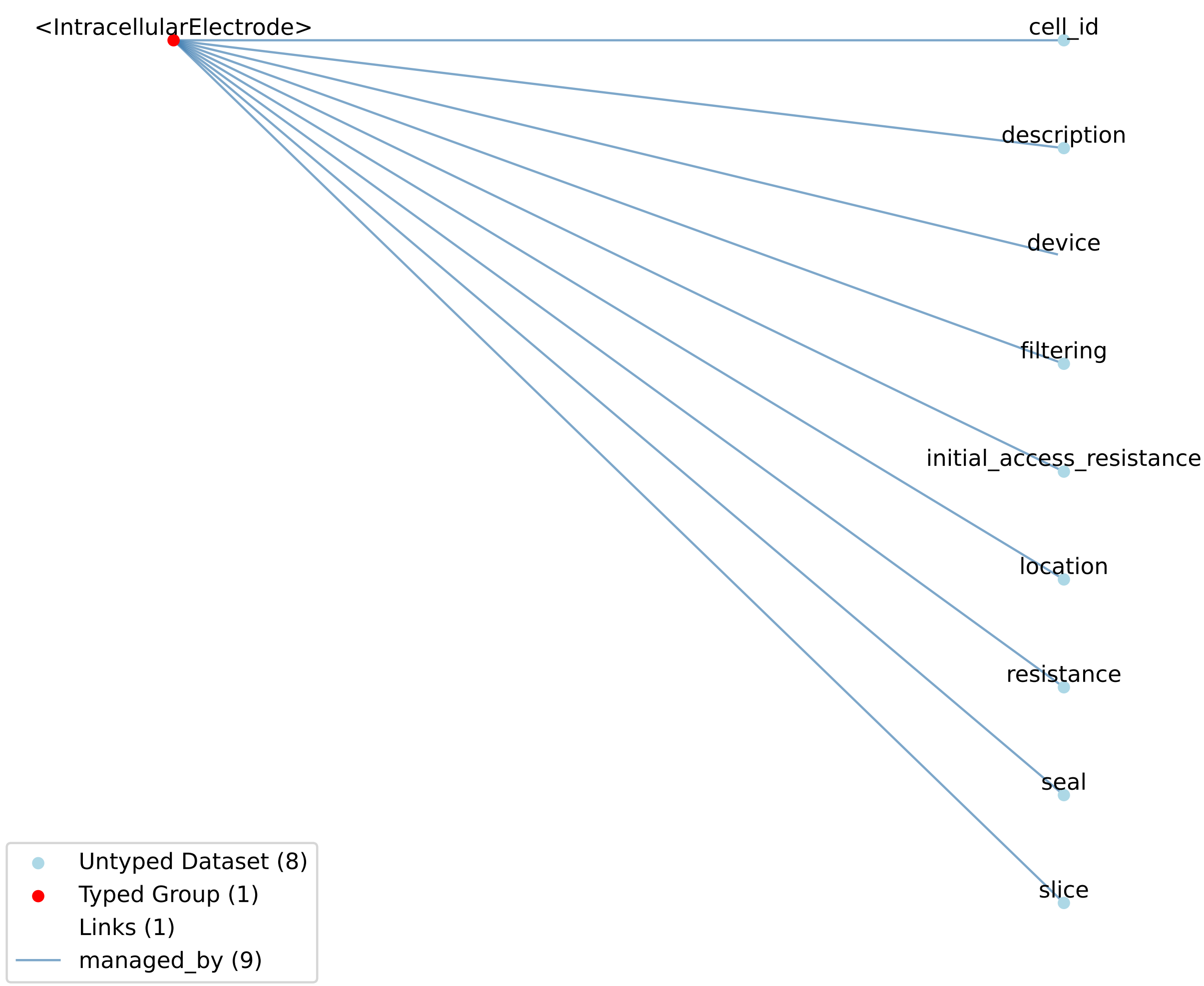


Table 4.89: Datasets, Links, and Attributes contained in <IntracellularElectrode>

Id	Type	Description
<IntracellularElectrode>	Group	Top level Group for <IntracellularElectrode> • **Neurodata Type:** *IntracellularElectrode* • **Extends:** *NWBContainer*
—cell_id	Dataset	unique ID of the cell • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** cell_id
—description	Dataset	Description of electrode (e.g., whole-cell, sharp, etc.). • **Data Type:** text • **Name:** description
—filtering	Dataset	Electrode specific filtering. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** filtering
—initial_access_resistance	Dataset	Initial access resistance. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** initial_access_resistance
—location	Dataset	Location of the electrode. Specify the area, layer, comments on estimation of area/layer, stereotaxic coordinates if in vivo, etc. Use standard atlas names for anatomical regions when possible. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** location
—resistance	Dataset	Electrode resistance, in ohms. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** resistance
—seal	Dataset	Information about seal used for recording. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** seal
—slice	Dataset	Information about slice used for recording. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** slice
—device	Link	Device that was used to record from this electrode. • **Target Type** *Device* • **Name:** device

Table 4.90: Groups contained in <IntracellularElectrode>

Id	Type	Description
<IntracellularElectrode>	Group	Top level Group for <IntracellularElectrode> • **Neurodata Type:** *IntracellularElectrode* • **Extends:** *NWBContainer*
—device	Link	Device that was used to record from this electrode. • **Target Type** *Device* • **Name:** device

4.9.8 SweepTable

Overview: [DEPRECATED] Table used to group different PatchClampSeries. SweepTable is being replaced by IntracellularRecordingsTable and SimultaneousRecordingsTable tables. Additional SequentialRecordingsTable, RepetitionsTable, and ExperimentalConditions tables provide enhanced support for experiment metadata.

`SweepTable` extends `DynamicTable` and includes all elements of DynamicTable with the following additions or changes.

- **Extends:** DynamicTable
- **Primitive Type:** Group
- **Inherits from:** DynamicTable, Container
- **Source filename:** nwb.icephys.yaml
- **Source Specification:** see Section 5.10.8

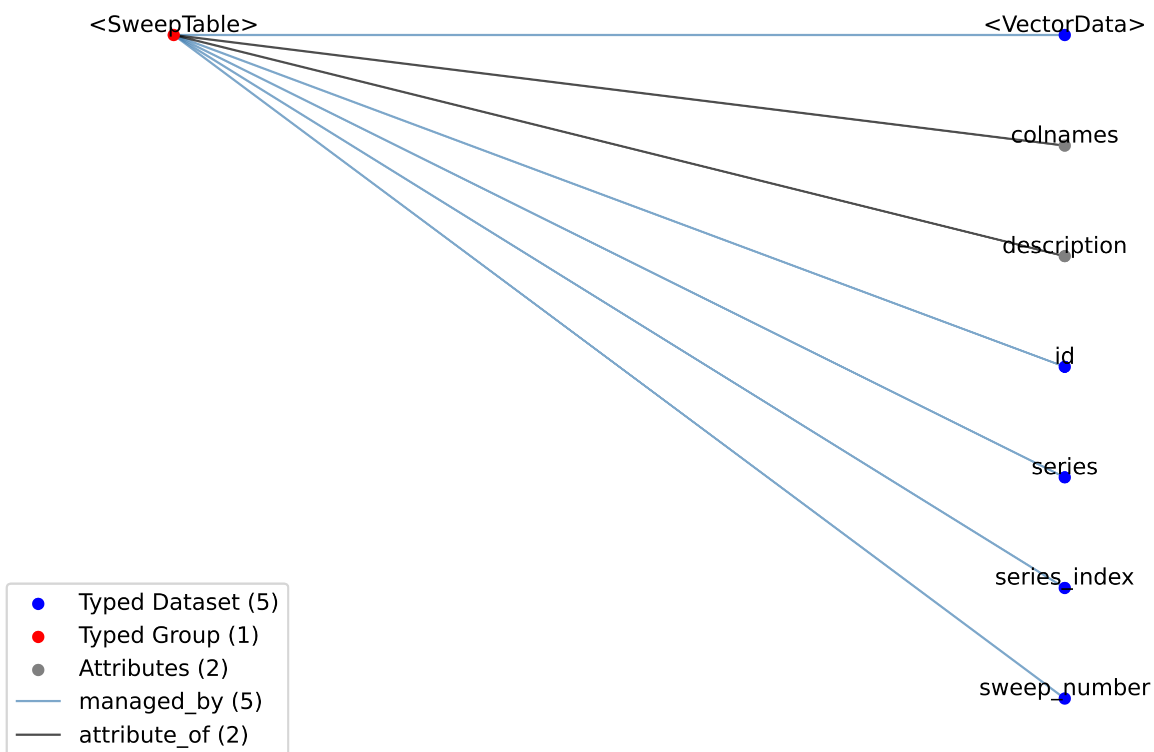


Table 4.91: Datasets, Links, and Attributes contained in <SweepTable>

Id	Type	Description
<SweepTable>	Group	Top level Group for <SweepTable> • **Neurodata Type:** *SweepTable* • **Extends:** DynamicTable

continues on next page

Table 4.91 – continued from previous page

Id	Type	Description
—colnames	Attribute	The names of the columns in this table. This should be used to specify an order to the columns. • **Data Type:** text • **Dimensions:** [‘num_columns’] • **Shape:** [None] • **Name:** colnames
—description	Attribute	Description of what is in this dynamic table. • **Data Type:** text • **Name:** description
—sweep_number	Dataset	Sweep number of the PatchClampSeries in that row. • **Extends:** VectorData • **Data Type:** uint32 • **Name:** sweep_number
—series	Dataset	The PatchClampSeries with the sweep number in that row. • **Extends:** VectorData • **Data Type:** object reference to *PatchClampSeries* • **Name:** series
—series_index	Dataset	Index for series. • **Extends:** VectorIndex • **Name:** series_index
—id	Dataset	Array of unique identifiers for the rows of this dynamic table. • **Extends:** ElementIdentifiers • **Data Type:** int • **Dimensions:** [‘num_rows’] • **Shape:** [None] • **Name:** id
—<VectorData>	Dataset	Vector columns, including index columns, of this dynamic table. • **Extends:** VectorData • **Quantity:** 0 or more

4.9.9 IntracellularElectrodesTable

Overview: Table for storing intracellular electrode related metadata.

`IntracellularElectrodesTable` extends `DynamicTable` and includes all elements of DynamicTable with the following additions or changes.

- **Extends:** DynamicTable
- **Primitive Type:** Group
- **Inherits from:** DynamicTable, Container
- **Source filename:** nwb.icephys.yaml
- **Source Specification:** see Section 5.10.9

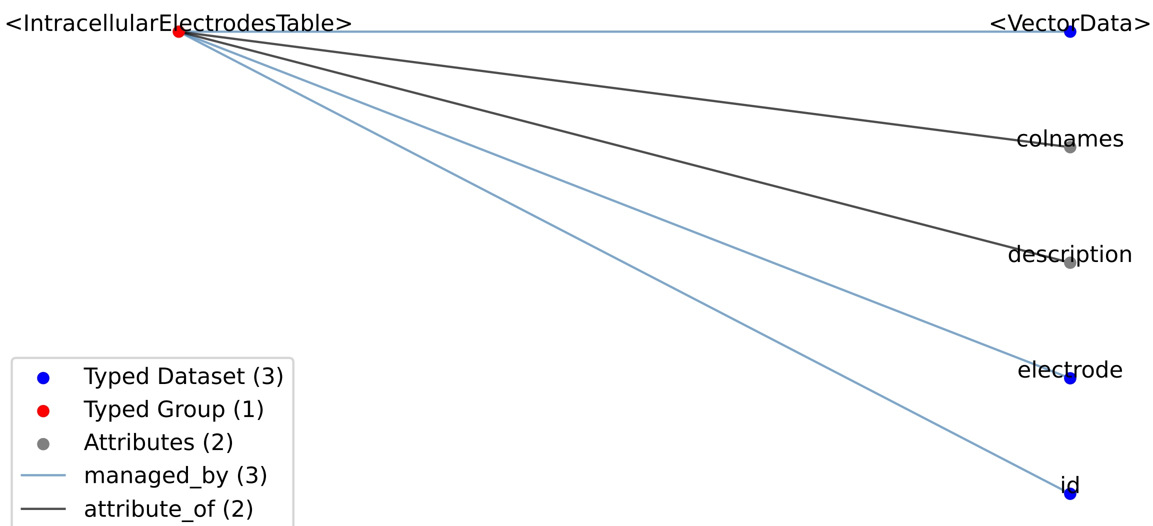


Table 4.92: Datasets, Links, and Attributes contained in <IntracellularElectrodesTable>

Id	Type	Description
<IntracellularElectrodesTable>	Group	Top level Group for <IntracellularElectrodesTable> • **Neurodata Type:** *IntracellularElectrodesTable* • **Extends:** DynamicTable
—description	Attribute	Description of what is in this dynamic table. • **Data Type:** text • **Value:** Table for storing intracellular electrode related metadata. • **Name:** description
—colnames	Attribute	The names of the columns in this table. This should be used to specify an order to the columns. • **Data Type:** text • **Dimensions:** [‘num_columns’] • **Shape:** [None] • **Name:** colnames

continues on next page

Table 4.92 – continued from previous page

Id	Type	Description
—electrode	Dataset	Column for storing the reference to the intracellular electrode. • **Extends:** VectorData • **Data Type:** object reference to *IntracellularElectrode* • **Name:** electrode
—id	Dataset	Array of unique identifiers for the rows of this dynamic table. • **Extends:** ElementIdentifiers • **Data Type:** int • **Dimensions:** ['num_rows'] • **Shape:** [None] • **Name:** id
—<VectorData>	Dataset	Vector columns, including index columns, of this dynamic table. • **Extends:** VectorData • **Quantity:** 0 or more

4.9.10 IntracellularStimuliTable

Overview: Table for storing intracellular stimulus related metadata.

`IntracellularStimuliTable` extends `DynamicTable` and includes all elements of DynamicTable with the following additions or changes.

- **Extends:** DynamicTable
- **Primitive Type:** Group
- **Inherits from:** DynamicTable, Container
- **Source filename:** nwb.icephys.yaml
- **Source Specification:** see Section 5.10.10

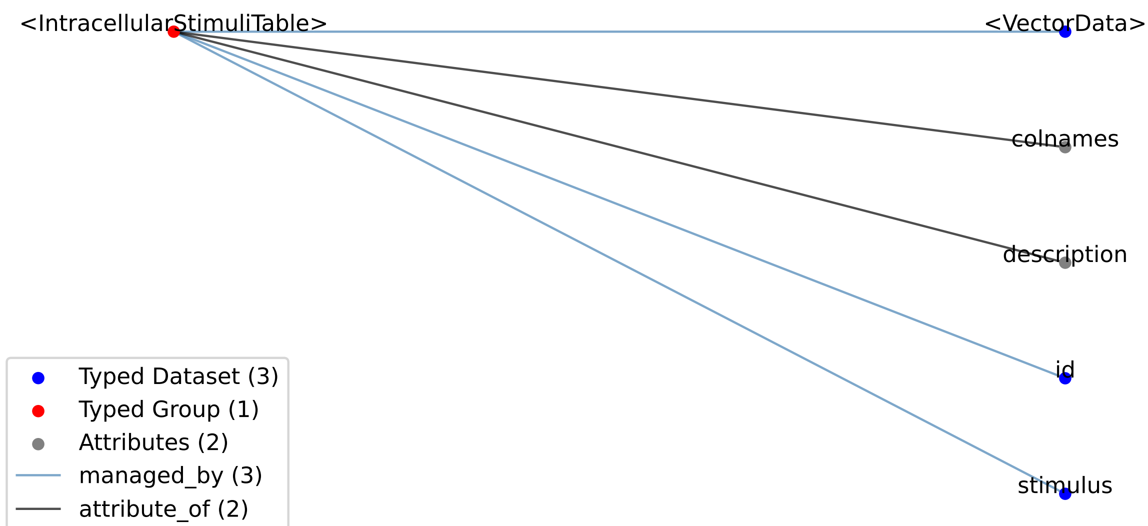


Table 4.93: Datasets, Links, and Attributes contained in <IntracellularStimuliTable>

Id	Type	Description
<IntracellularStimuliTable>	Group	Top level Group for <IntracellularStimuliTable> • **Neurodata Type:** *IntracellularStimuliTable* • **Extends:** DynamicTable
—description	Attribute	Description of what is in this dynamic table. • **Data Type:** text • **Value:** Table for storing intracellular stimulus related metadata. • **Name:** description
—colnames	Attribute	The names of the columns in this table. This should be used to specify an order to the columns. • **Data Type:** text • **Dimensions:** [‘num_columns’] • **Shape:** [None] • **Name:** colnames

continues on next page

Table 4.93 – continued from previous page

Id	Type	Description
—stimulus	Dataset	Column storing the reference to the recorded stimulus for the recording (rows). • **Extends:** *TimeSeriesReferenceVectorData* • **Name:** stimulus
—id	Dataset	Array of unique identifiers for the rows of this dynamic table. • **Extends:** ElementIdentifiers • **Data Type:** int • **Dimensions:** ['num_rows'] • **Shape:** [None] • **Name:** id
—<VectorData>	Dataset	Vector columns, including index columns, of this dynamic table. • **Extends:** VectorData • **Quantity:** 0 or more

4.9.11 IntracellularResponsesTable

Overview: Table for storing intracellular response related metadata.

`IntracellularResponsesTable` extends `DynamicTable` and includes all elements of DynamicTable with the following additions or changes.

- **Extends:** DynamicTable
- **Primitive Type:** Group
- **Inherits from:** DynamicTable, Container
- **Source filename:** nwb.icephys.yaml
- **Source Specification:** see Section 5.10.11

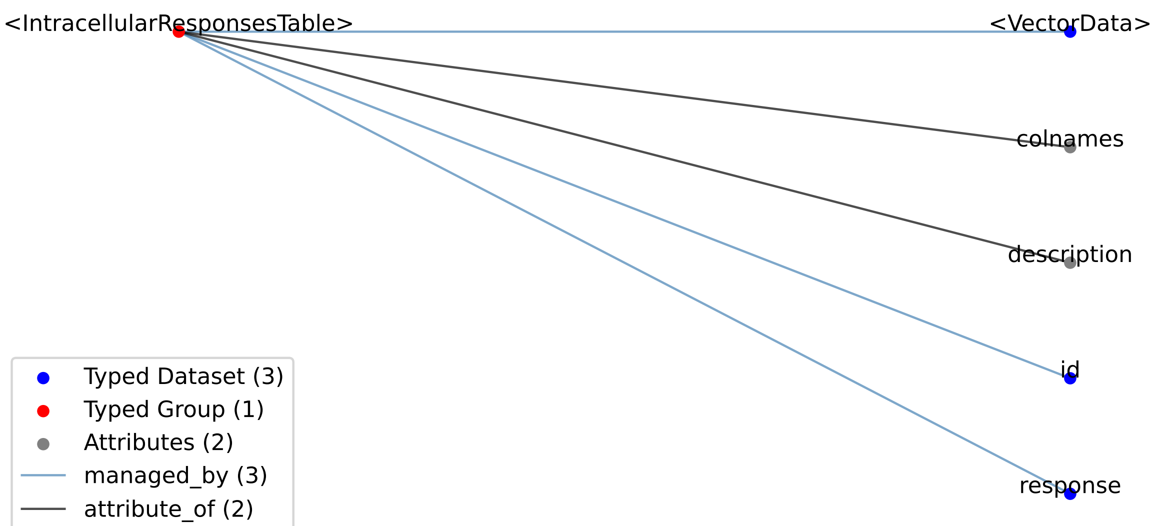


Table 4.94: Datasets, Links, and Attributes contained in <IntracellularResponsesTable>

Id	Type	Description
<IntracellularResponses-Table>	Group	Top level Group for <IntracellularResponsesTable> • **Neurodata Type:** *IntracellularResponsesTable* • **Extends:** DynamicTable
—description	Attribute	Description of what is in this dynamic table. • **Data Type:** text • **Value:** Table for storing intracellular response related metadata. • **Name:** description
—colnames	Attribute	The names of the columns in this table. This should be used to specify an order to the columns. • **Data Type:** text • **Dimensions:** [‘num_columns’] • **Shape:** [None] • **Name:** colnames

continues on next page

Table 4.94 – continued from previous page

Id	Type	Description
—response	Dataset	Column storing the reference to the recorded response for the recording (rows) • **Extends:** *TimeSeriesReferenceVectorData* • **Name:** response
—id	Dataset	Array of unique identifiers for the rows of this dynamic table. • **Extends:** ElementIdentifiers • **Data Type:** int • **Dimensions:** ['num_rows'] • **Shape:** [None] • **Name:** id
—<VectorData>	Dataset	Vector columns, including index columns, of this dynamic table. • **Extends:** VectorData • **Quantity:** 0 or more

4.9.12 IntracellularRecordingsTable

Overview: A table to group together a stimulus and response from a single electrode and a single simultaneous recording. Each row in the table represents a single recording consisting typically of a stimulus and a corresponding response. In some cases, however, only a stimulus or a response is recorded as part of an experiment. In this case, both the stimulus and response will point to the same TimeSeries while the idx_start and count of the invalid column will be set to -1, thus, indicating that no values have been recorded for the stimulus or response, respectively. Note, a recording MUST contain at least a stimulus or a response. Typically the stimulus and response are PatchClampSeries. However, the use of AD/DA channels that are not associated to an electrode is also common in intracellular electrophysiology, in which case other TimeSeries may be used.

`IntracellularRecordingsTable` extends `AlignedDynamicTable` and includes all elements of AlignedDynamicTable with the following additions or changes.

- **Extends:** AlignedDynamicTable
- **Primitive Type:** Group
- **Name:** intracellular_recordings
- **Inherits from:** AlignedDynamicTable, DynamicTable, Container
- **Source filename:** nwb.icephys.yaml
- **Source Specification:** see Section 5.10.12

Table 4.95: Datasets, Links, and Attributes contained in <IntracellularRecordingsTable>

Id	Type	Description
intracellular_recordings	Group	Top level Group for intracellular_recordings • **Neurodata Type:** *IntracellularRecordingsTable* • **Extends:** AlignedDynamicTable • **Name:** intracellular_recordings
—description	Attribute	Description of the contents of this table. Inherited from AlignedDynamicTable and overwritten here to fix the value of the attribute. • **Data Type:** text • **Value:** A table to group together a stimulus and response from a single electrode and a single simultaneous recording and for storing metadata about the intracellular recording. • **Name:** description
—categories	Attribute	The names of the categories in this AlignedDynamicTable. Each category is represented by one DynamicTable stored in the parent group. This attribute should be used to specify an order of categories and the category names must match the names of the corresponding DynamicTable in the group. • **Data Type:** text • **Dimensions:** [‘num_categories’] • **Shape:** [None] • **Name:** categories

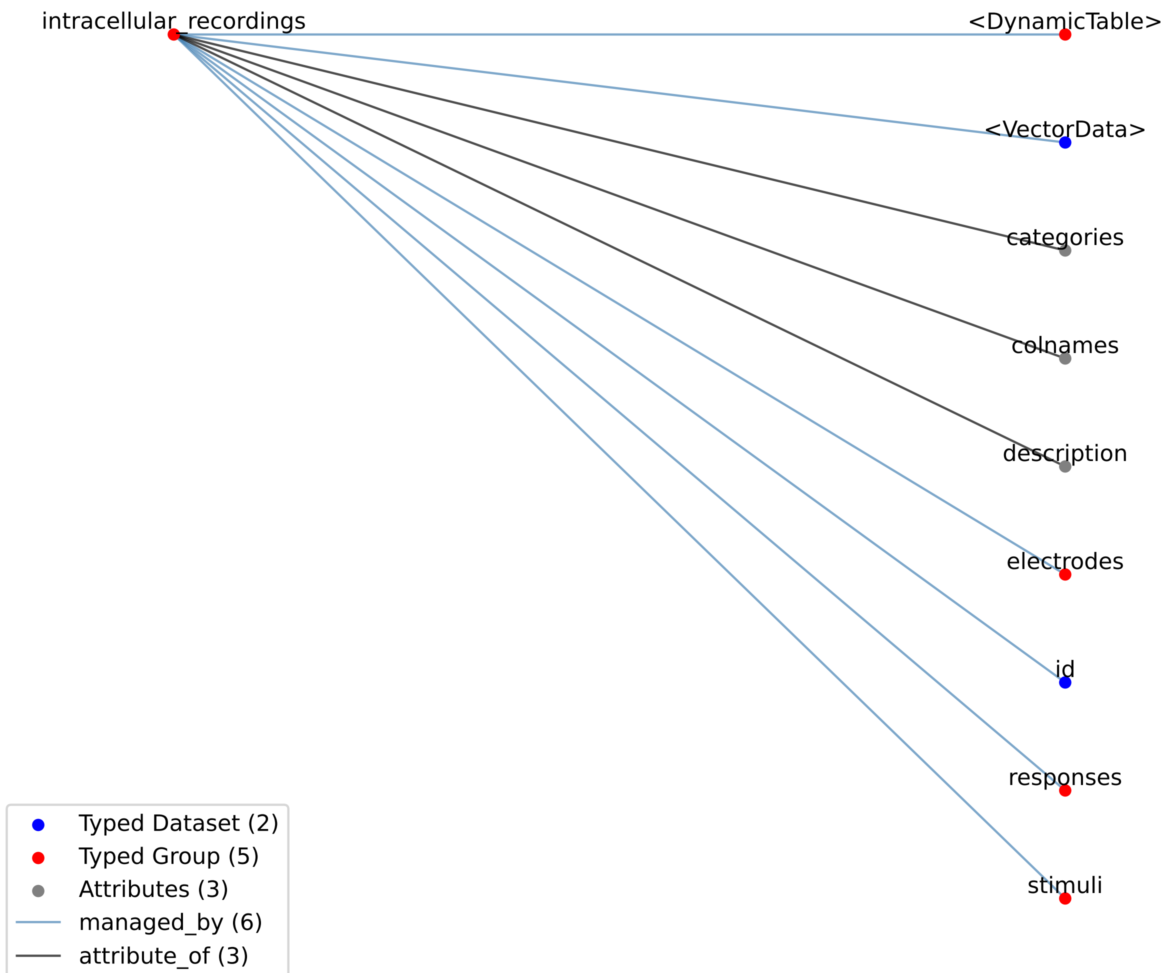
continues on next page

Table 4.95 – continued from previous page

Id	Type	Description
—colnames	Attribute	The names of the columns in this table. This should be used to specify an order to the columns. • **Data Type:** text • **Dimensions:** ['num_columns'] • **Shape:** [None] • **Name:** colnames
—id	Dataset	Array of unique identifiers for the rows of this dynamic table. • **Extends:** ElementIdentifiers • **Data Type:** int • **Dimensions:** ['num_rows'] • **Shape:** [None] • **Name:** id
—<VectorData>	Dataset	Vector columns, including index columns, of this dynamic table. • **Extends:** VectorData • **Quantity:** 0 or more

Table 4.96: Groups contained in <IntracellularRecordingsTable>

Id	Type	Description
intracellular_recordings	Group	Top level Group for intracellular_recordings • **Neurodata Type:** *IntracellularRecordingsTable* • **Extends:** AlignedDynamicTable • **Name:** intracellular_recordings
—electrodes	Group	Table for storing intracellular electrode related metadata. • **Extends:** *IntracellularElectrodesTable* • **Name:** electrodes
—stimuli	Group	Table for storing intracellular stimulus related metadata. • **Extends:** *IntracellularStimuliTable* • **Name:** stimuli
—responses	Group	Table for storing intracellular response related metadata. • **Extends:** *IntracellularResponsesTable* • **Name:** responses
—<DynamicTable>	Group	A DynamicTable representing a particular category for columns in the AlignedDynamicTable parent container. The table MUST be aligned with (i.e., have the same number of rows) as all other DynamicTables stored in the AlignedDynamicTable parent container. The name of the category is given by the name of the DynamicTable and its description by the description attribute of the DynamicTable. • **Extends:** DynamicTable • **Quantity:** 0 or more


intracellular_recordings
<DynamicTable>
<VectorData>
categories
colnames
description
electrodes
id
responses
stimuli
Typed Dataset (2)
Typed Group (5)
Attributes (3)
managed_by (6)
attribute_of (3)

4.9.12.1 Groups: electrodes

Table for storing intracellular electrode related metadata.

- **Extends:** *IntracellularElectrodesTable*
- **Name:** electrodes

4.9.12.2 Groups: stimuli

Table for storing intracellular stimulus related metadata.

- **Extends:** *IntracellularStimuliTable*
- **Name:** stimuli

4.9.12.3 Groups: responses

Table for storing intracellular response related metadata.

- **Extends:** *IntracellularResponsesTable*
- **Name:** responses

4.9.12.4 Groups: <DynamicTable>

A DynamicTable representing a particular category for columns in the AlignedDynamicTable parent container. The table MUST be aligned with (i.e., have the same number of rows) as all other DynamicTables stored in the AlignedDynamicTable parent container. The name of the category is given by the name of the DynamicTable and its description by the description attribute of the DynamicTable.

- **Extends:** DynamicTable
- **Quantity:** 0 or more

4.9.13 SimultaneousRecordingsTable

Overview: A table for grouping different intracellular recordings from the IntracellularRecordingsTable table together that were recorded simultaneously from different electrodes.

`SimultaneousRecordingsTable` extends `DynamicTable` and includes all elements of DynamicTable with the following additions or changes.

- **Extends:** DynamicTable
- **Primitive Type:** Group
- **Name:** simultaneous_recordings
- **Inherits from:** DynamicTable, Container
- **Source filename:** nwb.icephys.yaml
- **Source Specification:** see Section 5.10.13

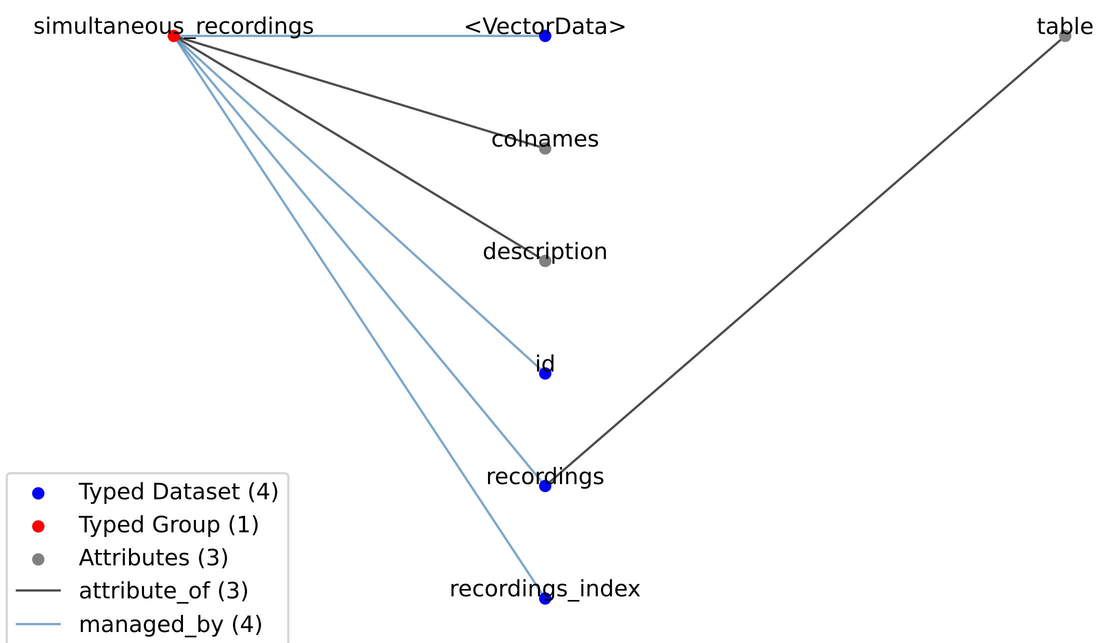


Table 4.97: Datasets, Links, and Attributes contained in <SimultaneousRecordingsTable>

Id	Type	Description
simultaneous_recordings	Group	Top level Group for simultaneous_recordings • **Neurodata Type:** *SimultaneousRecordingsTable* • **Extends:** DynamicTable • **Name:** simultaneous_recordings

continues on next page

Table 4.97 – continued from previous page

Id	Type	Description
—colnames	Attribute	The names of the columns in this table. This should be used to specify an order to the columns. • **Data Type:** text • **Dimensions:** ['num_columns'] • **Shape:** [None] • **Name:** colnames
—description	Attribute	Description of what is in this dynamic table. • **Data Type:** text • **Name:** description
—recordings	Dataset	A reference to one or more rows in the IntracellularRecordingsTable table. • **Extends:** DynamicTableRegion • **Name:** recordings
——table	Attribute	Reference to the IntracellularRecordingsTable table that this table region applies to. This specializes the attribute inherited from DynamicTableRegion to fix the type of table that can be referenced here. • **Data Type:** object reference to *IntracellularRecordingsTable* • **Name:** table
—recordings_index	Dataset	Index dataset for the recordings column. • **Extends:** VectorIndex • **Name:** recordings_index
—id	Dataset	Array of unique identifiers for the rows of this dynamic table. • **Extends:** ElementIdentifiers • **Data Type:** int • **Dimensions:** ['num_rows'] • **Shape:** [None] • **Name:** id
—<VectorData>	Dataset	Vector columns, including index columns, of this dynamic table. • **Extends:** VectorData • **Quantity:** 0 or more

4.9.14 SequentialRecordingsTable

Overview: A table for grouping different sequential recordings from the SimultaneousRecordingsTable table together. This is typically used to group together sequential recordings where a sequence of stimuli of the same type with varying parameters have been presented in a sequence.

`SequentialRecordingsTable` extends `DynamicTable` and includes all elements of DynamicTable with the following additions or changes.

- **Extends:** DynamicTable
- **Primitive Type:** Group
- **Name:** sequential_recordings
- **Inherits from:** DynamicTable, Container
- **Source filename:** nwb.icephys.yaml
- **Source Specification:** see Section 5.10.14

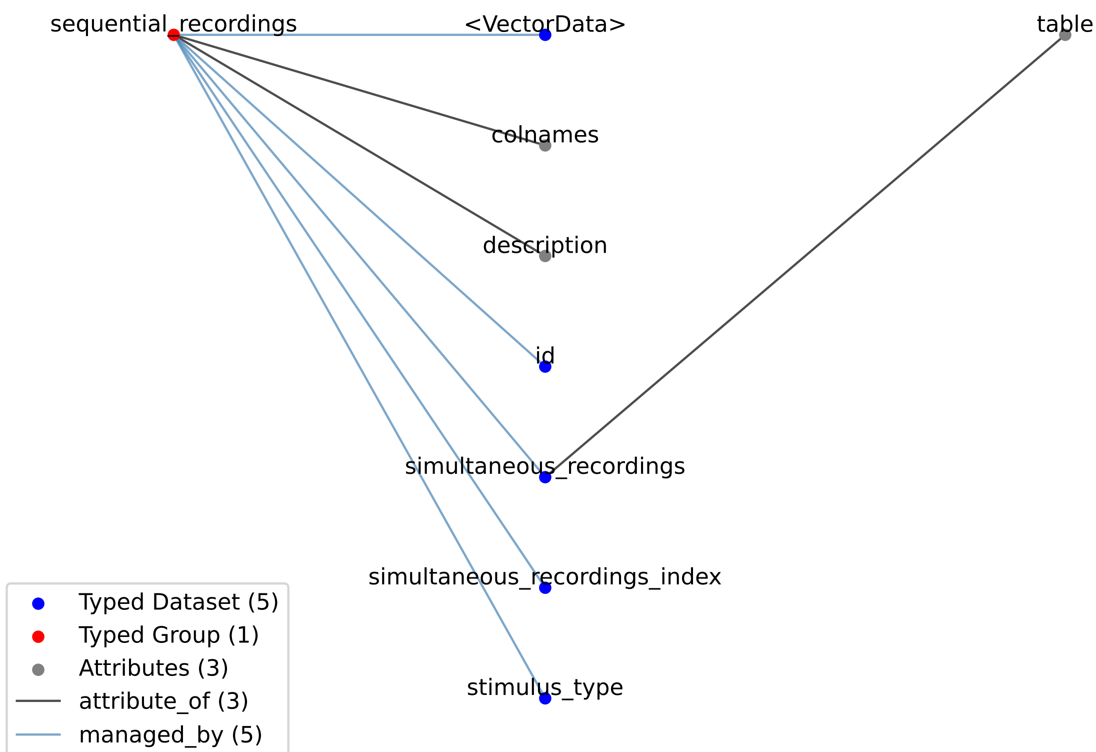


Table 4.98: Datasets, Links, and Attributes contained in <SequentialRecordingsTable>

Id	Type	Description
sequential_recordings	Group	Top level Group for sequential_recordings • **Neurodata Type:** *SequentialRecordingsTable* • **Extends:** DynamicTable • **Name:** sequential_recordings
—colnames	Attribute	The names of the columns in this table. This should be used to specify an order to the columns. • **Data Type:** text • **Dimensions:** ['num_columns'] • **Shape:** [None] • **Name:** colnames
—description	Attribute	Description of what is in this dynamic table. • **Data Type:** text • **Name:** description
—simultaneous_recordings	Dataset	A reference to one or more rows in the SimultaneousRecordingsTable table. • **Extends:** DynamicTableRegion • **Name:** simultaneous_recordings
——table	Attribute	Reference to the SimultaneousRecordingsTable table that this table region applies to. This specializes the attribute inherited from DynamicTableRegion to fix the type of table that can be referenced here. • **Data Type:** object reference to *SimultaneousRecordingsTable* • **Name:** table
—simultaneous_recordings_index	Dataset	Index dataset for the simultaneous_recordings column. • **Extends:** VectorIndex • **Name:** simultaneous_recordings_index
—stimulus_type	Dataset	The type of stimulus used for the sequential recording. • **Extends:** VectorData • **Data Type:** text • **Name:** stimulus_type
—id	Dataset	Array of unique identifiers for the rows of this dynamic table. • **Extends:** ElementIdentifiers • **Data Type:** int • **Dimensions:** ['num_rows'] • **Shape:** [None] • **Name:** id
—<VectorData>	Dataset	Vector columns, including index columns, of this dynamic table. • **Extends:** VectorData • **Quantity:** 0 or more

4.9.15 RepetitionsTable

Overview: A table for grouping different sequential intracellular recordings together. With each SequentialRecording typically representing a particular type of stimulus, the RepetitionsTable table is typically used to group sets of stimuli applied in sequence.

`RepetitionsTable` extends `DynamicTable` and includes all elements of DynamicTable with the following additions or changes.

- **Extends:** DynamicTable
- **Primitive Type:** Group
- **Name:** repetitions
- **Inherits from:** DynamicTable, Container
- **Source filename:** nwb.icephys.yaml
- **Source Specification:** see Section 5.10.15

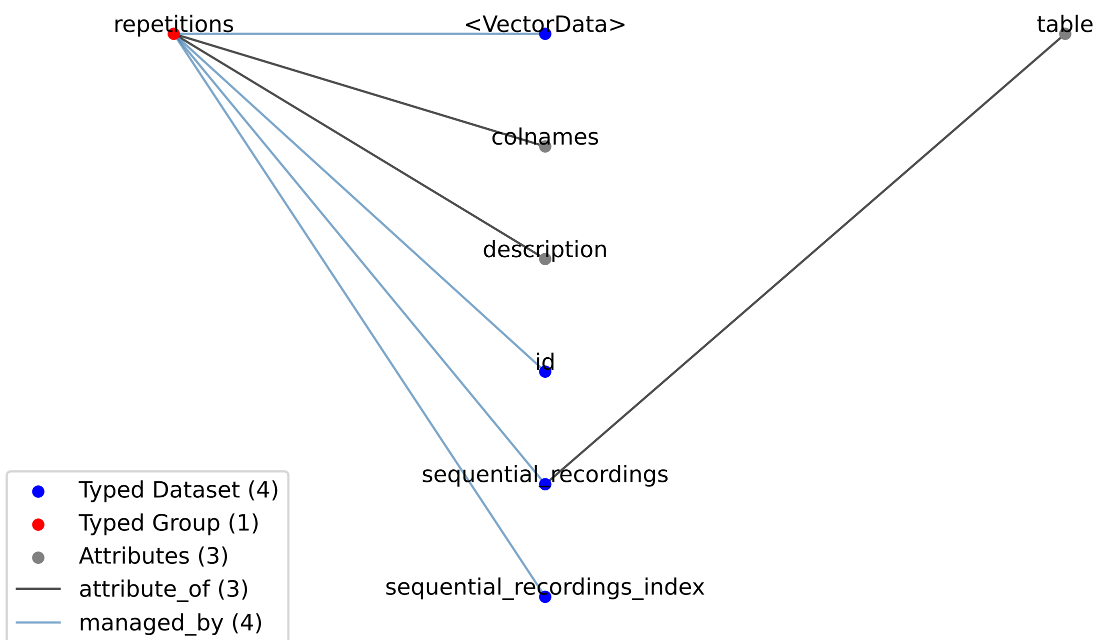


Table 4.99: Datasets, Links, and Attributes contained in <RepetitionsTable>

Id	Type	Description
repetitions	Group	Top level Group for repetitions • **Neurodata Type:** *RepetitionsTable* • **Extends:** DynamicTable • **Name:** repetitions

continues on next page

Table 4.99 – continued from previous page

Id	Type	Description
—colnames	Attribute	The names of the columns in this table. This should be used to specify an order to the columns. • **Data Type:** text • **Dimensions:** ['num_columns'] • **Shape:** [None] • **Name:** colnames
—description	Attribute	Description of what is in this dynamic table. • **Data Type:** text • **Name:** description
—sequential_recordings	Dataset	A reference to one or more rows in the SequentialRecordingsTable table. • **Extends:** DynamicTableRegion • **Name:** sequential_recordings
——table	Attribute	Reference to the SequentialRecordingsTable table that this table region applies to. This specializes the attribute inherited from DynamicTableRegion to fix the type of table that can be referenced here. • **Data Type:** object reference to *SequentialRecordingsTable* • **Name:** table
—sequential_recordings_index	Dataset	Index dataset for the sequential_recordings column. • **Extends:** VectorIndex • **Name:** sequential_recordings_index
—id	Dataset	Array of unique identifiers for the rows of this dynamic table. • **Extends:** ElementIdentifiers • **Data Type:** int • **Dimensions:** ['num_rows'] • **Shape:** [None] • **Name:** id
—<VectorData>	Dataset	Vector columns, including index columns, of this dynamic table. • **Extends:** VectorData • **Quantity:** 0 or more

4.9.16 ExperimentalConditionsTable

Overview: A table for grouping different intracellular recording repetitions together that belong to the same experimental condition.

`ExperimentalConditionsTable` extends `DynamicTable` and includes all elements of DynamicTable with the following additions or changes.

- **Extends:** DynamicTable
- **Primitive Type:** Group
- **Name:** experimental_conditions
- **Inherits from:** DynamicTable, Container
- **Source filename:** nwb.icephys.yaml
- **Source Specification:** see Section 5.10.16

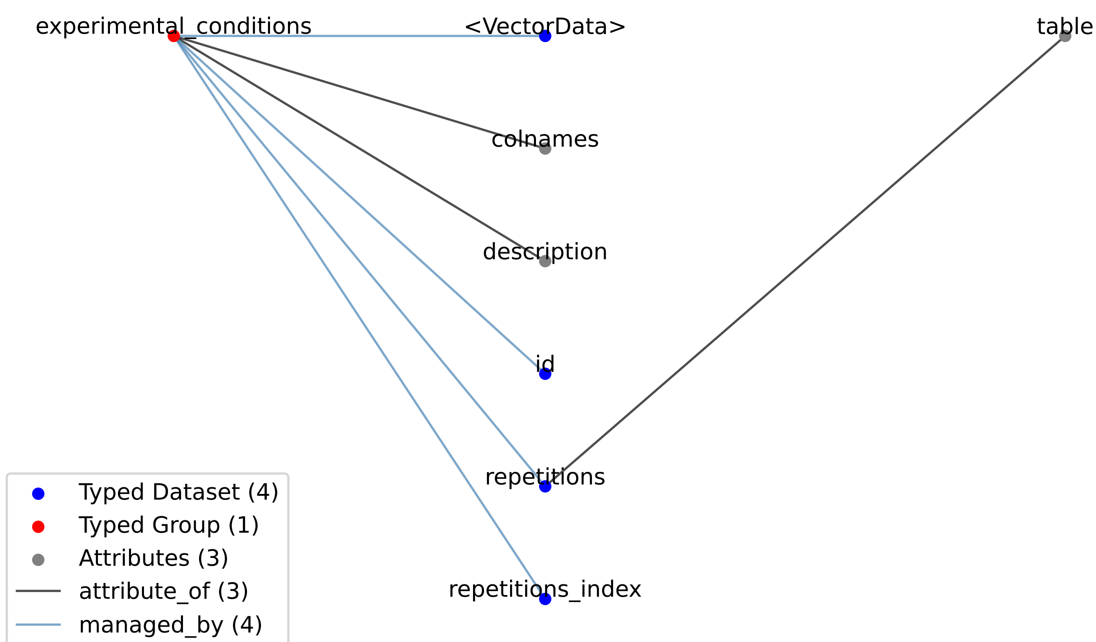


Table 4.100: Datasets, Links, and Attributes contained in <ExperimentalConditionsTable>

Id	Type	Description
experimental_conditions	Group	Top level Group for experimental_conditions • **Neurodata Type:** *ExperimentalConditionsTable* • **Extends:** DynamicTable • **Name:** experimental_conditions

continues on next page

Table 4.100 – continued from previous page

Id	Type	Description
—colnames	Attribute	The names of the columns in this table. This should be used to specify an order to the columns. • **Data Type:** text • **Dimensions:** ['num_columns'] • **Shape:** [None] • **Name:** colnames
—description	Attribute	Description of what is in this dynamic table. • **Data Type:** text • **Name:** description
—repetitions	Dataset	A reference to one or more rows in the RepetitionsTable table. • **Extends:** DynamicTableRegion • **Name:** repetitions
——table	Attribute	Reference to the RepetitionsTable table that this table region applies to. This specializes the attribute inherited from DynamicTableRegion to fix the type of table that can be referenced here. • **Data Type:** object reference to *RepetitionsTable* • **Name:** table
—repetitions_index	Dataset	Index dataset for the repetitions column. • **Extends:** VectorIndex • **Name:** repetitions_index
—id	Dataset	Array of unique identifiers for the rows of this dynamic table. • **Extends:** ElementIdentifiers • **Data Type:** int • **Dimensions:** ['num_rows'] • **Shape:** [None] • **Name:** id
—<VectorData>	Dataset	Vector columns, including index columns, of this dynamic table. • **Extends:** VectorData • **Quantity:** 0 or more

4.10 Optogenetics

This source module contains neurodata_types for opto-genetics data.

4.10.1 OptogeneticSeries

Overview: An optogenetic stimulus.

`OptogeneticSeries` extends `TimeSeries` and includes all elements of *TimeSeries* with the following additions or changes.

- **Extends:** *TimeSeries*
- **Primitive Type:** Group
- **Inherits from:** *TimeSeries*, *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.ogen.yaml
- **Source Specification:** see Section 5.11.1

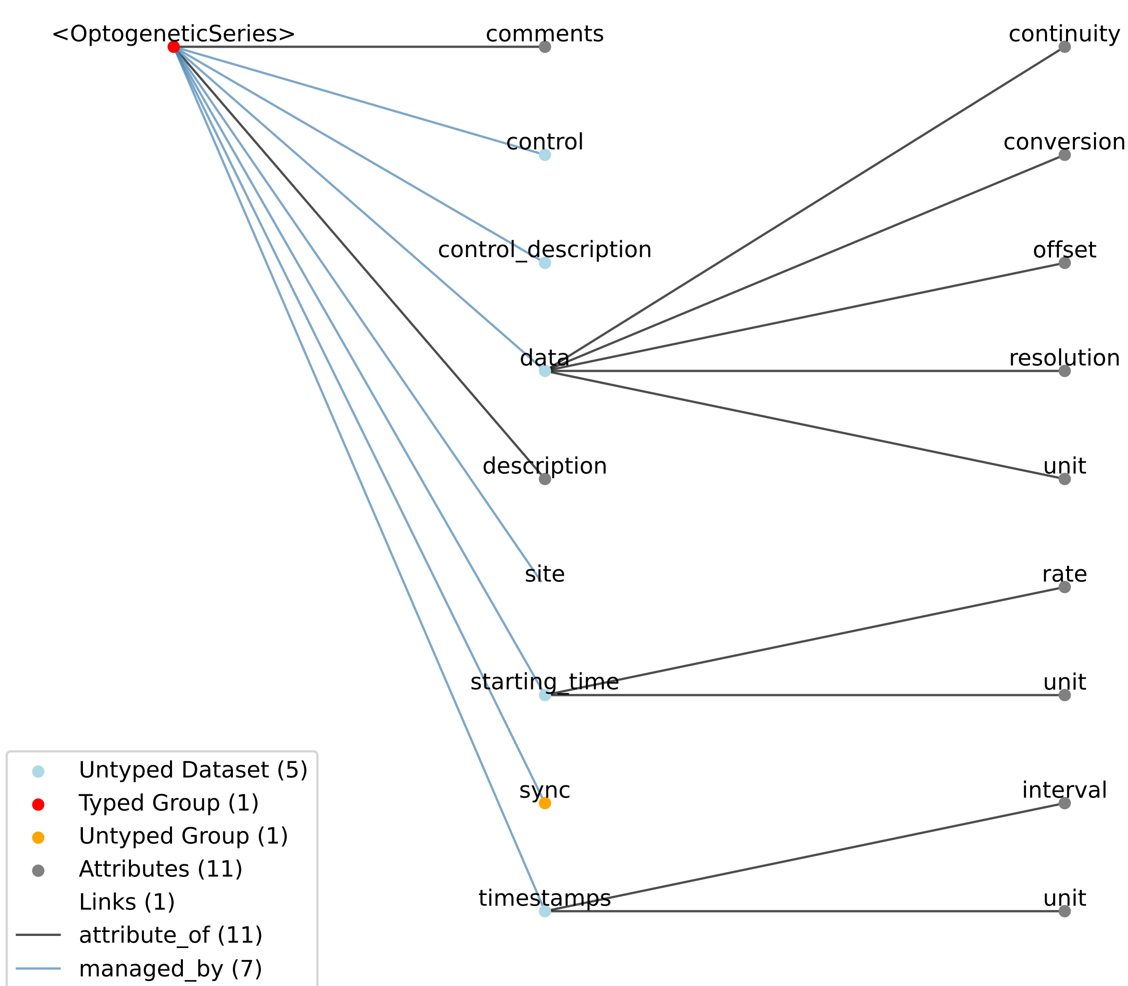


Table 4.101: Datasets, Links, and Attributes contained in <OptogeneticSeries>

Id	Type	Description
<OptogeneticSeries>	Group	Top level Group for <OptogeneticSeries> • **Neurodata Type:** *OptogeneticSeries* • **Extends:** *TimeSeries*
—description	Attribute	Description of the time series. • **Data Type:** text • **Required:** False • **Default Value:** no description • **Name:** description
—comments	Attribute	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. • **Data Type:** text • **Required:** False • **Default Value:** no comments • **Name:** comments
—data	Dataset	Applied power for optogenetic stimulus, in watts. • **Data Type:** numeric • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** data
——unit	Attribute	Unit of measurement for data, which is fixed to 'watts'. • **Data Type:** text • **Value:** watts • **Name:** unit
——conversion	Attribute	Scalar to multiply each element in data to convert it to the specified 'unit'. If the data are stored in acquisition system units or other units that require a conversion to be interpretable, multiply the data by 'conversion' to convert the data to the specified 'unit'. e.g. if the data acquisition system stores values in this object as signed 16-bit integers (int16 range -32,768 to 32,767) that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition values to recorded volts is 2.5/32768/8000 = 9.5367e-9. • **Data Type:** float32 • **Required:** False • **Default Value:** 1.0 • **Name:** conversion

continues on next page

Table 4.101 – continued from previous page

Id	Type	Description
——offset	Attribute	Scalar to add to the data after scaling by ‘conversion’ to finalize its coercion to the specified ‘unit’. Two common examples of this include (a) data stored in an unsigned type that requires a shift after scaling to re-center the data, and (b) specialized recording devices that naturally cause a scalar offset with respect to the true units. • **Data Type:** float32 • **Required:** False • **Default Value:** 0.0 • **Name:** offset
——resolution	Attribute	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 -1.0. • **Data Type:** float32 • **Required:** False • **Default Value:** -1.0 • **Name:** resolution
——continuity	Attribute	Optionally describe the continuity of the data. Can be “continuous”, “instantaneous”, or “step”. For example, a voltage trace would be “continuous”, because samples are recorded from a continuous process. An array of lick times would be “instantaneous”, because the data represents distinct moments in time. Times of image presentations would be “step” because the picture remains the same until the next timepoint. This field is optional, but is useful in providing information about the underlying data. It may inform the way this data is interpreted, the way it is visualized, and what analysis methods are applicable. • **Data Type:** text • **Required:** False • **Name:** continuity
—starting_time	Dataset	Timestamp of the first sample in seconds. When timestamps are uniformly spaced, the timestamp of the first sample can be specified and all subsequent ones calculated from the sampling rate attribute. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Name:** starting_time
——rate	Attribute	Sampling rate, in Hz. • **Data Type:** float32 • **Name:** rate
——unit	Attribute	Unit of measurement for time, which is fixed to ‘seconds’. • **Data Type:** text • **Value:** seconds • **Name:** unit

continues on next page

Table 4.101 – continued from previous page

Id	Type	Description
—timestamps	Dataset	Timestamps for samples stored in data, in seconds, relative to the common experiment master-clock stored in NWBFile.timestamps_reference_time. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** timestamps
——interval	Attribute	Value is '1' • **Data Type:** int32 • **Value:** 1 • **Name:** interval
——unit	Attribute	Unit of measurement for timestamps, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—control	Dataset	Numerical labels that apply to each time point in data for the purpose of querying and slicing data by these values. If present, the length of this array should be the same size as the first dimension of data. • **Quantity:** 0 or 1 • **Data Type:** uint8 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** control
—control_description	Dataset	Description of each control value. Must be present if control is present. If present, control_description[0] should describe time points where control == 0. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** ['num_control_values'] • **Shape:** [None] • **Name:** control_description
—site	Link	Link to OptogeneticStimulusSite object that describes the site to which this stimulus was applied. • **Target Type** *OptogeneticStimulusSite* • **Name:** site

Table 4.102: Groups contained in <OptogeneticSeries>

Id	Type	Description
<OptogeneticSeries>	Group	Top level Group for <OptogeneticSeries> • **Neurodata Type:** *OptogeneticSeries* • **Extends:** *TimeSeries*
—site	Link	Link to OptogeneticStimulusSite object that describes the site to which this stimulus was applied. • **Target Type** *OptogeneticStimulusSite* • **Name:** site
—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. • **Quantity:** 0 or 1 • **Name:** sync

4.10.1.1 Groups: sync

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.

- **Quantity:** 0 or 1
- **Name:** sync

4.10.2 OptogeneticStimulusSite

Overview: A site of optogenetic stimulation.

`OptogeneticStimulusSite` extends `NWBContainer` and includes all elements of *NWBContainer* with the following additions or changes.

- **Extends:** *NWBContainer*
- **Primitive Type:** Group
- **Inherits from:** *NWBContainer*, Container
- **Source filename:** nwb.ogen.yaml
- **Source Specification:** see Section 5.11.2

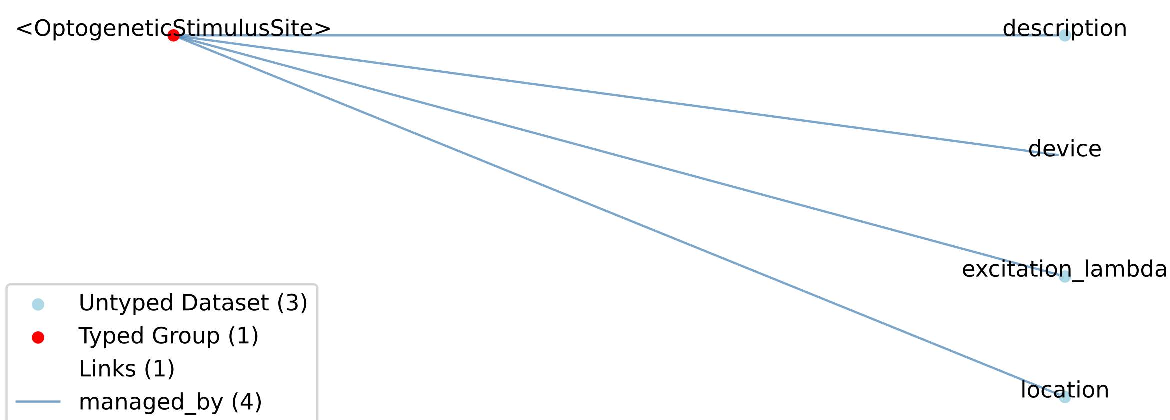


Table 4.103: Datasets, Links, and Attributes contained in <OptogeneticStimulusSite>

Id	Type	Description
<OptogeneticStimulusSite>	Group	Top level Group for <OptogeneticStimulusSite> • **Neurodata Type:** *OptogeneticStimulusSite* • **Extends:** *NWBContainer*
—description	Dataset	Description of stimulation site. • **Data Type:** text • **Name:** description
—excitation_lambda	Dataset	Excitation wavelength, in nm. • **Data Type:** float32 • **Name:** excitation_lambda
—location	Dataset	Location of the stimulation site. Specify the area, layer, comments on estimation of area/layer, stereotaxic coordinates if in vivo, etc. Use standard atlas names for anatomical regions when possible. • **Data Type:** text • **Name:** location

continues on next page

Table 4.103 – continued from previous page

Id	Type	Description
—device	Link	Device that generated the stimulus. • **Target Type** *Device* • **Name:** device

Table 4.104: Groups contained in <OptogeneticStimulusSite>

Id	Type	Description
<OptogeneticStimulusSite>	Group	Top level Group for <OptogeneticStimulusSite> • **Neurodata Type:** *OptogeneticStimulusSite* • **Extends:** *NWBContainer*
—device	Link	Device that generated the stimulus. • **Target Type** *Device* • **Name:** device

4.11 Optical physiology

This source module contains neurodata_types for optical physiology data.

4.11.1 OnePhotonSeries

Overview: Image stack recorded over time from 1-photon microscope.

`OnePhotonSeries` extends `ImageSeries` and includes all elements of *ImageSeries* with the following additions or changes.

- **Extends:** *ImageSeries*
- **Primitive Type:** Group
- **Inherits from:** *ImageSeries*, *TimeSeries*, *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.ophys.yaml
- **Source Specification:** see Section 5.12.1

Table 4.105: Datasets, Links, and Attributes contained in <OnePhotonSeries>

Id	Type	Description
<OnePhotonSeries>	Group	Top level Group for <OnePhotonSeries> • **Neurodata Type:** *OnePhotonSeries* • **Extends:** *ImageSeries*
—pmt_gain	Attribute	Photomultiplier gain. • **Data Type:** float32 • **Required:** False • **Name:** pmt_gain
—scan_line_rate	Attribute	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. • **Data Type:** float32 • **Required:** False • **Name:** scan_line_rate
—exposure_time	Attribute	Exposure time of the sample; often the inverse of the frequency. • **Data Type:** float32 • **Required:** False • **Name:** exposure_time
—binning	Attribute	Amount of pixels combined into 'bins'; could be 1, 2, 4, 8, etc. • **Data Type:** uint8 • **Required:** False • **Name:** binning

continues on next page

Table 4.105 – continued from previous page

Id	Type	Description
—power	Attribute	Power of the excitation in mW, if known. • **Data Type:** float32 • **Required:** False • **Name:** power
—intensity	Attribute	Intensity of the excitation in mW/mm^2, if known. • **Data Type:** float32 • **Required:** False • **Name:** intensity
—description	Attribute	Description of the time series. • **Data Type:** text • **Required:** False • **Default Value:** no description • **Name:** description
—comments	Attribute	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. • **Data Type:** text • **Required:** False • **Default Value:** no comments • **Name:** comments
—data	Dataset	Binary data representing images across frames. If data are stored in an external file, this should be an empty 3D array. • **Data Type:** numeric • **Dimensions:** [['frame', 'x', 'y'], ['frame', 'x', 'y', 'z']] • **Shape:** [[None, None, None], [None, None, None, None]] • **Name:** data
——conversion	Attribute	Scalar to multiply each element in data to convert it to the specified 'unit'. If the data are stored in acquisition system units or other units that require a conversion to be interpretable, multiply the data by 'conversion' to convert the data to the specified 'unit'. e.g. if the data acquisition system stores values in this object as signed 16-bit integers (int16 range -32,768 to 32,767) that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition values to recorded volts is 2.5/32768/8000 = 9.5367e-9. • **Data Type:** float32 • **Required:** False • **Default Value:** 1.0 • **Name:** conversion

continues on next page

Table 4.105 – continued from previous page

Id	Type	Description
——offset	Attribute	Scalar to add to the data after scaling by 'conversion' to finalize its coercion to the specified 'unit'. Two common examples of this include (a) data stored in an unsigned type that requires a shift after scaling to re-center the data, and (b) specialized recording devices that naturally cause a scalar offset with respect to the true units. • **Data Type:** float32 • **Required:** False • **Default Value:** 0.0 • **Name:** offset
——resolution	Attribute	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 -1.0. • **Data Type:** float32 • **Required:** False • **Default Value:** -1.0 • **Name:** resolution
——unit	Attribute	Base unit of measurement for working with the data. Actual stored values are not necessarily stored in these units. To access the data in these units, multiply 'data' by 'conversion' and add 'offset'. • **Data Type:** text • **Name:** unit
——continuity	Attribute	Optionally describe the continuity of the data. Can be "continuous", "instantaneous", or "step". For example, a voltage trace would be "continuous", because samples are recorded from a continuous process. An array of lick times would be "instantaneous", because the data represents distinct moments in time. Times of image presentations would be "step" because the picture remains the same until the next timepoint. This field is optional, but is useful in providing information about the underlying data. It may inform the way this data is interpreted, the way it is visualized, and what analysis methods are applicable. • **Data Type:** text • **Required:** False • **Name:** continuity
—dimension	Dataset	Number of pixels on x, y, (and z) axes. • **Quantity:** 0 or 1 • **Data Type:** int32 • **Dimensions:** ['rank'] • **Shape:** [None] • **Name:** dimension

continues on next page

Table 4.105 – continued from previous page

Id	Type	Description
—external_file	Dataset	Paths to one or more external file(s). The field is only present if format='external'. This is only relevant if the image series 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 NWB file and that file is linked to this file. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** ['num_files'] • **Shape:** [None] • **Name:** external_file
——starting_frame	Attribute	Each external image may contain one or more consecutive frames of the full ImageSeries. This attribute serves as an index to indicate which frames each file contains, to faciliate random access. The 'starting_frame' attribute, hence, contains a list of frame numbers within the full ImageSeries of the first frame of each file listed in the parent 'external_file' dataset. Zero-based indexing is used (hence, the first element will always be zero). For example, if the 'external_file' dataset has three paths to files and the first file has 5 frames, the second file has 10 frames, and the third file has 20 frames, then this attribute will have values [0, 5, 15]. If there is a single external file that holds all of the frames of the ImageSeries (and so there is a single element in the 'external_file' dataset), then this attribute should have value [0]. • **Data Type:** int32 • **Dimensions:** ['num_files'] • **Shape:** [None] • **Name:** starting_frame
—format	Dataset	Format of image. If this is 'external', then the attribute 'external_file' contains the path information to the image files. If this is 'raw', then the raw (single-channel) binary data is stored in the 'data' dataset. If this attribute is not present, then the default format='raw' case is assumed. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** format
—starting_time	Dataset	Timestamp of the first sample in seconds. When timestamps are uniformly spaced, the timestamp of the first sample can be specified and all subsequent ones calculated from the sampling rate attribute. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Name:** starting_time
——rate	Attribute	Sampling rate, in Hz. • **Data Type:** float32 • **Name:** rate

continues on next page

Table 4.105 – continued from previous page

Id	Type	Description
——unit	Attribute	Unit of measurement for time, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—timestamps	Dataset	Timestamps for samples stored in data, in seconds, relative to the common experiment master-clock stored in NWBFile.timestamps_reference_time. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** timestamps
——interval	Attribute	Value is '1' • **Data Type:** int32 • **Value:** 1 • **Name:** interval
——unit	Attribute	Unit of measurement for timestamps, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—control	Dataset	Numerical labels that apply to each time point in data for the purpose of querying and slicing data by these values. If present, the length of this array should be the same size as the first dimension of data. • **Quantity:** 0 or 1 • **Data Type:** uint8 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** control
—control_description	Dataset	Description of each control value. Must be present if control is present. If present, control_description[0] should describe time points where control == 0. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** ['num_control_values'] • **Shape:** [None] • **Name:** control_description
—imaging_plane	Link	Link to ImagingPlane object from which this TimeSeries data was generated. • **Target Type** *ImagingPlane* • **Name:** imaging_plane

continues on next page

Table 4.105 – continued from previous page

Id	Type	Description
—device	Link	Link to the Device object that was used to capture these images. • **Target Type** *Device* • **Name:** device

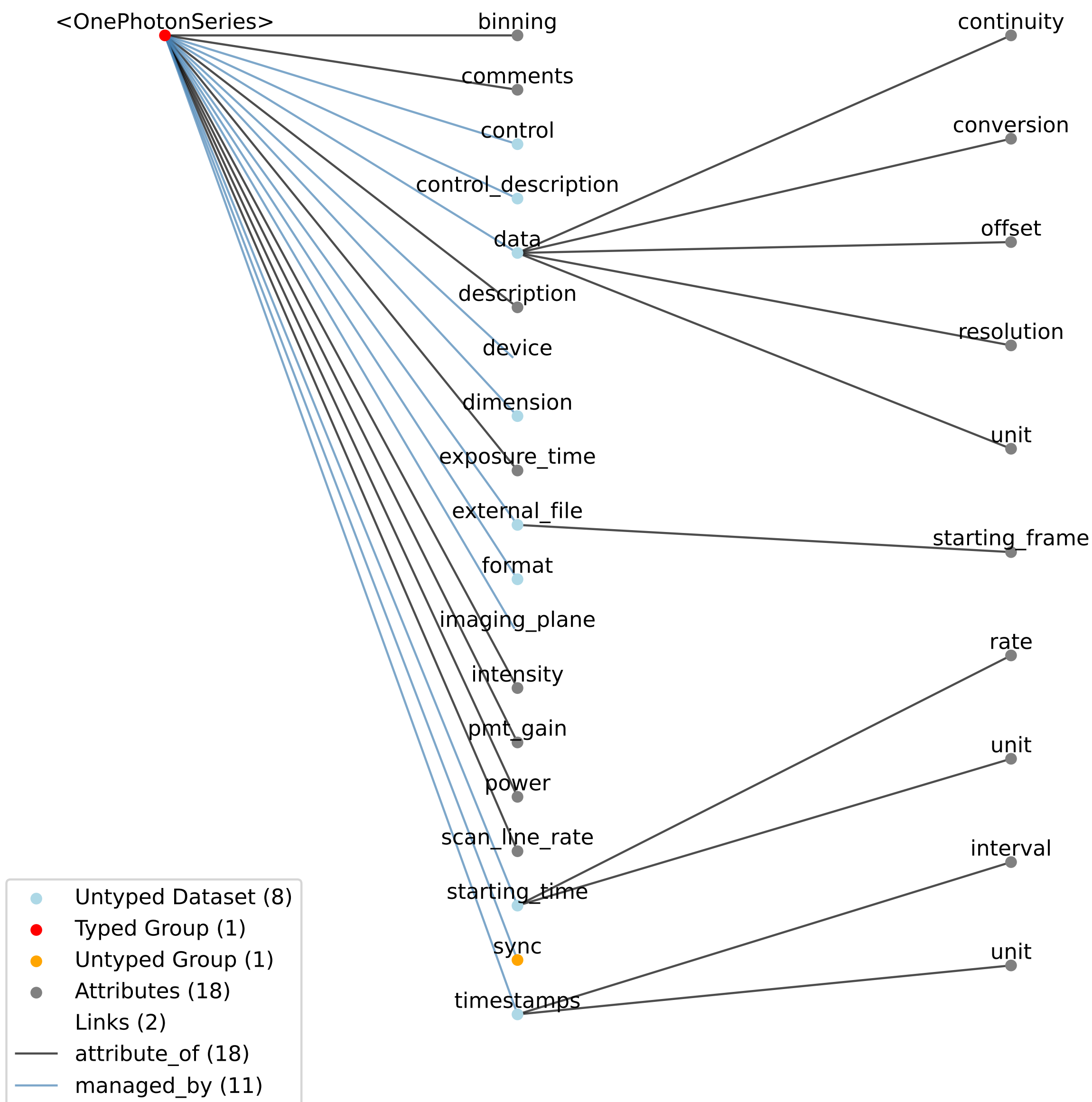
Table 4.106: Groups contained in <OnePhotonSeries>

Id	Type	Description
<OnePhotonSeries>	Group	Top level Group for <OnePhotonSeries> • **Neurodata Type:** *OnePhotonSeries* • **Extends:** *ImageSeries*
—imaging_plane	Link	Link to ImagingPlane object from which this TimeSeries data was generated. • **Target Type** *ImagingPlane* • **Name:** imaging_plane
—device	Link	Link to the Device object that was used to capture these images. • **Target Type** *Device* • **Name:** device
—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. • **Quantity:** 0 or 1 • **Name:** sync

4.11.1.1 Groups: sync

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.

- **Quantity:** 0 or 1
- **Name:** sync


<OnePhotonSeries>
binning
comments
control
control_description
data
description
device
dimension
exposure_time
external_file
format
imaging_plane
intensity
pmt_gain
power
scan_line_rate
starting_time
sync
timestamps
continuity
conversion
offset
resolution
unit
starting_frame
rate
unit
interval
unit
Untyped Dataset (8)
Typed Group (1)
Untyped Group (1)
Attributes (18)
Links (2)
attribute_of (18)
managed_by (11)

4.11.2 TwoPhotonSeries

Overview: Image stack recorded over time from 2-photon microscope.

`TwoPhotonSeries` extends `ImageSeries` and includes all elements of *ImageSeries* with the following additions or changes.

- **Extends:** *ImageSeries*
- **Primitive Type:** Group
- **Inherits from:** *ImageSeries*, *TimeSeries*, *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.ophys.yaml
- **Source Specification:** see Section 5.12.2

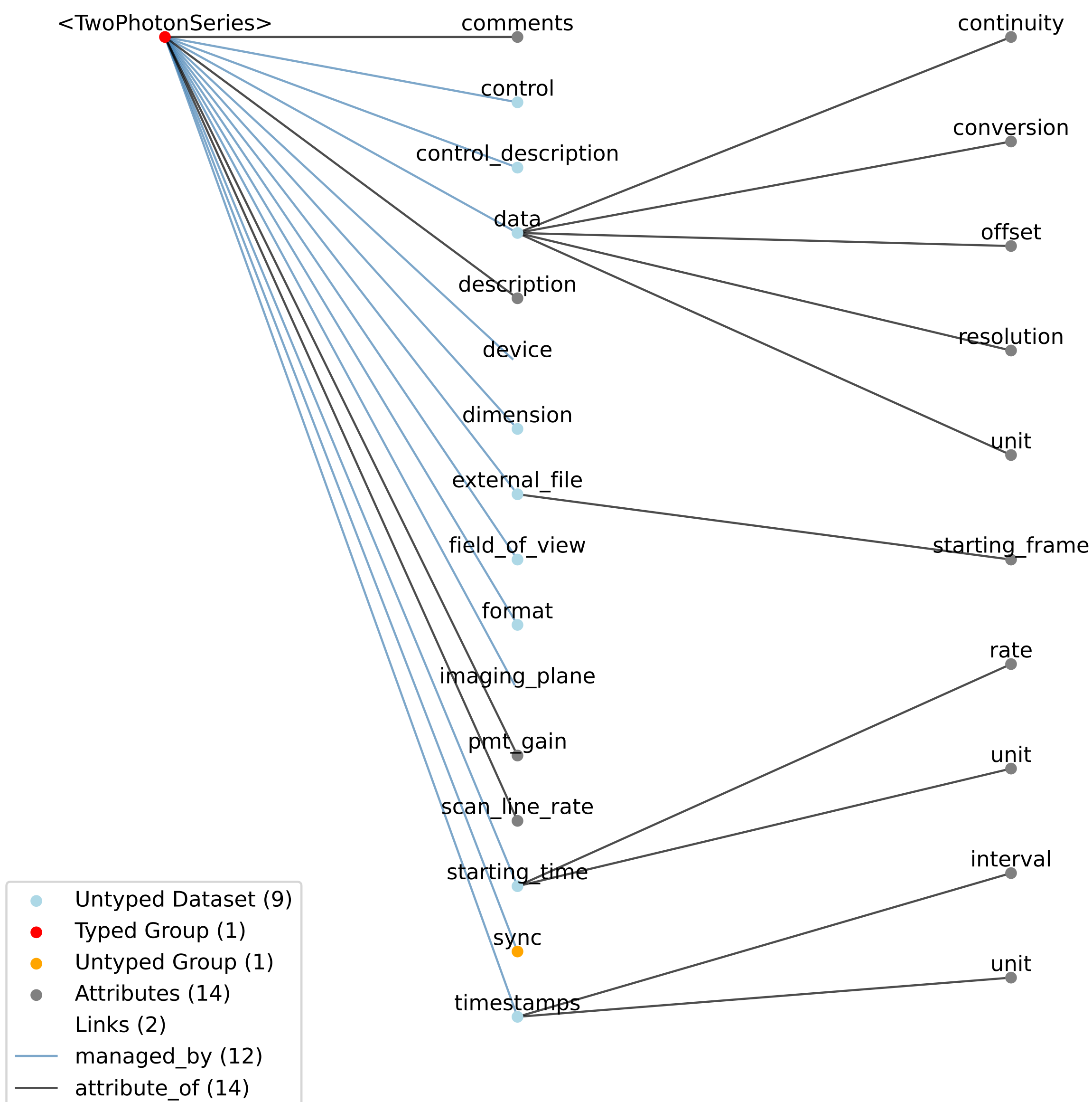


Table 4.107: Datasets, Links, and Attributes contained in <TwoPhotonSeries>

Id	Type	Description
<TwoPhotonSeries>	Group	Top level Group for <TwoPhotonSeries> • **Neurodata Type:** *TwoPhotonSeries* • **Extends:** *ImageSeries*
—pmt_gain	Attribute	Photomultiplier gain. • **Data Type:** float32 • **Required:** False • **Name:** pmt_gain
—scan_line_rate	Attribute	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. • **Data Type:** float32 • **Required:** False • **Name:** scan_line_rate
—description	Attribute	Description of the time series. • **Data Type:** text • **Required:** False • **Default Value:** no description • **Name:** description
—comments	Attribute	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. • **Data Type:** text • **Required:** False • **Default Value:** no comments • **Name:** comments
—field_of_view	Dataset	Width, height and depth of image, or imaged area, in meters. • **Quantity:** 0 or 1 • **Data Type:** float32 • **Dimensions:** [['width|height'], ['width|height|depth']] • **Shape:** [[2], [3]] • **Name:** field_of_view
—data	Dataset	Binary data representing images across frames. If data are stored in an external file, this should be an empty 3D array. • **Data Type:** numeric • **Dimensions:** [['frame', 'x', 'y'], ['frame', 'x', 'y', 'z']] • **Shape:** [[None, None, None], [None, None, None, None]] • **Name:** data

continues on next page

Table 4.107 – continued from previous page

Id	Type	Description
——conversion	Attribute	Scalar to multiply each element in data to convert it to the specified 'unit'. If the data are stored in acquisition system units or other units that require a conversion to be interpretable, multiply the data by 'conversion' to convert the data to the specified 'unit'. e.g. if the data acquisition system stores values in this object as signed 16-bit integers (int16 range -32,768 to 32,767) that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition values to recorded volts is 2.5/32768/8000 = 9.5367e-9. • **Data Type:** float32 • **Required:** False • **Default Value:** 1.0 • **Name:** conversion
——offset	Attribute	Scalar to add to the data after scaling by 'conversion' to finalize its coercion to the specified 'unit'. Two common examples of this include (a) data stored in an unsigned type that requires a shift after scaling to re-center the data, and (b) specialized recording devices that naturally cause a scalar offset with respect to the true units. • **Data Type:** float32 • **Required:** False • **Default Value:** 0.0 • **Name:** offset
——resolution	Attribute	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 -1.0. • **Data Type:** float32 • **Required:** False • **Default Value:** -1.0 • **Name:** resolution
——unit	Attribute	Base unit of measurement for working with the data. Actual stored values are not necessarily stored in these units. To access the data in these units, multiply 'data' by 'conversion' and add 'offset'. • **Data Type:** text • **Name:** unit

continues on next page

Table 4.107 – continued from previous page

Id	Type	Description
——continuity	Attribute	Optionally describe the continuity of the data. Can be “continuous”, “instantaneous”, or “step”. For example, a voltage trace would be “continuous”, because samples are recorded from a continuous process. An array of lick times would be “instantaneous”, because the data represents distinct moments in time. Times of image presentations would be “step” because the picture remains the same until the next timepoint. This field is optional, but is useful in providing information about the underlying data. It may inform the way this data is interpreted, the way it is visualized, and what analysis methods are applicable. • **Data Type:** text • **Required:** False • **Name:** continuity
—dimension	Dataset	Number of pixels on x, y, (and z) axes. • **Quantity:** 0 or 1 • **Data Type:** int32 • **Dimensions:** [‘rank’] • **Shape:** [None] • **Name:** dimension
—external_file	Dataset	Paths to one or more external file(s). The field is only present if format=’external’. This is only relevant if the image series 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 NWB file and that file is linked to this file. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** [‘num_files’] • **Shape:** [None] • **Name:** external_file
——starting_frame	Attribute	Each external image may contain one or more consecutive frames of the full ImageSeries. This attribute serves as an index to indicate which frames each file contains, to faciliate random access. The ‘starting_frame’ attribute, hence, contains a list of frame numbers within the full ImageSeries of the first frame of each file listed in the parent ‘external_file’ dataset. Zero-based indexing is used (hence, the first element will always be zero). For example, if the ‘external_file’ dataset has three paths to files and the first file has 5 frames, the second file has 10 frames, and the third file has 20 frames, then this attribute will have values [0, 5, 15]. If there is a single external file that holds all of the frames of the ImageSeries (and so there is a single element in the ‘external_file’ dataset), then this attribute should have value [0]. • **Data Type:** int32 • **Dimensions:** [‘num_files’] • **Shape:** [None] • **Name:** starting_frame

continues on next page

Table 4.107 – continued from previous page

Id	Type	Description
—format	Dataset	Format of image. If this is 'external', then the attribute 'external_file' contains the path information to the image files. If this is 'raw', then the raw (single-channel) binary data is stored in the 'data' dataset. If this attribute is not present, then the default format='raw' case is assumed. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** format
—starting_time	Dataset	Timestamp of the first sample in seconds. When timestamps are uniformly spaced, the timestamp of the first sample can be specified and all subsequent ones calculated from the sampling rate attribute. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Name:** starting_time
——rate	Attribute	Sampling rate, in Hz. • **Data Type:** float32 • **Name:** rate
——unit	Attribute	Unit of measurement for time, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—timestamps	Dataset	Timestamps for samples stored in data, in seconds, relative to the common experiment master-clock stored in NWBFile.timestamps_reference_time. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** timestamps
——interval	Attribute	Value is '1' • **Data Type:** int32 • **Value:** 1 • **Name:** interval
——unit	Attribute	Unit of measurement for timestamps, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit

continues on next page

Table 4.107 – continued from previous page

Id	Type	Description
—control	Dataset	Numerical labels that apply to each time point in data for the purpose of querying and slicing data by these values. If present, the length of this array should be the same size as the first dimension of data. • **Quantity:** 0 or 1 • **Data Type:** uint8 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** control
—control_description	Dataset	Description of each control value. Must be present if control is present. If present, control_description[0] should describe time points where control == 0. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** ['num_control_values'] • **Shape:** [None] • **Name:** control_description
—imaging_plane	Link	Link to ImagingPlane object from which this TimeSeries data was generated. • **Target Type** *ImagingPlane* • **Name:** imaging_plane
—device	Link	Link to the Device object that was used to capture these images. • **Target Type** *Device* • **Name:** device

Table 4.108: Groups contained in <TwoPhotonSeries>

Id	Type	Description
<TwoPhotonSeries>	Group	Top level Group for <TwoPhotonSeries> • **Neurodata Type:** *TwoPhotonSeries* • **Extends:** *ImageSeries*
—imaging_plane	Link	Link to ImagingPlane object from which this TimeSeries data was generated. • **Target Type** *ImagingPlane* • **Name:** imaging_plane
—device	Link	Link to the Device object that was used to capture these images. • **Target Type** *Device* • **Name:** device

continues on next page

Table 4.108 – continued from previous page

Id	Type	Description
—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. • **Quantity:** 0 or 1 • **Name:** sync

4.11.2.1 Groups: sync

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.

- **Quantity:** 0 or 1
- **Name:** sync

4.11.3 RoiResponseSeries

Overview: ROI responses over an imaging plane. The first dimension represents time. The second dimension, if present, represents ROIs.

`RoiResponseSeries` extends `TimeSeries` and includes all elements of *TimeSeries* with the following additions or changes.

- **Extends:** *TimeSeries*
- **Primitive Type:** Group
- **Inherits from:** *TimeSeries*, *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.ophys.yaml
- **Source Specification:** see Section 5.12.3

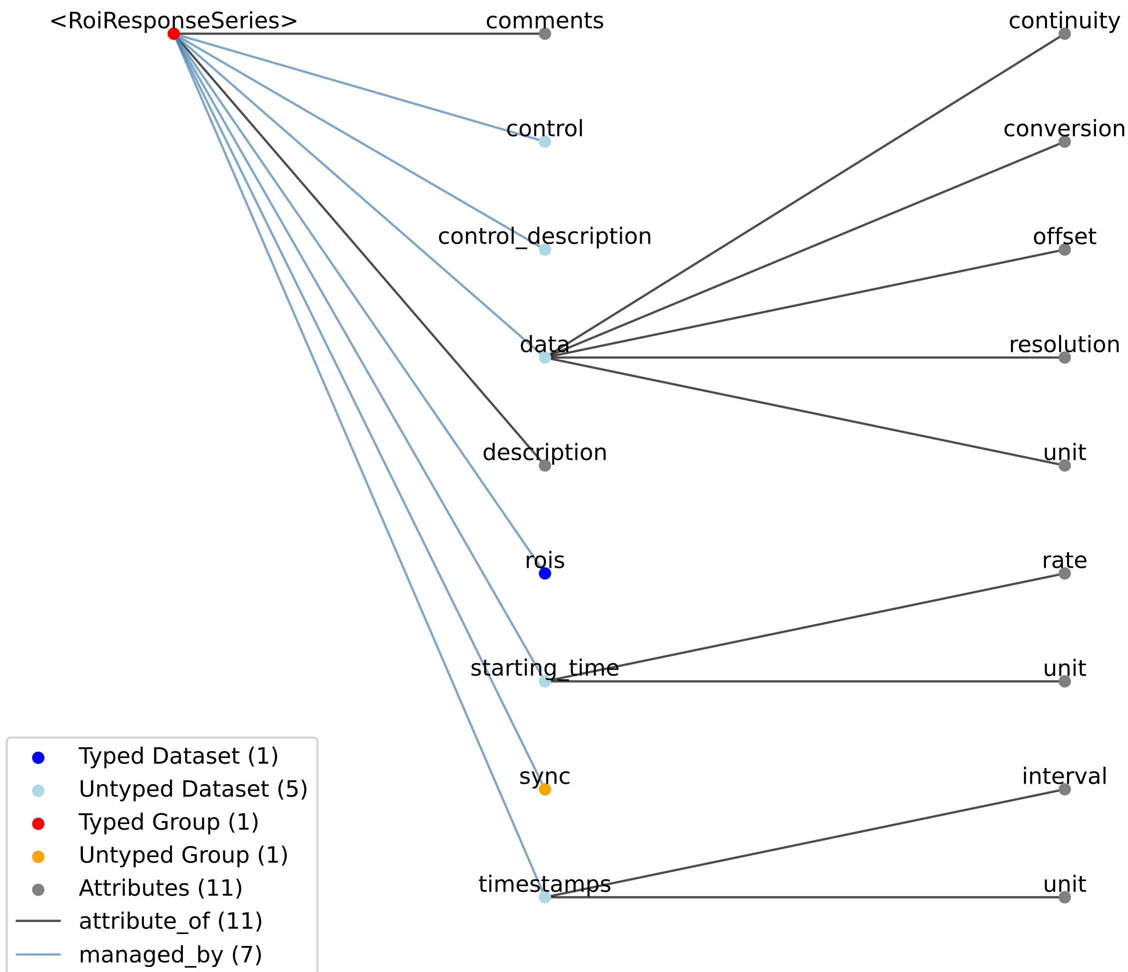


Table 4.109: Datasets, Links, and Attributes contained in <RoiResponseSeries>

Id	Type	Description
<RoiResponseSeries>	Group	Top level Group for <RoiResponseSeries> • **Neurodata Type:** *RoiResponseSeries* • **Extends:** *TimeSeries*
—description	Attribute	Description of the time series. • **Data Type:** text • **Required:** False • **Default Value:** no description • **Name:** description
—comments	Attribute	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. • **Data Type:** text • **Required:** False • **Default Value:** no comments • **Name:** comments
—data	Dataset	Signals from ROIs. • **Data Type:** numeric • **Dimensions:** [['num_times'], ['num_times', 'num_ROIs']] • **Shape:** [[None], [None, None]] • **Name:** data
——conversion	Attribute	Scalar to multiply each element in data to convert it to the specified 'unit'. If the data are stored in acquisition system units or other units that require a conversion to be interpretable, multiply the data by 'conversion' to convert the data to the specified 'unit'. e.g. if the data acquisition system stores values in this object as signed 16-bit integers (int16 range -32,768 to 32,767) that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition values to recorded volts is 2.5/32768/8000 = 9.5367e-9. • **Data Type:** float32 • **Required:** False • **Default Value:** 1.0 • **Name:** conversion
——offset	Attribute	Scalar to add to the data after scaling by 'conversion' to finalize its coercion to the specified 'unit'. Two common examples of this include (a) data stored in an unsigned type that requires a shift after scaling to re-center the data, and (b) specialized recording devices that naturally cause a scalar offset with respect to the true units. • **Data Type:** float32 • **Required:** False • **Default Value:** 0.0 • **Name:** offset

continues on next page

Table 4.109 – continued from previous page

Id	Type	Description
——resolution	Attribute	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 -1.0. • **Data Type:** float32 • **Required:** False • **Default Value:** -1.0 • **Name:** resolution
——unit	Attribute	Base unit of measurement for working with the data. Actual stored values are not necessarily stored in these units. To access the data in these units, multiply 'data' by 'conversion' and add 'offset'. • **Data Type:** text • **Name:** unit
——continuity	Attribute	Optionally describe the continuity of the data. Can be "continuous", "instantaneous", or "step". For example, a voltage trace would be "continuous", because samples are recorded from a continuous process. An array of lick times would be "instantaneous", because the data represents distinct moments in time. Times of image presentations would be "step" because the picture remains the same until the next timepoint. This field is optional, but is useful in providing information about the underlying data. It may inform the way this data is interpreted, the way it is visualized, and what analysis methods are applicable. • **Data Type:** text • **Required:** False • **Name:** continuity
—rois	Dataset	DynamicTableRegion referencing into an ROITable containing information on the ROIs stored in this timeseries. • **Extends:** DynamicTableRegion • **Name:** rois
—starting_time	Dataset	Timestamp of the first sample in seconds. When timestamps are uniformly spaced, the timestamp of the first sample can be specified and all subsequent ones calculated from the sampling rate attribute. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Name:** starting_time
——rate	Attribute	Sampling rate, in Hz. • **Data Type:** float32 • **Name:** rate
——unit	Attribute	Unit of measurement for time, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit

continues on next page

Table 4.109 – continued from previous page

Id	Type	Description
—timestamps	Dataset	Timestamps for samples stored in data, in seconds, relative to the common experiment master-clock stored in NWBFile.timestamps_reference_time. • **Quantity:** 0 or 1 • **Data Type:** float64 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** timestamps
——interval	Attribute	Value is '1' • **Data Type:** int32 • **Value:** 1 • **Name:** interval
——unit	Attribute	Unit of measurement for timestamps, which is fixed to 'seconds'. • **Data Type:** text • **Value:** seconds • **Name:** unit
—control	Dataset	Numerical labels that apply to each time point in data for the purpose of querying and slicing data by these values. If present, the length of this array should be the same size as the first dimension of data. • **Quantity:** 0 or 1 • **Data Type:** uint8 • **Dimensions:** ['num_times'] • **Shape:** [None] • **Name:** control
—control_description	Dataset	Description of each control value. Must be present if control is present. If present, control_description[0] should describe time points where control == 0. • **Quantity:** 0 or 1 • **Data Type:** text • **Dimensions:** ['num_control_values'] • **Shape:** [None] • **Name:** control_description

Table 4.110: Groups contained in <RoiResponseSeries>

Id	Type	Description
<RoiResponseSeries>	Group	Top level Group for <RoiResponseSeries> • **Neurodata Type:** *RoiResponseSeries* • **Extends:** *TimeSeries*

continues on next page

Table 4.110 – continued from previous page

Id	Type	Description
—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. • **Quantity:** 0 or 1 • **Name:** sync

4.11.3.1 Groups: sync

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.

- **Quantity:** 0 or 1
- **Name:** sync

4.11.4 DfOverF

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

`DfOverF` extends `NWBDataInterface` and includes all elements of *NWBDataInterface* with the following additions or changes.

- **Extends:** *NWBDataInterface*
- **Primitive Type:** Group
- **Default Name:** DfOverF
- **Inherits from:** *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.ophys.yaml
- **Source Specification:** see Section 5.12.4

Table 4.111: Groups contained in <DfOverF>

Id	Type	Description
<DfOverF>	Group	Top level Group for <DfOverF> • **Neurodata Type:** *DfOverF* • **Extends:** *NWBDataInterface* • **Default Name:** DfOverF
—<*RoiResponseSeries*>	Group	RoiResponseSeries object(s) containing dF/F for a ROI. • **Extends:** *RoiResponseSeries* • **Quantity:** 1 or more

4.11.4.1 Groups: <RoiResponseSeries>

RoiResponseSeries object(s) containing dF/F for a ROI.

- **Extends:** *RoiResponseSeries*
- **Quantity:** 1 or more

4.11.5 Fluorescence

Overview: 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` extends `NWBDataInterface` and includes all elements of *NWBDataInterface* with the following additions or changes.

- **Extends:** *NWBDataInterface*
- **Primitive Type:** Group
- **Default Name:** Fluorescence
- **Inherits from:** *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.ophys.yaml
- **Source Specification:** see Section 5.12.5

Table 4.112: Groups contained in <Fluorescence>

Id	Type	Description
<Fluorescence>	Group	Top level Group for <Fluorescence> • **Neurodata Type:** *Fluorescence* • **Extends:** *NWBDataInterface* • **Default Name:** Fluorescence
—<*RoiResponseSeries*>	Group	RoiResponseSeries object(s) containing fluorescence data for a ROI. • **Extends:** *RoiResponseSeries* • **Quantity:** 1 or more

4.11.5.1 Groups: <RoiResponseSeries>

RoiResponseSeries object(s) containing fluorescence data for a ROI.

- **Extends:** *RoiResponseSeries*
- **Quantity:** 1 or more

4.11.6 ImageSegmentation

Overview: 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` extends `NWBDataInterface` and includes all elements of *NWBDataInterface* with the following additions or changes.

- **Extends:** *NWBDataInterface*
- **Primitive Type:** Group
- **Default Name:** ImageSegmentation
- **Inherits from:** *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.ophys.yaml
- **Source Specification:** see Section 5.12.6

Table 4.113: Groups contained in <ImageSegmentation>

Id	Type	Description
<ImageSegmentation>	Group	Top level Group for <ImageSegmentation> • **Neurodata Type:** *ImageSegmentation* • **Extends:** *NWBDataInterface* • **Default Name:** ImageSegmentation
—<*PlaneSegmentation*>	Group	Results from image segmentation of a specific imaging plane. • **Extends:** *PlaneSegmentation* • **Quantity:** 1 or more

4.11.6.1 Groups: <PlaneSegmentation>

Results from image segmentation of a specific imaging plane.

- **Extends:** *PlaneSegmentation*
- **Quantity:** 1 or more

4.11.7 PlaneSegmentation

Overview: Results from image segmentation of a specific imaging plane.

`PlaneSegmentation` extends `DynamicTable` and includes all elements of DynamicTable with the following additions or changes.

- **Extends:** DynamicTable
- **Primitive Type:** Group
- **Inherits from:** DynamicTable, Container
- **Source filename:** nwb.ophys.yaml
- **Source Specification:** see Section 5.12.7

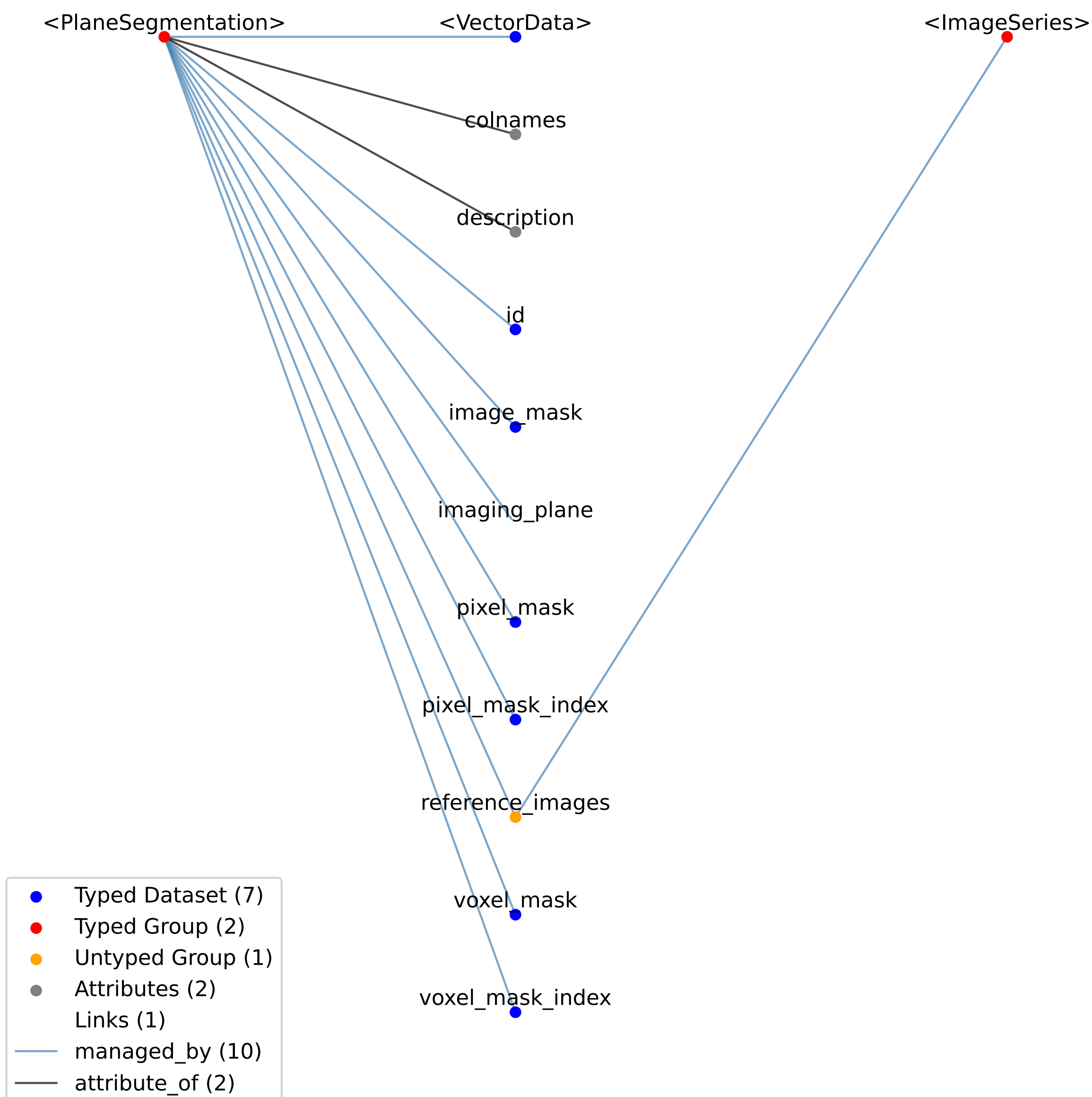


Table 4.114: Datasets, Links, and Attributes contained in <PlaneSegmentation>

Id	Type	Description
<PlaneSegmentation>	Group	Top level Group for <PlaneSegmentation> • **Neurodata Type:** *PlaneSegmentation* • **Extends:** DynamicTable
—colnames	Attribute	The names of the columns in this table. This should be used to specify an order to the columns. • **Data Type:** text • **Dimensions:** [‘num_columns’] • **Shape:** [None] • **Name:** colnames
—description	Attribute	Description of what is in this dynamic table. • **Data Type:** text • **Name:** description
—image_mask	Dataset	ROI masks for each ROI. Each image mask is the size of the original imaging plane (or volume) and members of the ROI are finite non-zero. • **Extends:** VectorData • **Quantity:** 0 or 1 • **Dimensions:** [[‘num_roi’, ‘num_x’, ‘num_y’], [‘num_roi’, ‘num_x’, ‘num_y’, ‘num_z’]] • **Shape:** [[None, None, None], [None, None, None, None]] • **Name:** image_mask
—pixel_mask_index	Dataset	Index into pixel_mask. • **Extends:** VectorIndex • **Quantity:** 0 or 1 • **Name:** pixel_mask_index
—pixel_mask	Dataset	Pixel masks for each ROI: a list of indices and weights for the ROI. Pixel masks are concatenated and parsing of this dataset is maintained by the PlaneSegmentation • **Extends:** VectorData • **Quantity:** 0 or 1 • **Data Type: Compound data type with the following elements:** – **x:** Pixel x-coordinate. (*dtype*= uint32) – **y:** Pixel y-coordinate. (*dtype*= uint32) – **weight:** Weight of the pixel. (*dtype*= float32) • **Name:** pixel_mask
—voxel_mask_index	Dataset	Index into voxel_mask. • **Extends:** VectorIndex • **Quantity:** 0 or 1 • **Name:** voxel_mask_index

continues on next page

Table 4.114 – continued from previous page

Id	Type	Description
—voxel_mask	Dataset	Voxel masks for each ROI: a list of indices and weights for the ROI. Voxel masks are concatenated and parsing of this dataset is maintained by the PlaneSegmentation • **Extends:** VectorData • **Quantity:** 0 or 1 • **Data Type: Compound data type with the following elements:** – **x:** Voxel x-coordinate. (*dtype*= uint32) – **y:** Voxel y-coordinate. (*dtype*= uint32) – **z:** Voxel z-coordinate. (*dtype*= uint32) – **weight:** Weight of the voxel. (*dtype*= float32) • **Name:** voxel_mask
—id	Dataset	Array of unique identifiers for the rows of this dynamic table. • **Extends:** ElementIdentifiers • **Data Type:** int • **Dimensions:** [‘num_rows’] • **Shape:** [None] • **Name:** id
—<VectorData>	Dataset	Vector columns, including index columns, of this dynamic table. • **Extends:** VectorData • **Quantity:** 0 or more
—imaging_plane	Link	Link to ImagingPlane object from which this data was generated. • **Target Type** *ImagingPlane* • **Name:** imaging_plane

Table 4.115: Groups contained in <PlaneSegmentation>

Id	Type	Description
<PlaneSegmentation>	Group	Top level Group for <PlaneSegmentation> • **Neurodata Type:** *PlaneSegmentation* • **Extends:** DynamicTable
—imaging_plane	Link	Link to ImagingPlane object from which this data was generated. • **Target Type** *ImagingPlane* • **Name:** imaging_plane
—reference_images	Group	Image stacks that the segmentation masks apply to. • **Name:** reference_images

4.11.7.1 Groups: reference_images

Image stacks that the segmentation masks apply to.

- **Name:** reference_images

Table 4.116: Groups contained in <reference_images>

Id	Type	Description
reference_images	Group	Top level Group for reference_images • **Name:** reference_images
—<*ImageSeries*>	Group	One or more image stacks that the masks apply to (can be one-element stack). • **Extends:** *ImageSeries* • **Quantity:** 0 or more

4.11.7.2 Groups: reference_images/<ImageSeries>

One or more image stacks that the masks apply to (can be one-element stack).

- **Extends:** *ImageSeries*
- **Quantity:** 0 or more

4.11.8 ImagingPlane

Overview: An imaging plane and its metadata.

`ImagingPlane` extends `NWBContainer` and includes all elements of *NWBContainer* with the following additions or changes.

- **Extends:** *NWBContainer*
- **Primitive Type:** Group
- **Inherits from:** *NWBContainer*, Container
- **Source filename:** nwb.ophys.yaml
- **Source Specification:** see Section 5.12.8

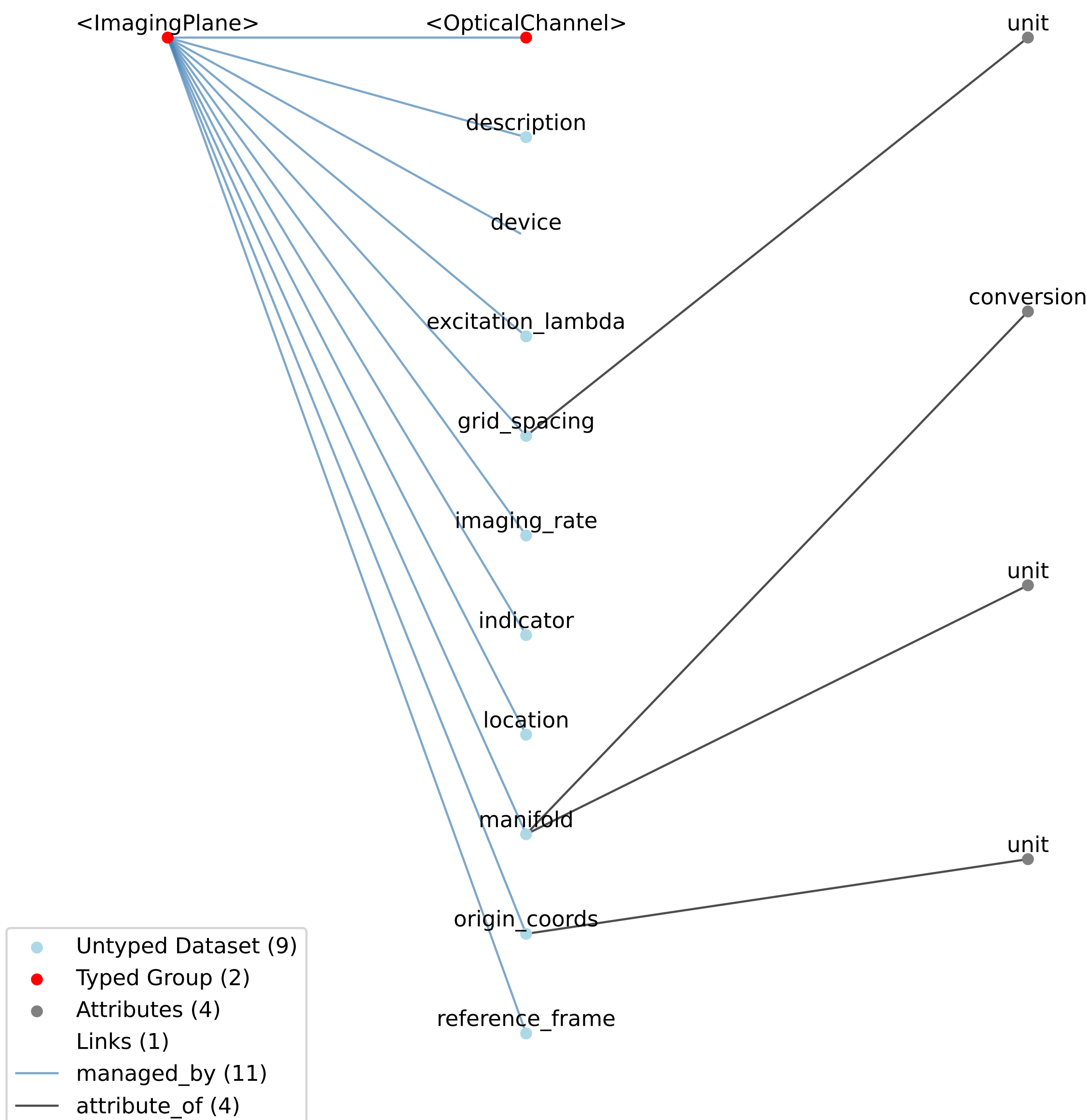


Table 4.117: Datasets, Links, and Attributes contained in <ImagingPlane>

Id	Type	Description
<ImagingPlane>	Group	Top level Group for <ImagingPlane> • **Neurodata Type:** *ImagingPlane* • **Extends:** *NWBContainer*
—description	Dataset	Description of the imaging plane. • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** description
—excitation_lambda	Dataset	Excitation wavelength, in nm. • **Data Type:** float32 • **Name:** excitation_lambda
—imaging_rate	Dataset	Rate that images are acquired, in Hz. If the corresponding TimeSeries is present, the rate should be stored there instead. • **Quantity:** 0 or 1 • **Data Type:** float32 • **Name:** imaging_rate
—indicator	Dataset	Calcium indicator. • **Data Type:** text • **Name:** indicator
—location	Dataset	Location of the imaging plane. Specify the area, layer, comments on estimation of area/layer, stereotaxic coordinates if in vivo, etc. Use standard atlas names for anatomical regions when possible. • **Data Type:** text • **Name:** location
—manifold	Dataset	DEPRECATED Physical position of each pixel. 'xyz' represents the position of the pixel relative to the defined coordinate space. Deprecated in favor of origin_coords and grid_spacing. • **Quantity:** 0 or 1 • **Data Type:** float32 • **Dimensions:** [['height', 'width', 'x, y, z'], ['height', 'width', 'depth', 'x, y, z']] • **Shape:** [[None, None, 3], [None, None, None, 3]] • **Name:** manifold

continues on next page

Table 4.117 – continued from previous page

Id	Type	Description
——conversion	Attribute	Scalar to multiply each element in data to convert it to the specified ‘unit’. If the data are stored in acquisition system units or other units that require a conversion to be interpretable, multiply the data by ‘conversion’ to convert the data to the specified ‘unit’. e.g. if the data acquisition system stores values in this object as pixels from x = -500 to 499, y = -500 to 499 that correspond to a 2 m x 2 m range, then the ‘conversion’ multiplier to get from raw data acquisition pixel units to meters is 2/1000. • **Data Type:** float32 • **Required:** False • **Default Value:** 1.0 • **Name:** conversion
——unit	Attribute	Base unit of measurement for working with the data. The default value is ‘meters’. • **Data Type:** text • **Required:** False • **Default Value:** meters • **Name:** unit
—origin_coords	Dataset	Physical location of the first element of the imaging plane (0, 0) for 2-D data or (0, 0, 0) for 3-D data. See also reference_frame for what the physical location is relative to (e.g., bregma). • **Quantity:** 0 or 1 • **Data Type:** float32 • **Dimensions:** [[‘x, y’], [‘x, y, z’]] • **Shape:** [[2], [3]] • **Name:** origin_coords
——unit	Attribute	Measurement units for origin_coords. The default value is ‘meters’. • **Data Type:** text • **Required:** False • **Default Value:** meters • **Name:** unit
—grid_spacing	Dataset	Space between pixels in (x, y) or voxels in (x, y, z) directions, in the specified unit. Assumes imaging plane is a regular grid. See also reference_frame to interpret the grid. • **Quantity:** 0 or 1 • **Data Type:** float32 • **Dimensions:** [[‘x, y’], [‘x, y, z’]] • **Shape:** [[2], [3]] • **Name:** grid_spacing
——unit	Attribute	Measurement units for grid_spacing. The default value is ‘meters’. • **Data Type:** text • **Required:** False • **Default Value:** meters • **Name:** unit

continues on next page

Table 4.117 – continued from previous page

Id	Type	Description
—reference_frame	Dataset	Describes reference frame of origin_coords and grid_spacing. For example, this can be a text description of the anatomical location and orientation of the grid defined by origin_coords and grid_spacing or the vectors needed to transform or rotate the grid to a common anatomical axis (e.g., AP/DV/ML). This field is necessary to interpret origin_coords and grid_spacing. If origin_coords and grid_spacing are not present, then this field is not required. For example, if the microscope takes 10 x 10 x 2 images, where the first value of the data matrix (index (0, 0, 0)) corresponds to (-1.2, -0.6, -2) mm relative to bregma, the spacing between pixels is 0.2 mm in x, 0.2 mm in y and 0.5 mm in z, and larger numbers in x means more anterior, larger numbers in y means more rightward, and larger numbers in z means more ventral, then enter the following – origin_coords = (-1.2, -0.6, -2) grid_spacing = (0.2, 0.2, 0.5) reference_frame = "Origin coordinates are relative to bregma. First dimension corresponds to anterior-posterior axis (larger index = more anterior). Second dimension corresponds to medial-lateral axis (larger index = more rightward). Third dimension corresponds to dorsal-ventral axis (larger index = more ventral)." • **Quantity:** 0 or 1 • **Data Type:** text • **Name:** reference_frame
—device	Link	Link to the Device object that was used to record from this electrode. • **Target Type** *Device* • **Name:** device

Table 4.118: Groups contained in <ImagingPlane>

Id	Type	Description
<ImagingPlane>	Group	Top level Group for <ImagingPlane> • **Neurodata Type:** *ImagingPlane* • **Extends:** *NWBContainer*
—device	Link	Link to the Device object that was used to record from this electrode. • **Target Type** *Device* • **Name:** device
—<*OpticalChannel*>	Group	An optical channel used to record from an imaging plane. • **Extends:** *OpticalChannel* • **Quantity:** 1 or more

4.11.8.1 Groups: <OpticalChannel>

An optical channel used to record from an imaging plane.

- **Extends:** *OpticalChannel*
- **Quantity:** 1 or more

4.11.9 OpticalChannel

Overview: An optical channel used to record from an imaging plane.

`OpticalChannel` extends `NWBContainer` and includes all elements of *NWBContainer* with the following additions or changes.

- **Extends:** *NWBContainer*
- **Primitive Type:** Group
- **Inherits from:** *NWBContainer*, Container
- **Source filename:** nwb.ophys.yaml
- **Source Specification:** see Section 5.12.9

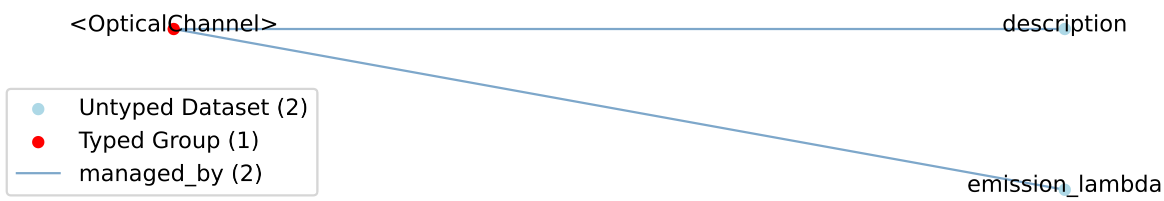


Table 4.119: Datasets, Links, and Attributes contained in <OpticalChannel>

Id	Type	Description
<OpticalChannel>	Group	Top level Group for <OpticalChannel> • **Neurodata Type:** *OpticalChannel* • **Extends:** *NWBContainer*
—description	Dataset	Description or other notes about the channel. • **Data Type:** text • **Name:** description
—emission_lambda	Dataset	Emission wavelength for channel, in nm. • **Data Type:** float32 • **Name:** emission_lambda

4.11.10 MotionCorrection

Overview: 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` extends `NWBDataInterface` and includes all elements of *NWBDataInterface* with the following additions or changes.

- **Extends:** *NWBDataInterface*
- **Primitive Type:** Group
- **Default Name:** MotionCorrection
- **Inherits from:** *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.ophys.yaml
- **Source Specification:** see Section 5.12.10

Table 4.120: Groups contained in <MotionCorrection>

Id	Type	Description
<MotionCorrection>	Group	Top level Group for <MotionCorrection> • **Neurodata Type:** *MotionCorrection* • **Extends:** *NWBDataInterface* • **Default Name:** MotionCorrection
—<*CorrectedImageStack*>	Group	Reuslts from motion correction of an image stack. • **Extends:** *CorrectedImageStack* • **Quantity:** 1 or more

4.11.10.1 Groups: <CorrectedImageStack>

Reuslts from motion correction of an image stack.

- **Extends:** *CorrectedImageStack*
- **Quantity:** 1 or more

4.11.11 CorrectedImageStack

Overview: Reuslts from motion correction of an image stack.

`CorrectedImageStack` extends `NWBDataInterface` and includes all elements of *NWBDataInterface* with the following additions or changes.

- **Extends:** *NWBDataInterface*
- **Primitive Type:** Group
- **Inherits from:** *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.ophys.yaml
- **Source Specification:** see Section 5.12.11

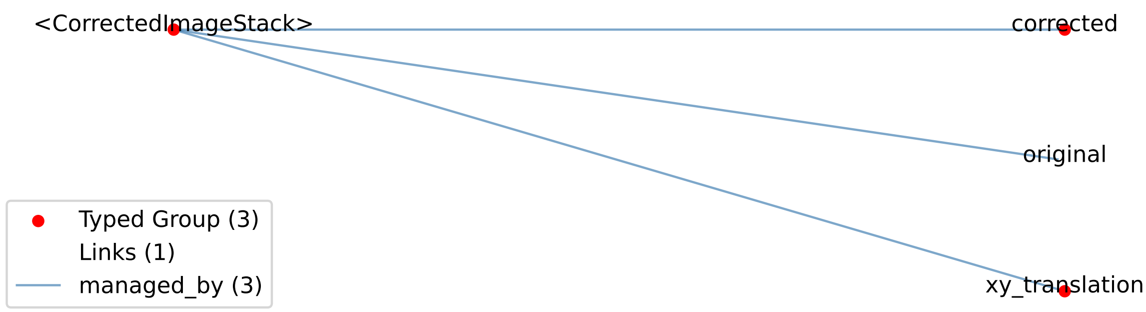


Table 4.121: Datasets, Links, and Attributes contained in <CorrectedImageStack>

Id	Type	Description
<CorrectedImageStack>	Group	Top level Group for <CorrectedImageStack> • **Neurodata Type:** *CorrectedImageStack* • **Extends:** *NWBDataInterface*
—original	Link	Link to ImageSeries object that is being registered. • **Target Type** *ImageSeries* • **Name:** original

Table 4.122: Groups contained in <CorrectedImageStack>

Id	Type	Description
<CorrectedImageStack>	Group	Top level Group for <CorrectedImageStack> • **Neurodata Type:** *CorrectedImageStack* • **Extends:** *NWBDataInterface*
—original	Link	Link to ImageSeries object that is being registered. • **Target Type** *ImageSeries* • **Name:** original

continues on next page

Table 4.122 – continued from previous page

Id	Type	Description
—corrected	Group	Image stack with frames shifted to the common coordinates. • **Extends:** *ImageSeries* • **Name:** corrected
—xy_translation	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. • **Extends:** *TimeSeries* • **Name:** xy_translation

4.11.11.1 Groups: corrected

Image stack with frames shifted to the common coordinates.

- **Extends:** *ImageSeries*
- **Name:** corrected

4.11.11.2 Groups: xy_translation

Stores the x,y delta necessary to align each frame to the common coordinates, for example, to align each frame to a reference image.

- **Extends:** *TimeSeries*
- **Name:** xy_translation

4.12 Retinotopy

This source module contains neurodata_type for retinotopy data.

4.12.1 ImagingRetinotopy

Overview: 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. This group does not store the raw responses imaged during retinotopic mapping or the stimuli presented, but rather the resulting phase and power maps after applying a Fourier transform on the averaged responses. 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` extends `NWBDataInterface` and includes all elements of *NWBDataInterface* with the following additions or changes.

- **Extends:** *NWBDataInterface*
- **Primitive Type:** Group
- **Default Name:** ImagingRetinotopy
- **Inherits from:** *NWBDataInterface*, *NWBContainer*, Container
- **Source filename:** nwb.retinotopy.yaml
- **Source Specification:** see Section 5.13.1

Table 4.123: Datasets, Links, and Attributes contained in <ImagingRetinotopy>

Id	Type	Description
<ImagingRetinotopy>	Group	Top level Group for <ImagingRetinotopy> • **Neurodata Type:** *ImagingRetinotopy* • **Extends:** *NWBDataInterface* • **Default Name:** ImagingRetinotopy
—axis_1_phase_map	Dataset	Phase response to stimulus on the first measured axis. • **Data Type:** float32 • **Dimensions:** ['num_rows', 'num_cols'] • **Shape:** [None, None] • **Name:** axis_1_phase_map
——dimension	Attribute	Number of rows and columns in the image. **Additional Information:** row, column representation is equivalent to height, width. • **Data Type:** int32 • **Dimensions:** ['num_rows, num_cols'] • **Shape:** [2] • **Name:** dimension

continues on next page

Table 4.123 – continued from previous page

Id	Type	Description
——field_of_view	Attribute	Size of viewing area, in meters. • **Data Type:** float32 • **Dimensions:** [‘height, width’] • **Shape:** [2] • **Name:** field_of_view
——unit	Attribute	Unit that axis data is stored in (e.g., degrees). • **Data Type:** text • **Name:** unit
—axis_1_power_map	Dataset	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. • **Quantity:** 0 or 1 • **Data Type:** float32 • **Dimensions:** [‘num_rows’, ‘num_cols’] • **Shape:** [None, None] • **Name:** axis_1_power_map
——dimension	Attribute	Number of rows and columns in the image. **Additional Information:** row, column representation is equivalent to height, width. • **Data Type:** int32 • **Dimensions:** [‘num_rows, num_cols’] • **Shape:** [2] • **Name:** dimension
——field_of_view	Attribute	Size of viewing area, in meters. • **Data Type:** float32 • **Dimensions:** [‘height, width’] • **Shape:** [2] • **Name:** field_of_view
——unit	Attribute	Unit that axis data is stored in (e.g., degrees). • **Data Type:** text • **Name:** unit
—axis_2_phase_map	Dataset	Phase response to stimulus on the second measured axis. • **Data Type:** float32 • **Dimensions:** [‘num_rows’, ‘num_cols’] • **Shape:** [None, None] • **Name:** axis_2_phase_map
——dimension	Attribute	Number of rows and columns in the image. **Additional Information:** row, column representation is equivalent to height, width. • **Data Type:** int32 • **Dimensions:** [‘num_rows, num_cols’] • **Shape:** [2] • **Name:** dimension

continues on next page

Table 4.123 – continued from previous page

Id	Type	Description
——field_of_view	Attribute	Size of viewing area, in meters. • **Data Type:** float32 • **Dimensions:** ['height, width'] • **Shape:** [2] • **Name:** field_of_view
——unit	Attribute	Unit that axis data is stored in (e.g., degrees). • **Data Type:** text • **Name:** unit
—axis_2_power_map	Dataset	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. • **Quantity:** 0 or 1 • **Data Type:** float32 • **Dimensions:** ['num_rows', 'num_cols'] • **Shape:** [None, None] • **Name:** axis_2_power_map
——dimension	Attribute	Number of rows and columns in the image. **Additional Information:** row, column representation is equivalent to height, width. • **Data Type:** int32 • **Dimensions:** ['num_rows, num_cols'] • **Shape:** [2] • **Name:** dimension
——field_of_view	Attribute	Size of viewing area, in meters. • **Data Type:** float32 • **Dimensions:** ['height, width'] • **Shape:** [2] • **Name:** field_of_view
——unit	Attribute	Unit that axis data is stored in (e.g., degrees). • **Data Type:** text • **Name:** unit
—axis_descriptions	Dataset	Two-element array describing the contents of the two response axis fields. Description should be something like ['altitude', 'azimuth'] or '['radius', 'theta']. • **Data Type:** text • **Dimensions:** ['axis_1, axis_2'] • **Shape:** [2] • **Name:** axis_descriptions

continues on next page

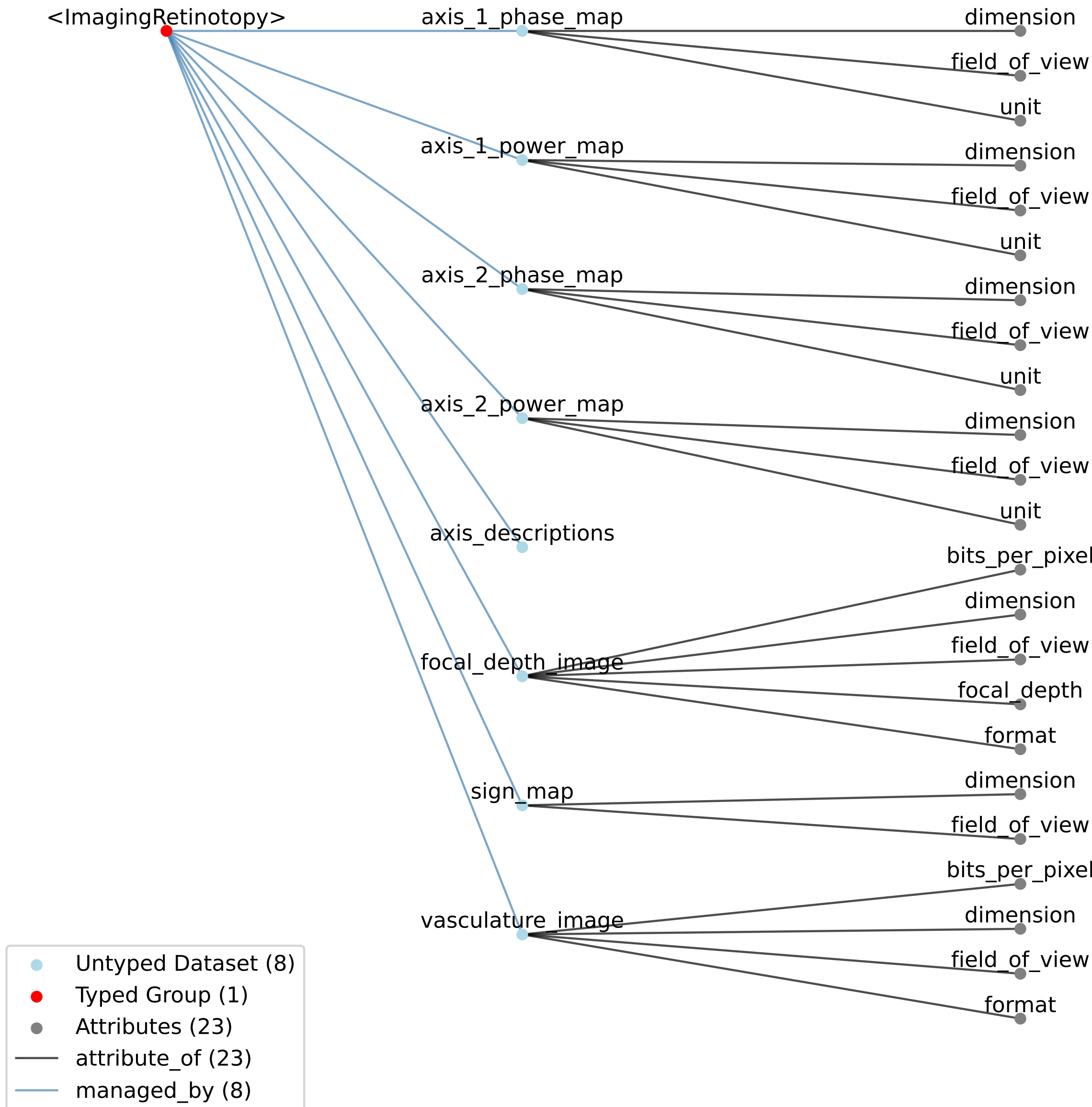
Table 4.123 – continued from previous page

Id	Type	Description
—focal_depth_image	Dataset	Gray-scale image taken with same settings/parameters (e.g., focal depth, wavelength) as data collection. Array format: [rows][columns]. • **Quantity:** 0 or 1 • **Data Type:** uint16 • **Dimensions:** ['num_rows', 'num_cols'] • **Shape:** [None, None] • **Name:** focal_depth_image
——bits_per_pixel	Attribute	Number of bits used to represent each value. This is necessary to determine maximum (white) pixel value. • **Data Type:** int32 • **Name:** bits_per_pixel
——dimension	Attribute	Number of rows and columns in the image. **Additional Information:** row, column representation is equivalent to height, width. • **Data Type:** int32 • **Dimensions:** ['num_rows, num_cols'] • **Shape:** [2] • **Name:** dimension
——field_of_view	Attribute	Size of viewing area, in meters. • **Data Type:** float32 • **Dimensions:** ['height, width'] • **Shape:** [2] • **Name:** field_of_view
——focal_depth	Attribute	Focal depth offset, in meters. • **Data Type:** float32 • **Name:** focal_depth
——format	Attribute	Format of image. Right now only 'raw' is supported. • **Data Type:** text • **Name:** format
—sign_map	Dataset	Sine of the angle between the direction of the gradient in axis_1 and axis_2. • **Quantity:** 0 or 1 • **Data Type:** float32 • **Dimensions:** ['num_rows', 'num_cols'] • **Shape:** [None, None] • **Name:** sign_map

continues on next page

Table 4.123 – continued from previous page

Id	Type	Description
——dimension	Attribute	Number of rows and columns in the image. **Additional Information:** row, column representation is equivalent to height, width. • **Data Type:** int32 • **Dimensions:** [‘num_rows, num_cols’] • **Shape:** [2] • **Name:** dimension
——field_of_view	Attribute	Size of viewing area, in meters. • **Data Type:** float32 • **Dimensions:** [‘height, width’] • **Shape:** [2] • **Name:** field_of_view
—vasculature_image	Dataset	Gray-scale anatomical image of cortical surface. Array structure: [rows][columns] • **Data Type:** uint16 • **Dimensions:** [‘num_rows’, ‘num_cols’] • **Shape:** [None, None] • **Name:** vasculature_image
——bits_per_pixel	Attribute	Number of bits used to represent each value. This is necessary to determine maximum (white) pixel value • **Data Type:** int32 • **Name:** bits_per_pixel
——dimension	Attribute	Number of rows and columns in the image. **Additional Information:** row, column representation is equivalent to height, width. • **Data Type:** int32 • **Dimensions:** [‘num_rows, num_cols’] • **Shape:** [2] • **Name:** dimension
——field_of_view	Attribute	Size of viewing area, in meters. • **Data Type:** float32 • **Dimensions:** [‘height, width’] • **Shape:** [2] • **Name:** field_of_view
——format	Attribute	Format of image. Right now only ‘raw’ is supported. • **Data Type:** text • **Name:** format


<ImagingRetinotopy>
axis_1_phase_map
axis_1_power_map
axis_2_phase_map
axis_2_power_map
axis_descriptions
focal_depth_image
sign_map
vasculature_image
dimension
field_of_view
unit
dimension
field_of_view
unit
dimension
field_of_view
unit
dimension
field_of_view
unit
bits_per_pixel
dimension
field_of_view
focal_depth
format
dimension
field_of_view
bits_per_pixel
dimension
field_of_view
format
Untyped Dataset (8)
Typed Group (1)
Attributes (23)
attribute_of (23)
managed_by (8)

CHAPTER

FIVE

SCHEMA SOURCES

Source Specification: see Section 5.1

5.1 Namespace – NWB core

Description: see Section 3.1

YAML Specification:


```yaml
author:
- Andrew Tritt
- Oliver Ruebel
- Ryan Ly
- Ben Dichter
- Keith Godfrey
- Jeff Teeters
contact:
- ajtritt@lbl.gov
- oruebel@lbl.gov
- rly@lbl.gov
- bdichter@lbl.gov
- keithg@alleninstitute.org
- jteeters@berkeley.edu
doc: NWB namespace
full_name: NWB core
name: core
schema:
- namespace: hdmf-common
- doc: This source module contains base data types used throughout the NWB data format.
  source: nwb.base.yaml
  title: Base data types
- doc: This source module contains neurodata_types for device data.
  source: nwb.device.yaml
  title: Devices
- doc: This source module contains neurodata_types for epoch data.
  source: nwb.epoch.yaml
  title: Epochs
- doc: This source module contains neurodata_types for image data.
  source: nwb.image.yaml
  title: Image data
```


(continues on next page)

(continued from previous page)

```yaml
- doc: Main NWB file specification.
  source: nwb.file.yaml
  title: NWB file
- doc: Miscellaneous types.
  source: nwb.misc.yaml
  title: Miscellaneous neurodata_types.
- doc: This source module contains neurodata_types for behavior data.
  source: nwb.behavior.yaml
  title: Behavior
- doc: This source module contains neurodata_types for extracellular electrophysiology
    data.
  source: nwb.ecephys.yaml
  title: Extracellular electrophysiology
- doc: This source module contains neurodata_types for intracellular electrophysiology
    data.
  source: nwb.icephys.yaml
  title: Intracellular electrophysiology
- doc: This source module contains neurodata_types for opto-genetics data.
  source: nwb.ogen.yaml
  title: Optogenetics
- doc: This source module contains neurodata_types for optical physiology data.
  source: nwb.ophys.yaml
  title: Optical physiology
- doc: This source module contains neurodata_type for retinotopy data.
  source: nwb.retinotopy.yaml
  title: Retinotopy
version: 2.6.0-alpha
```

5.2 Base data types

This source module contains base data types used throughout the NWB data format.

5.2.1 NWBData

Extends: Data

Description: see Section 4.1.1

YAML Specification:

```yaml
doc: An abstract data type for a dataset.
neurodata_type_def: NWBData
neurodata_type_inc: Data
```

5.2.2 TimeSeriesReferenceVectorData

Extends: VectorData

Description: see Section 4.1.2

YAML Specification:

```yaml
attributes:
- doc: Description of what these vectors represent.
  dtype: text
  name: description
default_name: timeseries
doc: Column storing references to a TimeSeries (rows). For each TimeSeries this VectorData
  column stores the start_index and count to indicate the range in time to be selected
  as well as an object reference to the TimeSeries.
dtype:
- doc: Start index into the TimeSeries 'data' and 'timestamp' datasets of the referenced
    TimeSeries. The first dimension of those arrays is always time.
  dtype: int32
  name: idx_start
- doc: Number of data samples available in this time series, during this epoch
  dtype: int32
  name: count
- doc: The TimeSeries that this index applies to
  dtype:
    reftype: object
    target_type: TimeSeries
  name: timeseries
neurodata_type_def: TimeSeriesReferenceVectorData
neurodata_type_inc: VectorData
```

5.2.3 Image

Extends: *NWBData*

Description: see Section 4.1.3

YAML Specification:


```yaml
attributes:
- doc: Pixel resolution of the image, in pixels per centimeter.
  dtype: float32
  name: resolution
  required: false
- doc: Description of the image.
  dtype: text
  name: description
  required: false
dims:
- - x
  - y
- - x
  - y
  - r, g, b
- - x
  - y
  - r, g, b, a
doc: An abstract data type for an image. Shape can be 2-D (x, y), or 3-D where the
  third dimension can have three or four elements, e.g. (x, y, (r, g, b)) or (x, y,
  (r, g, b, a)).
dtype: numeric
neurodata_type_def: Image
neurodata_type_inc: NWBData
shape:
- -
  -
- -
  -
  - 3
- -
  -
  - 4
```

5.2.4 ImageReferences

Extends: *NWBData*

Description: see Section 4.1.4

YAML Specification:


```yaml
dims:
- num_images
doc: Ordered dataset of references to Image objects.
dtype:
  reftype: object
  target_type: Image
neurodata_type_def: ImageReferences
neurodata_type_inc: NWBData
shape:
-
```

5.2.5 NWBContainer

Extends: Container

Description: see Section 4.1.5

YAML Specification:

```yaml
doc: An abstract data type for a generic container storing collections of data and
  metadata. Base type for all data and metadata containers.
neurodata_type_def: NWBContainer
neurodata_type_inc: Container
```

5.2.6 NWBDataInterface

Extends: *NWBContainer*

Description: see Section 4.1.6

YAML Specification:


```yaml
doc: An abstract data type for a generic container storing collections of data, as
  opposed to metadata.
neurodata_type_def: NWBDataInterface
neurodata_type_inc: NWBContainer
```

5.2.7 TimeSeries

Extends: *NWBDataInterface*

Description: see Section 4.1.7

YAML Specification:

```yaml
attributes:
- default_value: no description
  doc: Description of the time series.
  dtype: text
  name: description
  required: false
- default_value: no comments
  doc: 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.
  dtype: text
  name: comments
  required: false
datasets:
- attributes:
  - default_value: 1.0
    doc: Scalar to multiply each element in data to convert it to the specified 'unit'.
      If the data are stored in acquisition system units or other units that require
      a conversion to be interpretable, multiply the data by 'conversion' to convert
      the data to the specified 'unit'. e.g. if the data acquisition system stores
      values in this object as signed 16-bit integers (int16 range -32,768 to 32,767)
      that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system
      gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition
      values to recorded volts is 2.5/32768/8000 = 9.5367e-9.
    dtype: float32
    name: conversion
    required: false
  - default_value: 0.0
    doc: Scalar to add to the data after scaling by 'conversion' to finalize its coercion
      to the specified 'unit'. Two common examples of this include (a) data stored
      in an unsigned type that requires a shift after scaling to re-center the data,
      and (b) specialized recording devices that naturally cause a scalar offset with
      respect to the true units.
    dtype: float32
    name: offset
    required: false
  - default_value: -1.0
    doc: 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 -1.0.
    dtype: float32
    name: resolution
    required: false
  - doc: Base unit of measurement for working with the data. Actual stored values
      are not necessarily stored in these units. To access the data in these units,
      multiply 'data' by 'conversion' and add 'offset'.
```

(continues on next page)

(continued from previous page)

```yaml
    dtype: text
    name: unit
  - doc: Optionally describe the continuity of the data. Can be "continuous", "instantaneous",
      or "step". For example, a voltage trace would be "continuous", because samples
      are recorded from a continuous process. An array of lick times would be "instantaneous",
      because the data represents distinct moments in time. Times of image presentations
      would be "step" because the picture remains the same until the next timepoint.
      This field is optional, but is useful in providing information about the underlying
      data. It may inform the way this data is interpreted, the way it is visualized,
      and what analysis methods are applicable.
    dtype: text
    name: continuity
    required: false
  dims:
  - - num_times
  - - num_times
    - num_DIM2
  - - num_times
    - num_DIM2
    - num_DIM3
  - - num_times
    - num_DIM2
    - num_DIM3
    - num_DIM4
  doc: Data values. Data can be in 1-D, 2-D, 3-D, or 4-D. The first dimension should
    always represent time. This can also be used to store binary data (e.g., image
    frames). This can also be a link to data stored in an external file.
  name: data
  shape:
  - - 
  - - 
    - 
  - - 
    - 
    - 
  - - 
    - 
    - 
    - 
- attributes:
  - doc: Sampling rate, in Hz.
    dtype: float32
    name: rate
  - doc: Unit of measurement for time, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  doc: Timestamp of the first sample in seconds. When timestamps are uniformly spaced,
    the timestamp of the first sample can be specified and all subsequent ones calculated
    from the sampling rate attribute.
```

(continues on next page)

(continued from previous page)

```yaml
  dtype: float64
  name: starting_time
  quantity: '?'
- attributes:
  - doc: Value is '1'
    dtype: int32
    name: interval
    value: 1
  - doc: Unit of measurement for timestamps, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  dims:
  - num_times
  doc: Timestamps for samples stored in data, in seconds, relative to the common experiment
    master-clock stored in NWBFile.timestamps_reference_time.
  dtype: float64
  name: timestamps
  quantity: '?'
  shape:
  -
- dims:
  - num_times
  doc: Numerical labels that apply to each time point in data for the purpose of querying
    and slicing data by these values. If present, the length of this array should
    be the same size as the first dimension of data.
  dtype: uint8
  name: control
  quantity: '?'
  shape:
  -
- dims:
  - num_control_values
  doc: Description of each control value. Must be present if control is present. If
    present, control_description[0] should describe time points where control == 0.
  dtype: text
  name: control_description
  quantity: '?'
  shape:
  -
doc: General purpose time series.
groups:
- doc: 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.
  name: sync
  quantity: '?'
```

(continues on next page)

(continued from previous page)

```
neurodata_type_def: TimeSeries
neurodata_type_inc: NWBDataInterface
```

5.2.8 ProcessingModule

Extends: *NWBContainer*

Description: see Section 4.1.8

YAML Specification:

```yaml
attributes:
- doc: Description of this collection of processed data.
  dtype: text
  name: description
doc: A collection of processed data.
groups:
- doc: Data objects stored in this collection.
  neurodata_type_inc: NWBDataInterface
  quantity: '*'
- doc: Tables stored in this collection.
  neurodata_type_inc: DynamicTable
  quantity: '*'
neurodata_type_def: ProcessingModule
neurodata_type_inc: NWBContainer
```

5.2.9 Images

Extends: *NWBDataInterface*

Description: see Section 4.1.9

YAML Specification:

```yaml
attributes:
- doc: Description of this collection of images.
  dtype: text
  name: description
datasets:
- doc: Images stored in this collection.
  neurodata_type_inc: Image
  quantity: +
- doc: Ordered dataset of references to Image objects stored in the parent group.
    Each Image object in the Images group should be stored once and only once, so
    the dataset should have the same length as the number of images.
  name: order_of_images
  neurodata_type_inc: ImageReferences
  quantity: '?'
default_name: Images
doc: A collection of images with an optional way to specify the order of the images
  using the "order_of_images" dataset. An order must be specified if the images are
  referenced by index, e.g., from an IndexSeries.
neurodata_type_def: Images
neurodata_type_inc: NWBDataInterface
```

5.3 Devices

This source module contains neurodata_types for device data.

5.3.1 Device

Extends: *NWBContainer*

Description: see Section 4.2.1

YAML Specification:

```yaml
attributes:
- doc: Description of the device (e.g., model, firmware version, processing software
    version, etc.) as free-form text.
  dtype: text
  name: description
  required: false
- doc: The name of the manufacturer of the device.
  dtype: text
  name: manufacturer
  required: false
doc: Metadata about a data acquisition device, e.g., recording system, electrode,
  microscope.
neurodata_type_def: Device
neurodata_type_inc: NWBContainer
```

5.4 Epochs

This source module contains neurodata_types for epoch data.

5.4.1 TimeIntervals

Extends: DynamicTable

Description: see Section 4.3.1

YAML Specification:

```yaml
attributes:
- dims:
  - num_columns
  doc: The names of the columns in this table. This should be used to specify an order
    to the columns.
  dtype: text
  name: colnames
  shape:
  -
- doc: Description of what is in this dynamic table.
  dtype: text
  name: description
datasets:
- doc: Start time of epoch, in seconds.
  dtype: float32
  name: start_time
  neurodata_type_inc: VectorData
- doc: Stop time of epoch, in seconds.
  dtype: float32
  name: stop_time
  neurodata_type_inc: VectorData
- doc: User-defined tags that identify or categorize events.
  dtype: text
  name: tags
  neurodata_type_inc: VectorData
  quantity: '?'
- doc: Index for tags.
  name: tags_index
  neurodata_type_inc: VectorIndex
  quantity: '?'
- doc: An index into a TimeSeries object.
  name: timeseries
  neurodata_type_inc: TimeSeriesReferenceVectorData
  quantity: '?'
- doc: Index for timeseries.
  name: timeseries_index
  neurodata_type_inc: VectorIndex
  quantity: '?'
- dims:
  - num_rows
  doc: Array of unique identifiers for the rows of this dynamic table.
```

(continues on next page)

(continued from previous page)

```yaml
  dtype: int
  name: id
  neurodata_type_inc: ElementIdentifiers
  shape:
  -
- doc: Vector columns, including index columns, of this dynamic table.
  neurodata_type_inc: VectorData
  quantity: '*'
doc: A container for aggregating epoch data and the TimeSeries that each epoch applies
  to.
neurodata_type_def: TimeIntervals
neurodata_type_inc: DynamicTable
```

5.5 Image data

This source module contains neurodata_types for image data.

5.5.1 GrayscaleImage

Extends: *Image*

Description: see Section 4.4.1

YAML Specification:


```
attributes:
- doc: Pixel resolution of the image, in pixels per centimeter.
  dtype: float32
  name: resolution
  required: false
- doc: Description of the image.
  dtype: text
  name: description
  required: false
dims:
- x
- y
doc: A grayscale image.
dtype: numeric
neurodata_type_def: GrayscaleImage
neurodata_type_inc: Image
shape:
-
-
```

5.5.2 RGBImage

Extends: *Image*

Description: see Section 4.4.2

YAML Specification:


```yaml
attributes:
- doc: Pixel resolution of the image, in pixels per centimeter.
  dtype: float32
  name: resolution
  required: false
- doc: Description of the image.
  dtype: text
  name: description
  required: false
dims:
- x
- y
- r, g, b
doc: A color image.
dtype: numeric
neurodata_type_def: RGBImage
neurodata_type_inc: Image
shape:
-
-
- 3
```

5.5.3 RGBAImage

Extends: *Image*

Description: see Section 4.4.3

YAML Specification:


```yaml
attributes:
- doc: Pixel resolution of the image, in pixels per centimeter.
  dtype: float32
  name: resolution
  required: false
- doc: Description of the image.
  dtype: text
  name: description
  required: false
dims:
- x
- y
- r, g, b, a
doc: A color image with transparency.
dtype: numeric
neurodata_type_def: RGBAImage
neurodata_type_inc: Image
shape:
-
-
- 4
```

5.5.4 ImageSeries

Extends: *TimeSeries*

Description: see Section 4.4.4

YAML Specification:


```yaml
attributes:
- default_value: no description
  doc: Description of the time series.
  dtype: text
  name: description
  required: false
- default_value: no comments
  doc: 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.
  dtype: text
  name: comments
  required: false
datasets:
- attributes:
  - default_value: 1.0
    doc: Scalar to multiply each element in data to convert it to the specified 'unit'.
      If the data are stored in acquisition system units or other units that require
      a conversion to be interpretable, multiply the data by 'conversion' to convert
      the data to the specified 'unit'. e.g. if the data acquisition system stores
      values in this object as signed 16-bit integers (int16 range -32,768 to 32,767)
      that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system
      gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition
      values to recorded volts is 2.5/32768/8000 = 9.5367e-9.
    dtype: float32
    name: conversion
    required: false
  - default_value: 0.0
    doc: Scalar to add to the data after scaling by 'conversion' to finalize its coercion
      to the specified 'unit'. Two common examples of this include (a) data stored
      in an unsigned type that requires a shift after scaling to re-center the data,
      and (b) specialized recording devices that naturally cause a scalar offset with
      respect to the true units.
    dtype: float32
    name: offset
    required: false
  - default_value: -1.0
    doc: 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 -1.0.
    dtype: float32
    name: resolution
    required: false
  - doc: Base unit of measurement for working with the data. Actual stored values
      are not necessarily stored in these units. To access the data in these units,
      multiply 'data' by 'conversion' and add 'offset'.
```


(continues on next page)

(continued from previous page)

```
  dtype: text
  name: unit
 - doc: Optionally describe the continuity of the data. Can be "continuous", "instantaneous",
   or "step". For example, a voltage trace would be "continuous", because samples
   are recorded from a continuous process. An array of lick times would be "instantaneous",
   because the data represents distinct moments in time. Times of image presentations
   would be "step" because the picture remains the same until the next timepoint.
   This field is optional, but is useful in providing information about the underlying
   data. It may inform the way this data is interpreted, the way it is visualized,
   and what analysis methods are applicable.
  dtype: text
  name: continuity
  required: false
 dims:
 - - frame
   - x
   - y
 - - frame
   - x
   - y
   - z
 doc: Binary data representing images across frames. If data are stored in an external
   file, this should be an empty 3D array.
 dtype: numeric
 name: data
 shape:
 - -
   -
   -
 - -
   -
   -
   -
- dims:
  - rank
  doc: Number of pixels on x, y, (and z) axes.
  dtype: int32
  name: dimension
  quantity: '?'
  shape:
  -
- attributes:
  - dims:
    - num_files
    doc: Each external image may contain one or more consecutive frames of the full
      ImageSeries. This attribute serves as an index to indicate which frames each
      file contains, to faciliate random access. The 'starting_frame' attribute, hence,
      contains a list of frame numbers within the full ImageSeries of the first frame
      of each file listed in the parent 'external_file' dataset. Zero-based indexing
      is used (hence, the first element will always be zero). For example, if the
```

(continues on next page)

(continued from previous page)

```
      'external_file' dataset has three paths to files and the first file has 5 frames,
      the second file has 10 frames, and the third file has 20 frames, then this attribute
      will have values [0, 5, 15]. If there is a single external file that holds all
      of the frames of the ImageSeries (and so there is a single element in the 'external_file'
      dataset), then this attribute should have value [0].
    dtype: int32
    name: starting_frame
    shape:
    -
  dims:
  - num_files
  doc: Paths to one or more external file(s). The field is only present if format='external'.
    This is only relevant if the image series 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 NWB file and that file is linked to this file.
  dtype: text
  name: external_file
  quantity: '?'
  shape:
  -
- default_value: raw
  doc: Format of image. If this is 'external', then the attribute 'external_file'
    contains the path information to the image files. If this is 'raw', then the raw
    (single-channel) binary data is stored in the 'data' dataset. If this attribute
    is not present, then the default format='raw' case is assumed.
  dtype: text
  name: format
  quantity: '?'
- attributes:
  - doc: Sampling rate, in Hz.
    dtype: float32
    name: rate
  - doc: Unit of measurement for time, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  doc: Timestamp of the first sample in seconds. When timestamps are uniformly spaced,
    the timestamp of the first sample can be specified and all subsequent ones calculated
    from the sampling rate attribute.
  dtype: float64
  name: starting_time
  quantity: '?'
- attributes:
  - doc: Value is '1'
    dtype: int32
    name: interval
    value: 1
  - doc: Unit of measurement for timestamps, which is fixed to 'seconds'.
    dtype: text
```

(continues on next page)

(continued from previous page)

```yaml
      name: unit
      value: seconds
    dims:
    - num_times
    doc: Timestamps for samples stored in data, in seconds, relative to the common experiment
      master-clock stored in NWBFile.timestamps_reference_time.
    dtype: float64
    name: timestamps
    quantity: '?'
    shape:
    -
  - dims:
    - num_times
    doc: Numerical labels that apply to each time point in data for the purpose of querying
      and slicing data by these values. If present, the length of this array should
      be the same size as the first dimension of data.
    dtype: uint8
    name: control
    quantity: '?'
    shape:
    -
  - dims:
    - num_control_values
    doc: Description of each control value. Must be present if control is present. If
      present, control_description[0] should describe time points where control == 0.
    dtype: text
    name: control_description
    quantity: '?'
    shape:
    -
  doc: General image data that is common between acquisition and stimulus time series.
    Sometimes the image data is stored in the file in a raw format while other times
    it will be stored as a series of external image files in the host file system. The
    data field will either be binary data, if the data is stored in the NWB file, or
    empty, if the data is stored in an external image stack. [frame][x][y] or [frame][x][y][z].
  groups:
  - doc: 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.
    name: sync
    quantity: '?'
  links:
  - doc: Link to the Device object that was used to capture these images.
    name: device
    quantity: '?'
    target_type: Device
```

(continues on next page)

(continued from previous page)

```
neurodata_type_def: ImageSeries
neurodata_type_inc: TimeSeries
```

5.5.5 ImageMaskSeries

Extends: *ImageSeries*

Description: see Section 4.4.5

YAML Specification:


```yaml
attributes:
- default_value: no description
  doc: Description of the time series.
  dtype: text
  name: description
  required: false
- default_value: no comments
  doc: 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.
  dtype: text
  name: comments
  required: false
datasets:
- attributes:
  - default_value: 1.0
    doc: Scalar to multiply each element in data to convert it to the specified 'unit'.
      If the data are stored in acquisition system units or other units that require
      a conversion to be interpretable, multiply the data by 'conversion' to convert
      the data to the specified 'unit'. e.g. if the data acquisition system stores
      values in this object as signed 16-bit integers (int16 range -32,768 to 32,767)
      that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system
      gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition
      values to recorded volts is 2.5/32768/8000 = 9.5367e-9.
    dtype: float32
    name: conversion
    required: false
  - default_value: 0.0
    doc: Scalar to add to the data after scaling by 'conversion' to finalize its coercion
      to the specified 'unit'. Two common examples of this include (a) data stored
      in an unsigned type that requires a shift after scaling to re-center the data,
      and (b) specialized recording devices that naturally cause a scalar offset with
      respect to the true units.
    dtype: float32
    name: offset
    required: false
  - default_value: -1.0
    doc: 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 -1.0.
    dtype: float32
    name: resolution
    required: false
  - doc: Base unit of measurement for working with the data. Actual stored values
      are not necessarily stored in these units. To access the data in these units,
      multiply 'data' by 'conversion' and add 'offset'.
```


(continues on next page)

(continued from previous page)

```
    dtype: text
    name: unit
  - doc: Optionally describe the continuity of the data. Can be "continuous", "instantaneous",
      or "step". For example, a voltage trace would be "continuous", because samples
      are recorded from a continuous process. An array of lick times would be "instantaneous",
      because the data represents distinct moments in time. Times of image presentations
      would be "step" because the picture remains the same until the next timepoint.
      This field is optional, but is useful in providing information about the underlying
      data. It may inform the way this data is interpreted, the way it is visualized,
      and what analysis methods are applicable.
    dtype: text
    name: continuity
    required: false
  dims:
  - - frame
    - x
    - y
  - - frame
    - x
    - y
    - z
  doc: Binary data representing images across frames. If data are stored in an external
    file, this should be an empty 3D array.
  dtype: numeric
  name: data
  shape:
  - - -
    -
    -
  - - -
    -
    -
    -
- dims:
  - rank
  doc: Number of pixels on x, y, (and z) axes.
  dtype: int32
  name: dimension
  quantity: '?'
  shape:
  -
- attributes:
  - dims:
    - num_files
    doc: Each external image may contain one or more consecutive frames of the full
      ImageSeries. This attribute serves as an index to indicate which frames each
      file contains, to faciliate random access. The 'starting_frame' attribute, hence,
      contains a list of frame numbers within the full ImageSeries of the first frame
      of each file listed in the parent 'external_file' dataset. Zero-based indexing
      is used (hence, the first element will always be zero). For example, if the
```

(continues on next page)

(continued from previous page)

```yaml
      'external_file' dataset has three paths to files and the first file has 5 frames,
      the second file has 10 frames, and the third file has 20 frames, then this attribute
      will have values [0, 5, 15]. If there is a single external file that holds all
      of the frames of the ImageSeries (and so there is a single element in the 'external_file'
      dataset), then this attribute should have value [0].
    dtype: int32
    name: starting_frame
    shape:
    -
  dims:
  - num_files
  doc: Paths to one or more external file(s). The field is only present if format='external'.
    This is only relevant if the image series 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 NWB file and that file is linked to this file.
  dtype: text
  name: external_file
  quantity: '?'
  shape:
  -
- default_value: raw
  doc: Format of image. If this is 'external', then the attribute 'external_file'
    contains the path information to the image files. If this is 'raw', then the raw
    (single-channel) binary data is stored in the 'data' dataset. If this attribute
    is not present, then the default format='raw' case is assumed.
  dtype: text
  name: format
  quantity: '?'
- attributes:
  - doc: Sampling rate, in Hz.
    dtype: float32
    name: rate
  - doc: Unit of measurement for time, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  doc: Timestamp of the first sample in seconds. When timestamps are uniformly spaced,
    the timestamp of the first sample can be specified and all subsequent ones calculated
    from the sampling rate attribute.
  dtype: float64
  name: starting_time
  quantity: '?'
- attributes:
  - doc: Value is '1'
    dtype: int32
    name: interval
    value: 1
  - doc: Unit of measurement for timestamps, which is fixed to 'seconds'.
    dtype: text
```

(continues on next page)

(continued from previous page)

```yaml
    name: unit
    value: seconds
  dims:
  - num_times
  doc: Timestamps for samples stored in data, in seconds, relative to the common experiment
    master-clock stored in NWBFile.timestamps_reference_time.
  dtype: float64
  name: timestamps
  quantity: '?'
  shape:
  -
- dims:
  - num_times
  doc: Numerical labels that apply to each time point in data for the purpose of querying
    and slicing data by these values. If present, the length of this array should
    be the same size as the first dimension of data.
  dtype: uint8
  name: control
  quantity: '?'
  shape:
  -
- dims:
  - num_control_values
  doc: Description of each control value. Must be present if control is present. If
    present, control_description[0] should describe time points where control == 0.
  dtype: text
  name: control_description
  quantity: '?'
  shape:
  -
doc: 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.
groups:
- doc: 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.
  name: sync
  quantity: '?'
links:
- doc: Link to ImageSeries object that this image mask is applied to.
  name: masked_imageseries
  target_type: ImageSeries
- doc: Link to the Device object that was used to capture these images.
  name: device
  quantity: '?'
```

(continues on next page)

(continued from previous page)

```
  target_type: Device
neurodata_type_def: ImageMaskSeries
neurodata_type_inc: ImageSeries
```

5.5.6 OpticalSeries

Extends: *ImageSeries*

Description: see Section 4.4.6

YAML Specification:


```yaml
attributes:
- default_value: no description
  doc: Description of the time series.
  dtype: text
  name: description
  required: false
- default_value: no comments
  doc: 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.
  dtype: text
  name: comments
  required: false
datasets:
- doc: Distance from camera/monitor to target/eye.
  dtype: float32
  name: distance
  quantity: '?'
- dims:
  - - width, height
  - - width, height, depth
  doc: Width, height and depth of image, or imaged area, in meters.
  dtype: float32
  name: field_of_view
  quantity: '?'
  shape:
  - - 2
  - - 3
- attributes:
  - default_value: 1.0
    doc: Scalar to multiply each element in data to convert it to the specified 'unit'.
      If the data are stored in acquisition system units or other units that require
      a conversion to be interpretable, multiply the data by 'conversion' to convert
      the data to the specified 'unit'. e.g. if the data acquisition system stores
      values in this object as signed 16-bit integers (int16 range -32,768 to 32,767)
      that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system
      gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition
      values to recorded volts is 2.5/32768/8000 = 9.5367e-9.
    dtype: float32
    name: conversion
    required: false
  - default_value: 0.0
    doc: Scalar to add to the data after scaling by 'conversion' to finalize its coercion
      to the specified 'unit'. Two common examples of this include (a) data stored
      in an unsigned type that requires a shift after scaling to re-center the data,
      and (b) specialized recording devices that naturally cause a scalar offset with
```


(continues on next page)

(continued from previous page)

```
        respect to the true units.
      dtype: float32
      name: offset
      required: false
    - default_value: -1.0
      doc: 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 -1.0.
      dtype: float32
      name: resolution
      required: false
    - doc: Base unit of measurement for working with the data. Actual stored values
        are not necessarily stored in these units. To access the data in these units,
        multiply 'data' by 'conversion' and add 'offset'.
      dtype: text
      name: unit
    - doc: Optionally describe the continuity of the data. Can be "continuous", "instantaneous",
        or "step". For example, a voltage trace would be "continuous", because samples
        are recorded from a continuous process. An array of lick times would be "instantaneous",
        because the data represents distinct moments in time. Times of image presentations
        would be "step" because the picture remains the same until the next timepoint.
        This field is optional, but is useful in providing information about the underlying
        data. It may inform the way this data is interpreted, the way it is visualized,
        and what analysis methods are applicable.
      dtype: text
      name: continuity
      required: false
    dims:
    - - frame
      - x
      - y
    - - frame
      - x
      - y
      - r, g, b
    doc: Images presented to subject, either grayscale or RGB
    dtype: numeric
    name: data
    shape:
    - - 
      - 
      - 
    - - 
      - 
      - 
      - 3
  - doc: Description of image relative to some reference frame (e.g., which way is up).
      Must also specify frame of reference.
    dtype: text
    name: orientation
```

(continues on next page)

(continued from previous page)

```
  quantity: '?'
- dims:
  - rank
  doc: Number of pixels on x, y, (and z) axes.
  dtype: int32
  name: dimension
  quantity: '?'
  shape:
  -
- attributes:
  - dims:
    - num_files
    doc: Each external image may contain one or more consecutive frames of the full
      ImageSeries. This attribute serves as an index to indicate which frames each
      file contains, to faciliate random access. The 'starting_frame' attribute, hence,
      contains a list of frame numbers within the full ImageSeries of the first frame
      of each file listed in the parent 'external_file' dataset. Zero-based indexing
      is used (hence, the first element will always be zero). For example, if the
      'external_file' dataset has three paths to files and the first file has 5 frames,
      the second file has 10 frames, and the third file has 20 frames, then this attribute
      will have values [0, 5, 15]. If there is a single external file that holds all
      of the frames of the ImageSeries (and so there is a single element in the 'external_file'
      dataset), then this attribute should have value [0].
    dtype: int32
    name: starting_frame
    shape:
    -
  dims:
  - num_files
  doc: Paths to one or more external file(s). The field is only present if format='external'.
    This is only relevant if the image series 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 NWB file and that file is linked to this file.
  dtype: text
  name: external_file
  quantity: '?'
  shape:
  -
- default_value: raw
  doc: Format of image. If this is 'external', then the attribute 'external_file'
    contains the path information to the image files. If this is 'raw', then the raw
    (single-channel) binary data is stored in the 'data' dataset. If this attribute
    is not present, then the default format='raw' case is assumed.
  dtype: text
  name: format
  quantity: '?'
- attributes:
  - doc: Sampling rate, in Hz.
    dtype: float32
```

(continues on next page)

(continued from previous page)

```
    name: rate
  - doc: Unit of measurement for time, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  doc: Timestamp of the first sample in seconds. When timestamps are uniformly spaced,
    the timestamp of the first sample can be specified and all subsequent ones calculated
    from the sampling rate attribute.
  dtype: float64
  name: starting_time
  quantity: '?'
- attributes:
  - doc: Value is '1'
    dtype: int32
    name: interval
    value: 1
  - doc: Unit of measurement for timestamps, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  dims:
  - num_times
  doc: Timestamps for samples stored in data, in seconds, relative to the common experiment
    master-clock stored in NWBFile.timestamps_reference_time.
  dtype: float64
  name: timestamps
  quantity: '?'
  shape:
  -
- dims:
  - num_times
  doc: Numerical labels that apply to each time point in data for the purpose of querying
    and slicing data by these values. If present, the length of this array should
    be the same size as the first dimension of data.
  dtype: uint8
  name: control
  quantity: '?'
  shape:
  -
- dims:
  - num_control_values
  doc: Description of each control value. Must be present if control is present. If
    present, control_description[0] should describe time points where control == 0.
  dtype: text
  name: control_description
  quantity: '?'
  shape:
  -
doc: 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 (e.g., how much of the visual field the image covers, or how
```

(continues on next page)

(continued from previous page)

```yaml
  what is the area of the target being imaged). If the OpticalSeries represents acquired
  imaging data, orientation is also important.
groups:
- doc: 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.
  name: sync
  quantity: '?'
links:
- doc: Link to the Device object that was used to capture these images.
  name: device
  quantity: '?'
  target_type: Device
neurodata_type_def: OpticalSeries
neurodata_type_inc: ImageSeries
```

5.5.7 IndexSeries

Extends: *TimeSeries*

Description: see Section 4.4.7

YAML Specification:

```yaml
attributes:
- default_value: no description
  doc: Description of the time series.
  dtype: text
  name: description
  required: false
- default_value: no comments
  doc: 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.
  dtype: text
  name: comments
  required: false
datasets:
- attributes:
  - doc: This field is unused by IndexSeries.
    dtype: float32
    name: conversion
    required: false
  - doc: This field is unused by IndexSeries.
    dtype: float32
    name: resolution
    required: false
  - doc: This field is unused by IndexSeries.
    dtype: float32
    name: offset
    required: false
  - doc: This field is unused by IndexSeries and has the value N/A.
    dtype: text
    name: unit
    value: N/A
  - doc: Optionally describe the continuity of the data. Can be "continuous",
↪"instantaneous",
      or "step". For example, a voltage trace would be "continuous", because samples
      are recorded from a continuous process. An array of lick times would be
↪"instantaneous",
      because the data represents distinct moments in time. Times of image presentations
      would be "step" because the picture remains the same until the next timepoint.
      This field is optional, but is useful in providing information about the underlying
      data. It may inform the way this data is interpreted, the way it is visualized,
      and what analysis methods are applicable.
    dtype: text
    name: continuity
    required: false
  dims:
  - num_times
```

(continues on next page)

(continued from previous page)

```
  doc: Index of the image (using zero-indexing) in the linked Images object.
  dtype: uint32
  name: data
  shape:
  -
- attributes:
  - doc: Sampling rate, in Hz.
    dtype: float32
    name: rate
  - doc: Unit of measurement for time, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  doc: Timestamp of the first sample in seconds. When timestamps are uniformly spaced,
    the timestamp of the first sample can be specified and all subsequent ones calculated
    from the sampling rate attribute.
  dtype: float64
  name: starting_time
  quantity: '?'
- attributes:
  - doc: Value is '1'
    dtype: int32
    name: interval
    value: 1
  - doc: Unit of measurement for timestamps, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  dims:
  - num_times
  doc: Timestamps for samples stored in data, in seconds, relative to the common␣
↪experiment
    master-clock stored in NWBFile.timestamps_reference_time.
  dtype: float64
  name: timestamps
  quantity: '?'
  shape:
  -
- dims:
  - num_times
  doc: Numerical labels that apply to each time point in data for the purpose of querying
    and slicing data by these values. If present, the length of this array should
    be the same size as the first dimension of data.
  dtype: uint8
  name: control
  quantity: '?'
  shape:
  -
- dims:
  - num_control_values
  doc: Description of each control value. Must be present if control is present. If
    present, control_description[0] should describe time points where control == 0.
```

(continues on next page)

(continued from previous page)

```yaml
  dtype: text
  name: control_description
  quantity: '?'
  shape:
  -
doc: Stores indices to image frames stored in an ImageSeries. The purpose of the IndexSeries
  is to allow a static image stack to be stored in an Images object, 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 Images object, and the timestamps
  array indicates when that image was displayed.
groups:
- doc: 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.
  name: sync
  quantity: '?'
links:
- doc: Link to ImageSeries object containing images that are indexed. Use of this
    link is discouraged and will be deprecated. Link to an Images type instead.
  name: indexed_timeseries
  quantity: '?'
  target_type: ImageSeries
- doc: Link to Images object containing an ordered set of images that are indexed.
    The Images object must contain a 'ordered_images' dataset specifying the order
    of the images in the Images type.
  name: indexed_images
  quantity: '?'
  target_type: Images
neurodata_type_def: IndexSeries
neurodata_type_inc: TimeSeries
```

5.6 NWB file

Main NWB file specification.

5.6.1 ScratchData

Extends: *NWBData*

Description: see Section 4.5.1

YAML Specification:

```yaml
attributes:
- doc: Any notes the user has about the dataset being stored
  dtype: text
  name: notes
doc: Any one-off datasets
neurodata_type_def: ScratchData
neurodata_type_inc: NWBData
```

5.6.2 NWBFile

Extends: *NWBContainer*

Description: see Section 4.5.2

YAML Specification:

```yaml
attributes:
- doc: File version string. Use semantic versioning, e.g. 1.2.1. This will be the
    name of the format with trailing major, minor and patch numbers.
  dtype: text
  name: nwb_version
  value: 2.6.0
datasets:
- dims:
  - num_modifications
  doc: 'A record of the date the file was created and of subsequent modifications.
    The date is stored in UTC with local timezone offset as ISO 8601 extended formatted
    strings: 2018-09-28T14:43:54.123+02:00. Dates stored in UTC end in "Z" with no
    timezone offset. Date accuracy is up to milliseconds. The file can be created
    after the experiment was run, so this may differ from the experiment start time.
    Each modification to the nwb file adds a new entry to the array.'
  dtype: isodatetime
  name: file_create_date
  shape:
  -
- doc: A unique text identifier for the file. For example, 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.
  dtype: text
  name: identifier
- doc: A description of the experimental session and data in the file.
  dtype: text
  name: session_description
- doc: 'Date and time of the experiment/session start. The date is stored in UTC with
    local timezone offset as ISO 8601 extended formatted string: 2018-09-28T14:43:54.
↪123+02:00.
    Dates stored in UTC end in "Z" with no timezone offset. Date accuracy is up to
    milliseconds.'
  dtype: isodatetime
  name: session_start_time
- doc: 'Date and time corresponding to time zero of all timestamps. The date is stored
    in UTC with local timezone offset as ISO 8601 extended formatted string: 2018-09-
↪28T14:43:54.123+02:00.
    Dates stored in UTC end in "Z" with no timezone offset. Date accuracy is up to
    milliseconds. All times stored in the file use this time as reference (i.e., time
    zero).'
  dtype: isodatetime
  name: timestamps_reference_time
doc: An NWB file storing cellular-based neurophysiology data from a single experimental
  session.
groups:
- doc: Data streams recorded from the system, including ephys, ophys, tracking, etc.
```

(continues on next page)

(continued from previous page)

```yaml
    This group should be 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 NWB files. This will allow keeping bulky raw data out of the
    file while preserving the option of keeping some/all in the file. Acquired data
    includes tracking and experimental data streams (i.e., everything measured from
    the system). If bulky data is stored in the /acquisition group, the data can exist
    in a separate NWB file that is linked to by the file being used for processing
    and analysis.
  groups:
  - doc: Acquired, raw data.
    neurodata_type_inc: NWBDataInterface
    quantity: '*'
  - doc: Tabular data that is relevent to acquisition
    neurodata_type_inc: DynamicTable
    quantity: '*'
  name: acquisition
- doc: 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 (e.g., neurodata_types) and appropriately documented. 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 could be shared with other labs.
  groups:
  - doc: Custom analysis results.
    neurodata_type_inc: NWBContainer
    quantity: '*'
  - doc: Tabular data that is relevent to data stored in analysis
    neurodata_type_inc: DynamicTable
    quantity: '*'
  name: analysis
- datasets:
  - doc: Any one-off datasets
    neurodata_type_inc: ScratchData
    quantity: '*'
  doc: A place to store one-off analysis results. Data placed here is not intended
    for sharing. By placing data here, users acknowledge that there is no guarantee
    that their data meets any standard.
  groups:
  - doc: Any one-off containers
    neurodata_type_inc: NWBContainer
    quantity: '*'
  - doc: Any one-off tables
    neurodata_type_inc: DynamicTable
    quantity: '*'
  name: scratch
  quantity: '?'
- doc: The home for ProcessingModules. 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. ProcessingModules can be large and express many data sets from relatively
```

(continues on next page)

(continued from previous page)

```yaml
    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.  'Processing' refers to intermediate analysis of the acquired data
    to make it more amenable to scientific analysis.
  groups:
  - doc: Intermediate analysis of acquired data.
    neurodata_type_inc: ProcessingModule
    quantity: '*'
  name: processing
- doc: 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 should be made 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. 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 linked to a remote library file.
  groups:
  - doc: Stimuli presented during the experiment.
    groups:
    - doc: TimeSeries objects containing data of presented stimuli.
      neurodata_type_inc: TimeSeries
      quantity: '*'
    name: presentation
  - doc: Template stimuli. Timestamps 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.
    groups:
    - doc: TimeSeries objects containing template data of presented stimuli.
      neurodata_type_inc: TimeSeries
      quantity: '*'
    - doc: Images objects containing images of presented stimuli.
      neurodata_type_inc: Images
      quantity: '*'
    name: templates
  name: stimulus
- datasets:
  - doc: Notes about data collection and analysis.
    dtype: text
    name: data_collection
    quantity: '?'
  - doc: General description of the experiment.
    dtype: text
    name: experiment_description
    quantity: '?'
```

(continues on next page)

(continued from previous page)

```
  - dims:
    - num_experimenters
    doc: Name of person(s) who performed the experiment. Can also specify roles of
      different people involved.
    dtype: text
    name: experimenter
    quantity: '?'
    shape:
    -
  - doc: Institution(s) where experiment was performed.
    dtype: text
    name: institution
    quantity: '?'
  - dims:
    - num_keywords
    doc: Terms to search over.
    dtype: text
    name: keywords
    quantity: '?'
    shape:
    -
  - doc: Laboratory where experiment was performed.
    dtype: text
    name: lab
    quantity: '?'
  - doc: Notes about the experiment.
    dtype: text
    name: notes
    quantity: '?'
  - doc: Description of drugs used, including how and when they were administered.
      Anesthesia(s), painkiller(s), etc., plus dosage, concentration, etc.
    dtype: text
    name: pharmacology
    quantity: '?'
  - doc: Experimental protocol, if applicable. e.g., include IACUC protocol number.
    dtype: text
    name: protocol
    quantity: '?'
  - dims:
    - num_publications
    doc: Publication information. PMID, DOI, URL, etc.
    dtype: text
    name: related_publications
    quantity: '?'
    shape:
    -
  - doc: Lab-specific ID for the session.
    dtype: text
    name: session_id
    quantity: '?'
  - doc: Description of slices, including information about preparation thickness,
      orientation, temperature, and bath solution.
```

(continues on next page)

(continued from previous page)

```yaml
    dtype: text
    name: slices
    quantity: '?'
  - attributes:
    - doc: Name of script file.
      dtype: text
      name: file_name
    doc: Script file or link to public source code used to create this NWB file.
    dtype: text
    name: source_script
    quantity: '?'
  - doc: Notes about stimuli, such as how and where they were presented.
    dtype: text
    name: stimulus
    quantity: '?'
  - doc: Narrative description about surgery/surgeries, including date(s) and who
      performed surgery.
    dtype: text
    name: surgery
    quantity: '?'
  - doc: Information about virus(es) used in experiments, including virus ID, source,
      date made, injection location, volume, etc.
    dtype: text
    name: virus
    quantity: '?'
  doc: 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.
    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 (e.g., 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.
  groups:
  - doc: Place-holder than can be extended so that lab-specific meta-data can be placed
      in /general.
    neurodata_type_inc: LabMetaData
    quantity: '*'
  - doc: Description of hardware devices used during experiment, e.g., monitors, ADC
      boards, microscopes, etc.
    groups:
    - doc: Data acquisition devices.
      neurodata_type_inc: Device
      quantity: '*'
    name: devices
    quantity: '?'
```

(continues on next page)

(continued from previous page)

```
- doc: Information about the animal or person from which the data was measured.
  name: subject
  neurodata_type_inc: Subject
  quantity: '?'
- doc: Metadata related to extracellular electrophysiology.
  groups:
  - doc: Physical group of electrodes.
    neurodata_type_inc: ElectrodeGroup
    quantity: '*'
  - datasets:
    - doc: x coordinate of the channel location in the brain (+x is posterior).
      dtype: float32
      name: x
      neurodata_type_inc: VectorData
      quantity: '?'
    - doc: y coordinate of the channel location in the brain (+y is inferior).
      dtype: float32
      name: y
      neurodata_type_inc: VectorData
      quantity: '?'
    - doc: z coordinate of the channel location in the brain (+z is right).
      dtype: float32
      name: z
      neurodata_type_inc: VectorData
      quantity: '?'
    - doc: Impedance of the channel, in ohms.
      dtype: float32
      name: imp
      neurodata_type_inc: VectorData
      quantity: '?'
    - doc: Location of the electrode (channel). Specify the area, layer, comments
        on estimation of area/layer, stereotaxic coordinates if in vivo, etc. Use
        standard atlas names for anatomical regions when possible.
      dtype: text
      name: location
      neurodata_type_inc: VectorData
    - doc: Description of hardware filtering, including the filter name and frequency
        cutoffs.
      dtype: text
      name: filtering
      neurodata_type_inc: VectorData
      quantity: '?'
    - doc: Reference to the ElectrodeGroup this electrode is a part of.
      dtype:
        reftype: object
        target_type: ElectrodeGroup
      name: group
      neurodata_type_inc: VectorData
    - doc: Name of the ElectrodeGroup this electrode is a part of.
      dtype: text
      name: group_name
      neurodata_type_inc: VectorData
```

(continues on next page)

(continued from previous page)

```
      - doc: x coordinate in electrode group
        dtype: float32
        name: rel_x
        neurodata_type_inc: VectorData
        quantity: '?'
      - doc: y coordinate in electrode group
        dtype: float32
        name: rel_y
        neurodata_type_inc: VectorData
        quantity: '?'
      - doc: z coordinate in electrode group
        dtype: float32
        name: rel_z
        neurodata_type_inc: VectorData
        quantity: '?'
      - doc: Description of the reference electrode and/or reference scheme used for
          this electrode, e.g., "stainless steel skull screw" or "online common average
          referencing".
        dtype: text
        name: reference
        neurodata_type_inc: VectorData
        quantity: '?'
      doc: A table of all electrodes (i.e. channels) used for recording.
      name: electrodes
      neurodata_type_inc: DynamicTable
      quantity: '?'
    name: extracellular_ephys
    quantity: '?'
  - datasets:
    - doc: '[DEPRECATED] Use IntracellularElectrode.filtering instead. 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.'
      dtype: text
      name: filtering
      quantity: '?'
    doc: Metadata related to intracellular electrophysiology.
    groups:
    - doc: An intracellular electrode.
      neurodata_type_inc: IntracellularElectrode
      quantity: '*'
    - doc: '[DEPRECATED] Table used to group different PatchClampSeries. SweepTable
        is being replaced by IntracellularRecordingsTable and SimultaneousRecordingsTable
        tabels. Additional SequentialRecordingsTable, RepetitionsTable and ExperimentalConditions
        tables provide enhanced support for experiment metadata.'
      name: sweep_table
      neurodata_type_inc: SweepTable
      quantity: '?'
    - doc: A table to group together a stimulus and response from a single electrode
        and a single simultaneous recording. Each row in the table represents a single
        recording consisting typically of a stimulus and a corresponding response.
```

(continues on next page)

(continued from previous page)

```
        In some cases, however, only a stimulus or a response are recorded as as part
        of an experiment. In this case both, the stimulus and response will point
        to the same TimeSeries while the idx_start and count of the invalid column
        will be set to -1, thus, indicating that no values have been recorded for
        the stimulus or response, respectively. Note, a recording MUST contain at
        least a stimulus or a response. Typically the stimulus and response are␣
↪PatchClampSeries.
        However, the use of AD/DA channels that are not associated to an electrode
        is also common in intracellular electrophysiology, in which case other TimeSeries
        may be used.
      name: intracellular_recordings
      neurodata_type_inc: IntracellularRecordingsTable
      quantity: '?'
    - doc: A table for grouping different intracellular recordings from the␣
↪IntracellularRecordingsTable
        table together that were recorded simultaneously from different electrodes
      name: simultaneous_recordings
      neurodata_type_inc: SimultaneousRecordingsTable
      quantity: '?'
    - doc: A table for grouping different sequential recordings from the␣
↪SimultaneousRecordingsTable
        table together. This is typically used to group together sequential recordings
        where the a sequence of stimuli of the same type with varying parameters have
        been presented in a sequence.
      name: sequential_recordings
      neurodata_type_inc: SequentialRecordingsTable
      quantity: '?'
    - doc: A table for grouping different sequential intracellular recordings together.
        With each SequentialRecording typically representing a particular type of
        stimulus, the RepetitionsTable table is typically used to group sets of stimuli
        applied in sequence.
      name: repetitions
      neurodata_type_inc: RepetitionsTable
      quantity: '?'
    - doc: A table for grouping different intracellular recording repetitions together
        that belong to the same experimental experimental_conditions.
      name: experimental_conditions
      neurodata_type_inc: ExperimentalConditionsTable
      quantity: '?'
    name: intracellular_ephys
    quantity: '?'
  - doc: Metadata describing optogenetic stimuluation.
    groups:
    - doc: An optogenetic stimulation site.
      neurodata_type_inc: OptogeneticStimulusSite
      quantity: '*'
    name: optogenetics
    quantity: '?'
  - doc: Metadata related to optophysiology.
    groups:
    - doc: An imaging plane.
      neurodata_type_inc: ImagingPlane
```

(continues on next page)

(continued from previous page)

```yaml
      quantity: '*'
    name: optophysiology
    quantity: '?'
  name: general
- doc: Experimental intervals, whether that be logically distinct sub-experiments
    having a particular scientific goal, trials (see trials subgroup) during an experiment,
    or epochs (see epochs subgroup) deriving from analysis of data.
  groups:
  - doc: Divisions in time marking experimental stages or sub-divisions of a single
      recording session.
    name: epochs
    neurodata_type_inc: TimeIntervals
    quantity: '?'
  - doc: Repeated experimental events that have a logical grouping.
    name: trials
    neurodata_type_inc: TimeIntervals
    quantity: '?'
  - doc: Time intervals that should be removed from analysis.
    name: invalid_times
    neurodata_type_inc: TimeIntervals
    quantity: '?'
  - doc: Optional additional table(s) for describing other experimental time intervals.
    neurodata_type_inc: TimeIntervals
    quantity: '*'
  name: intervals
  quantity: '?'
- doc: Data about sorted spike units.
  name: units
  neurodata_type_inc: Units
  quantity: '?'
name: root
neurodata_type_def: NWBFile
neurodata_type_inc: NWBContainer
```

5.6.3 LabMetaData

Extends: *NWBContainer*

Description: see Section 4.5.3

YAML Specification:

```yaml
doc: Lab-specific meta-data.
neurodata_type_def: LabMetaData
neurodata_type_inc: NWBContainer
```

5.6.4 Subject

Extends: *NWBContainer*

Description: see Section 4.5.4

YAML Specification:

```yaml
datasets:
- attributes:
  - default_value: birth
    doc: Age is with reference to this event. Can be 'birth' or 'gestational'. If
      reference is omitted, 'birth' is implied.
    dtype: text
    name: reference
    required: false
  doc: Age of subject. Can be supplied instead of 'date_of_birth'.
  dtype: text
  name: age
  quantity: '?'
- doc: Date of birth of subject. Can be supplied instead of 'age'.
  dtype: isodatetime
  name: date_of_birth
  quantity: '?'
- doc: Description of subject and where subject came from (e.g., breeder, if animal).
  dtype: text
  name: description
  quantity: '?'
- doc: Genetic strain. If absent, assume Wild Type (WT).
  dtype: text
  name: genotype
  quantity: '?'
- doc: Gender of subject.
  dtype: text
  name: sex
  quantity: '?'
- doc: Species of subject.
  dtype: text
  name: species
  quantity: '?'
- doc: Strain of subject.
  dtype: text
  name: strain
  quantity: '?'
- doc: ID of animal/person used/participating in experiment (lab convention).
  dtype: text
  name: subject_id
  quantity: '?'
- doc: Weight at time of experiment, at time of surgery and at other important times.
  dtype: text
  name: weight
  quantity: '?'
doc: Information about the animal or person from which the data was measured.
neurodata_type_def: Subject
neurodata_type_inc: NWBContainer
```

5.7 Miscellaneous neurodata_types.

Miscellaneous types.

5.7.1 AbstractFeatureSeries

Extends: *TimeSeries*

Description: see Section 4.6.1

YAML Specification:

```yaml
attributes:
- default_value: no description
  doc: Description of the time series.
  dtype: text
  name: description
  required: false
- default_value: no comments
  doc: 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.
  dtype: text
  name: comments
  required: false
datasets:
- attributes:
  - default_value: see 'feature_units'
    doc: Since there can be different units for different features, store the units
      in 'feature_units'. The default value for this attribute is "see 'feature_units'".
    dtype: text
    name: unit
    required: false
  - default_value: 1.0
    doc: Scalar to multiply each element in data to convert it to the specified 'unit'.
      If the data are stored in acquisition system units or other units that require
      a conversion to be interpretable, multiply the data by 'conversion' to convert
      the data to the specified 'unit'. e.g. if the data acquisition system stores
      values in this object as signed 16-bit integers (int16 range -32,768 to 32,767)
      that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system
      gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition
      values to recorded volts is 2.5/32768/8000 = 9.5367e-9.
    dtype: float32
    name: conversion
    required: false
  - default_value: 0.0
    doc: Scalar to add to the data after scaling by 'conversion' to finalize its coercion
      to the specified 'unit'. Two common examples of this include (a) data stored
      in an unsigned type that requires a shift after scaling to re-center the data,
      and (b) specialized recording devices that naturally cause a scalar offset with
      respect to the true units.
    dtype: float32
    name: offset
```

(continues on next page)

(continued from previous page)

```yaml
    required: false
  - default_value: -1.0
    doc: 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 -1.0.
    dtype: float32
    name: resolution
    required: false
  - doc: Optionally describe the continuity of the data. Can be "continuous", "instantaneous",
      or "step". For example, a voltage trace would be "continuous", because samples
      are recorded from a continuous process. An array of lick times would be "instantaneous",
      because the data represents distinct moments in time. Times of image presentations
      would be "step" because the picture remains the same until the next timepoint.
      This field is optional, but is useful in providing information about the underlying
      data. It may inform the way this data is interpreted, the way it is visualized,
      and what analysis methods are applicable.
    dtype: text
    name: continuity
    required: false
  dims:
  - - num_times
  - - num_times
    - num_features
  doc: Values of each feature at each time.
  dtype: numeric
  name: data
  shape:
  - -
  - -
    -
- dims:
  - num_features
  doc: Units of each feature.
  dtype: text
  name: feature_units
  quantity: '?'
  shape:
  -
- dims:
  - num_features
  doc: Description of the features represented in TimeSeries::data.
  dtype: text
  name: features
  shape:
  -
- attributes:
  - doc: Sampling rate, in Hz.
    dtype: float32
    name: rate
  - doc: Unit of measurement for time, which is fixed to 'seconds'.
```

(continues on next page)

(continued from previous page)

```yaml
    dtype: text
    name: unit
    value: seconds
  doc: Timestamp of the first sample in seconds. When timestamps are uniformly spaced,
    the timestamp of the first sample can be specified and all subsequent ones calculated
    from the sampling rate attribute.
  dtype: float64
  name: starting_time
  quantity: '?'
- attributes:
  - doc: Value is '1'
    dtype: int32
    name: interval
    value: 1
  - doc: Unit of measurement for timestamps, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  dims:
  - num_times
  doc: Timestamps for samples stored in data, in seconds, relative to the common experiment
    master-clock stored in NWBFile.timestamps_reference_time.
  dtype: float64
  name: timestamps
  quantity: '?'
  shape:
  -
- dims:
  - num_times
  doc: Numerical labels that apply to each time point in data for the purpose of querying
    and slicing data by these values. If present, the length of this array should
    be the same size as the first dimension of data.
  dtype: uint8
  name: control
  quantity: '?'
  shape:
  -
- dims:
  - num_control_values
  doc: Description of each control value. Must be present if control is present. If
    present, control_description[0] should describe time points where control == 0.
  dtype: text
  name: control_description
  quantity: '?'
  shape:
  -
doc: 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
```

(continues on next page)

(continued from previous page)

```
  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. This is useful when storing the raw stimulus is impractical.
groups:
- doc: 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.
  name: sync
  quantity: '?'
neurodata_type_def: AbstractFeatureSeries
neurodata_type_inc: TimeSeries
```

5.7.2 AnnotationSeries

Extends: *TimeSeries*

Description: see Section 4.6.2

YAML Specification:


```yaml
attributes:
- default_value: no description
  doc: Description of the time series.
  dtype: text
  name: description
  required: false
- default_value: no comments
  doc: 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.
  dtype: text
  name: comments
  required: false
datasets:
- attributes:
  - doc: Smallest meaningful difference between values in data. Annotations have no
      units, so the value is fixed to -1.0.
    dtype: float32
    name: resolution
    value: -1.0
  - doc: Base unit of measurement for working with the data. Annotations have no units,
      so the value is fixed to 'n/a'.
    dtype: text
    name: unit
    value: n/a
  - default_value: 1.0
    doc: Scalar to multiply each element in data to convert it to the specified 'unit'.
      If the data are stored in acquisition system units or other units that require
      a conversion to be interpretable, multiply the data by 'conversion' to convert
      the data to the specified 'unit'. e.g. if the data acquisition system stores
      values in this object as signed 16-bit integers (int16 range -32,768 to 32,767)
      that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system
      gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition
      values to recorded volts is 2.5/32768/8000 = 9.5367e-9.
    dtype: float32
    name: conversion
    required: false
  - default_value: 0.0
    doc: Scalar to add to the data after scaling by 'conversion' to finalize its coercion
      to the specified 'unit'. Two common examples of this include (a) data stored
      in an unsigned type that requires a shift after scaling to re-center the data,
      and (b) specialized recording devices that naturally cause a scalar offset with
      respect to the true units.
    dtype: float32
    name: offset
    required: false
```


(continues on next page)

(continued from previous page)

```
  - doc: Optionally describe the continuity of the data. Can be "continuous", "instantaneous",
      or "step". For example, a voltage trace would be "continuous", because samples
      are recorded from a continuous process. An array of lick times would be "instantaneous",
      because the data represents distinct moments in time. Times of image presentations
      would be "step" because the picture remains the same until the next timepoint.
      This field is optional, but is useful in providing information about the underlying
      data. It may inform the way this data is interpreted, the way it is visualized,
      and what analysis methods are applicable.
    dtype: text
    name: continuity
    required: false
  dims:
  - num_times
  doc: Annotations made during an experiment.
  dtype: text
  name: data
  shape:
  -
- attributes:
  - doc: Sampling rate, in Hz.
    dtype: float32
    name: rate
  - doc: Unit of measurement for time, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  doc: Timestamp of the first sample in seconds. When timestamps are uniformly spaced,
    the timestamp of the first sample can be specified and all subsequent ones calculated
    from the sampling rate attribute.
  dtype: float64
  name: starting_time
  quantity: '?'
- attributes:
  - doc: Value is '1'
    dtype: int32
    name: interval
    value: 1
  - doc: Unit of measurement for timestamps, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  dims:
  - num_times
  doc: Timestamps for samples stored in data, in seconds, relative to the common experiment
    master-clock stored in NWBFile.timestamps_reference_time.
  dtype: float64
  name: timestamps
  quantity: '?'
  shape:
```

(continues on next page)

(continued from previous page)

```yaml
  -
- dims:
  - num_times
  doc: Numerical labels that apply to each time point in data for the purpose of querying
    and slicing data by these values. If present, the length of this array should
    be the same size as the first dimension of data.
  dtype: uint8
  name: control
  quantity: '?'
  shape:
  -
- dims:
  - num_control_values
  doc: Description of each control value. Must be present if control is present. If
    present, control_description[0] should describe time points where control == 0.
  dtype: text
  name: control_description
  quantity: '?'
  shape:
  -
doc: Stores user annotations made during an experiment. The 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.
groups:
- doc: 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.
  name: sync
  quantity: '?'
neurodata_type_def: AnnotationSeries
neurodata_type_inc: TimeSeries
```

5.7.3 IntervalSeries

Extends: *TimeSeries*

Description: see Section 4.6.3

YAML Specification:

```yaml
attributes:
- default_value: no description
  doc: Description of the time series.
  dtype: text
  name: description
  required: false
- default_value: no comments
  doc: 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.
  dtype: text
  name: comments
  required: false
datasets:
- attributes:
  - doc: Smallest meaningful difference between values in data. Annotations have no
      units, so the value is fixed to -1.0.
    dtype: float32
    name: resolution
    value: -1.0
  - doc: Base unit of measurement for working with the data. Annotations have no units,
      so the value is fixed to 'n/a'.
    dtype: text
    name: unit
    value: n/a
  - default_value: 1.0
    doc: Scalar to multiply each element in data to convert it to the specified 'unit'.
      If the data are stored in acquisition system units or other units that require
      a conversion to be interpretable, multiply the data by 'conversion' to convert
      the data to the specified 'unit'. e.g. if the data acquisition system stores
      values in this object as signed 16-bit integers (int16 range -32,768 to 32,767)
      that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system
      gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition
      values to recorded volts is 2.5/32768/8000 = 9.5367e-9.
    dtype: float32
    name: conversion
    required: false
  - default_value: 0.0
    doc: Scalar to add to the data after scaling by 'conversion' to finalize its coercion
      to the specified 'unit'. Two common examples of this include (a) data stored
      in an unsigned type that requires a shift after scaling to re-center the data,
      and (b) specialized recording devices that naturally cause a scalar offset with
      respect to the true units.
    dtype: float32
    name: offset
    required: false
```

(continues on next page)

(continued from previous page)

```
  - doc: Optionally describe the continuity of the data. Can be "continuous", "instantaneous",
      or "step". For example, a voltage trace would be "continuous", because samples
      are recorded from a continuous process. An array of lick times would be "instantaneous",
      because the data represents distinct moments in time. Times of image presentations
      would be "step" because the picture remains the same until the next timepoint.
      This field is optional, but is useful in providing information about the underlying
      data. It may inform the way this data is interpreted, the way it is visualized,
      and what analysis methods are applicable.
    dtype: text
    name: continuity
    required: false
  dims:
  - num_times
  doc: Use values >0 if interval started, <0 if interval ended.
  dtype: int8
  name: data
  shape:
  -
- attributes:
  - doc: Sampling rate, in Hz.
    dtype: float32
    name: rate
  - doc: Unit of measurement for time, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  doc: Timestamp of the first sample in seconds. When timestamps are uniformly spaced,
    the timestamp of the first sample can be specified and all subsequent ones calculated
    from the sampling rate attribute.
  dtype: float64
  name: starting_time
  quantity: '?'
- attributes:
  - doc: Value is '1'
    dtype: int32
    name: interval
    value: 1
  - doc: Unit of measurement for timestamps, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  dims:
  - num_times
  doc: Timestamps for samples stored in data, in seconds, relative to the common experiment
    master-clock stored in NWBFile.timestamps_reference_time.
  dtype: float64
  name: timestamps
  quantity: '?'
  shape:
```

(continues on next page)

(continued from previous page)

```
  -
- dims:
  - num_times
  doc: Numerical labels that apply to each time point in data for the purpose of querying
    and slicing data by these values. If present, the length of this array should
    be the same size as the first dimension of data.
  dtype: uint8
  name: control
  quantity: '?'
  shape:
  -
- dims:
  - num_control_values
  doc: Description of each control value. Must be present if control is present. If
    present, control_description[0] should describe time points where control == 0.
  dtype: text
  name: control_description
  quantity: '?'
  shape:
  -
doc: 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.
groups:
- doc: 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.
  name: sync
  quantity: '?'
neurodata_type_def: IntervalSeries
neurodata_type_inc: TimeSeries
```

5.7.4 DecompositionSeries

Extends: *TimeSeries*

Description: see Section 4.6.4

YAML Specification:

```yaml
attributes:
- default_value: no description
  doc: Description of the time series.
  dtype: text
  name: description
  required: false
- default_value: no comments
  doc: 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.
  dtype: text
  name: comments
  required: false
datasets:
- attributes:
  - default_value: no unit
    doc: Base unit of measurement for working with the data. Actual stored values
      are not necessarily stored in these units. To access the data in these units,
      multiply 'data' by 'conversion'.
    dtype: text
    name: unit
    required: false
  - default_value: 1.0
    doc: Scalar to multiply each element in data to convert it to the specified 'unit'.
      If the data are stored in acquisition system units or other units that require
      a conversion to be interpretable, multiply the data by 'conversion' to convert
      the data to the specified 'unit'. e.g. if the data acquisition system stores
      values in this object as signed 16-bit integers (int16 range -32,768 to 32,767)
      that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system
      gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition
      values to recorded volts is 2.5/32768/8000 = 9.5367e-9.
    dtype: float32
    name: conversion
    required: false
  - default_value: 0.0
    doc: Scalar to add to the data after scaling by 'conversion' to finalize its coercion
      to the specified 'unit'. Two common examples of this include (a) data stored
      in an unsigned type that requires a shift after scaling to re-center the data,
      and (b) specialized recording devices that naturally cause a scalar offset with
      respect to the true units.
    dtype: float32
    name: offset
    required: false
  - default_value: -1.0
    doc: 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
```

(continues on next page)

(continued from previous page)

```
      number if signal noise is known to be present. If unknown, use -1.0.
    dtype: float32
    name: resolution
    required: false
  - doc: Optionally describe the continuity of the data. Can be "continuous",
↪"instantaneous",
      or "step". For example, a voltage trace would be "continuous", because samples
      are recorded from a continuous process. An array of lick times would be
↪"instantaneous",
      because the data represents distinct moments in time. Times of image presentations
      would be "step" because the picture remains the same until the next timepoint.
      This field is optional, but is useful in providing information about the underlying
      data. It may inform the way this data is interpreted, the way it is visualized,
      and what analysis methods are applicable.
    dtype: text
    name: continuity
    required: false
  dims:
  - num_times
  - num_channels
  - num_bands
  doc: Data decomposed into frequency bands.
  dtype: numeric
  name: data
  shape:
  -
  -
  -
- doc: The metric used, e.g. phase, amplitude, power.
  dtype: text
  name: metric
- doc: DynamicTableRegion pointer to the channels that this decomposition series was
    generated from.
  name: source_channels
  neurodata_type_inc: DynamicTableRegion
  quantity: '?'
- attributes:
  - doc: Sampling rate, in Hz.
    dtype: float32
    name: rate
  - doc: Unit of measurement for time, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  doc: Timestamp of the first sample in seconds. When timestamps are uniformly spaced,
    the timestamp of the first sample can be specified and all subsequent ones calculated
    from the sampling rate attribute.
  dtype: float64
  name: starting_time
  quantity: '?'
- attributes:
  - doc: Value is '1'
```

(continues on next page)

(continued from previous page)

```
    dtype: int32
    name: interval
    value: 1
  - doc: Unit of measurement for timestamps, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  dims:
  - num_times
  doc: Timestamps for samples stored in data, in seconds, relative to the common experiment
    master-clock stored in NWBFile.timestamps_reference_time.
  dtype: float64
  name: timestamps
  quantity: '?'
  shape:
  -
- dims:
  - num_times
  doc: Numerical labels that apply to each time point in data for the purpose of querying
    and slicing data by these values. If present, the length of this array should
    be the same size as the first dimension of data.
  dtype: uint8
  name: control
  quantity: '?'
  shape:
  -
- dims:
  - num_control_values
  doc: Description of each control value. Must be present if control is present. If
    present, control_description[0] should describe time points where control == 0.
  dtype: text
  name: control_description
  quantity: '?'
  shape:
  -
doc: Spectral analysis of a time series, e.g. of an LFP or a speech signal.
groups:
- datasets:
  - doc: Name of the band, e.g. theta.
    dtype: text
    name: band_name
    neurodata_type_inc: VectorData
  - dims:
    - num_bands
    - low, high
    doc: Low and high limit of each band in Hz. If it is a Gaussian filter, use 2
      SD on either side of the center.
    dtype: float32
    name: band_limits
    neurodata_type_inc: VectorData
    shape:
```

(continues on next page)

(continued from previous page)

```yaml
    -
    - 2
  - dims:
    - num_bands
    doc: The mean Gaussian filters, in Hz.
    dtype: float32
    name: band_mean
    neurodata_type_inc: VectorData
    shape:
    -
  - dims:
    - num_bands
    doc: The standard deviation of Gaussian filters, in Hz.
    dtype: float32
    name: band_stdev
    neurodata_type_inc: VectorData
    shape:
    -
  doc: Table for describing the bands that this series was generated from. There should
    be one row in this table for each band.
  name: bands
  neurodata_type_inc: DynamicTable
- doc: 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.
  name: sync
  quantity: '?'
links:
- doc: Link to TimeSeries object that this data was calculated from. Metadata about
    electrodes and their position can be read from that ElectricalSeries so it is
    not necessary to store that information here.
  name: source_timeseries
  quantity: '?'
  target_type: TimeSeries
neurodata_type_def: DecompositionSeries
neurodata_type_inc: TimeSeries
```

5.7.5 Units

Extends: DynamicTable

Description: see Section 4.6.5

YAML Specification:


```yaml
attributes:
- dims:
  - num_columns
  doc: The names of the columns in this table. This should be used to specify an order
    to the columns.
  dtype: text
  name: colnames
  shape:
  -
- doc: Description of what is in this dynamic table.
  dtype: text
  name: description
datasets:
- doc: Index into the spike_times dataset.
  name: spike_times_index
  neurodata_type_inc: VectorIndex
  quantity: '?'
- attributes:
  - doc: The smallest possible difference between two spike times. Usually 1 divided
      by the acquisition sampling rate from which spike times were extracted, but
      could be larger if the acquisition time series was downsampled or smaller if
      the acquisition time series was smoothed/interpolated and it is possible for
      the spike time to be between samples.
    dtype: float64
    name: resolution
    required: false
  doc: Spike times for each unit in seconds.
  dtype: float64
  name: spike_times
  neurodata_type_inc: VectorData
  quantity: '?'
- doc: Index into the obs_intervals dataset.
  name: obs_intervals_index
  neurodata_type_inc: VectorIndex
  quantity: '?'
- dims:
  - num_intervals
  - start|end
  doc: Observation intervals for each unit.
  dtype: float64
  name: obs_intervals
  neurodata_type_inc: VectorData
  quantity: '?'
  shape:
  -
  - 2
```


(continues on next page)

(continued from previous page)

```
- doc: Index into electrodes.
  name: electrodes_index
  neurodata_type_inc: VectorIndex
  quantity: '?'
- doc: Electrode that each spike unit came from, specified using a DynamicTableRegion.
  name: electrodes
  neurodata_type_inc: DynamicTableRegion
  quantity: '?'
- doc: Electrode group that each spike unit came from.
  dtype:
    reftype: object
    target_type: ElectrodeGroup
  name: electrode_group
  neurodata_type_inc: VectorData
  quantity: '?'
- attributes:
  - doc: Sampling rate, in hertz.
    dtype: float32
    name: sampling_rate
    required: false
  - doc: Unit of measurement. This value is fixed to 'volts'.
    dtype: text
    name: unit
    value: volts
  dims:
  - - num_units
    - num_samples
  - - num_units
    - num_samples
    - num_electrodes
  doc: Spike waveform mean for each spike unit.
  dtype: float32
  name: waveform_mean
  neurodata_type_inc: VectorData
  quantity: '?'
  shape:
  - -
    -
  - -
    -
    -
- attributes:
  - doc: Sampling rate, in hertz.
    dtype: float32
    name: sampling_rate
    required: false
  - doc: Unit of measurement. This value is fixed to 'volts'.
    dtype: text
    name: unit
    value: volts
  dims:
  - - num_units
```

(continues on next page)

(continued from previous page)

```
    - num_samples
  - - num_units
    - num_samples
    - num_electrodes
  doc: Spike waveform standard deviation for each spike unit.
  dtype: float32
  name: waveform_sd
  neurodata_type_inc: VectorData
  quantity: '?'
  shape:
  - -
    -
  - -
    -
    -
- attributes:
  - doc: Sampling rate, in hertz.
    dtype: float32
    name: sampling_rate
    required: false
  - doc: Unit of measurement. This value is fixed to 'volts'.
    dtype: text
    name: unit
    value: volts
  dims:
  - num_waveforms
  - num_samples
  doc: Individual waveforms for each spike on each electrode. This is a doubly indexed
    column. The 'waveforms_index' column indexes which waveforms in this column belong
    to the same spike event for a given unit, where each waveform was recorded from
    a different electrode. The 'waveforms_index_index' column indexes the 'waveforms_
↪index'
    column to indicate which spike events belong to a given unit. For example, if
    the 'waveforms_index_index' column has values [2, 5, 6], then the first 2 elements
    of the 'waveforms_index' column correspond to the 2 spike events of the first
    unit, the next 3 elements of the 'waveforms_index' column correspond to the 3
    spike events of the second unit, and the next 1 element of the 'waveforms_index'
    column corresponds to the 1 spike event of the third unit. If the 'waveforms_index'
    column has values [3, 6, 8, 10, 12, 13], then the first 3 elements of the 'waveforms'
    column contain the 3 spike waveforms that were recorded from 3 different electrodes
    for the first spike time of the first unit. See
    https://nwb-schema.readthedocs.io/en/stable/format_description.html#doubly-ragged-
↪arrays
    for a graphical representation of this example. When there is only one electrode
    for each unit (i.e., each spike time is associated with a single waveform), then
    the 'waveforms_index' column will have values 1, 2, ..., N, where N is the number
    of spike events. The number of electrodes for each spike event should be the same
    within a given unit. The 'electrodes' column should be used to indicate which
    electrodes are associated with each unit, and the order of the waveforms within
    a given unit x spike event should be in the same order as the electrodes referenced
    in the 'electrodes' column of this table. The number of samples for each waveform
    must be the same.
```

(continues on next page)

(continued from previous page)

```yaml
  dtype: numeric
  name: waveforms
  neurodata_type_inc: VectorData
  quantity: '?'
  shape:
  -
  -
- doc: Index into the waveforms dataset. One value for every spike event. See 'waveforms'
    for more detail.
  name: waveforms_index
  neurodata_type_inc: VectorIndex
  quantity: '?'
- doc: Index into the waveforms_index dataset. One value for every unit (row in the
    table). See 'waveforms' for more detail.
  name: waveforms_index_index
  neurodata_type_inc: VectorIndex
  quantity: '?'
- dims:
  - num_rows
  doc: Array of unique identifiers for the rows of this dynamic table.
  dtype: int
  name: id
  neurodata_type_inc: ElementIdentifiers
  shape:
  -
- doc: Vector columns, including index columns, of this dynamic table.
  neurodata_type_inc: VectorData
  quantity: '*'
default_name: Units
doc: Data about spiking units. Event times of observed units (e.g. cell, synapse,
  etc.) should be concatenated and stored in spike_times.
neurodata_type_def: Units
neurodata_type_inc: DynamicTable
```

5.8 Behavior

This source module contains neurodata_types for behavior data.

5.8.1 SpatialSeries

Extends: *TimeSeries*

Description: see Section 4.7.1

YAML Specification:


```yaml
attributes:
- default_value: no description
  doc: Description of the time series.
  dtype: text
  name: description
  required: false
- default_value: no comments
  doc: 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.
  dtype: text
  name: comments
  required: false
datasets:
- attributes:
  - default_value: meters
    doc: Base unit of measurement for working with the data. The default value is
      'meters'. Actual stored values are not necessarily stored in these units. To
      access the data in these units, multiply 'data' by 'conversion' and add 'offset'.
    dtype: text
    name: unit
    required: false
  - default_value: 1.0
    doc: Scalar to multiply each element in data to convert it to the specified 'unit'.
      If the data are stored in acquisition system units or other units that require
      a conversion to be interpretable, multiply the data by 'conversion' to convert
      the data to the specified 'unit'. e.g. if the data acquisition system stores
      values in this object as signed 16-bit integers (int16 range -32,768 to 32,767)
      that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system
      gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition
      values to recorded volts is 2.5/32768/8000 = 9.5367e-9.
    dtype: float32
    name: conversion
    required: false
  - default_value: 0.0
    doc: Scalar to add to the data after scaling by 'conversion' to finalize its coercion
      to the specified 'unit'. Two common examples of this include (a) data stored
      in an unsigned type that requires a shift after scaling to re-center the data,
      and (b) specialized recording devices that naturally cause a scalar offset with
      respect to the true units.
    dtype: float32
```


(continues on next page)

(continued from previous page)

```yaml
      name: offset
      required: false
    - default_value: -1.0
      doc: 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 -1.0.
      dtype: float32
      name: resolution
      required: false
    - doc: Optionally describe the continuity of the data. Can be "continuous", "instantaneous",
        or "step". For example, a voltage trace would be "continuous", because samples
        are recorded from a continuous process. An array of lick times would be "instantaneous",
        because the data represents distinct moments in time. Times of image presentations
        would be "step" because the picture remains the same until the next timepoint.
        This field is optional, but is useful in providing information about the underlying
        data. It may inform the way this data is interpreted, the way it is visualized,
        and what analysis methods are applicable.
      dtype: text
      name: continuity
      required: false
    dims:
    - - num_times
    - - num_times
      - x
    - - num_times
      - x,y
    - - num_times
      - x,y,z
    doc: 1-D or 2-D array storing position or direction relative to some reference frame.
    dtype: numeric
    name: data
    shape:
    - - null
    - - null
      - 1
    - - null
      - 2
    - - null
      - 3
  - doc: Description defining what exactly 'straight-ahead' means.
    dtype: text
    name: reference_frame
    quantity: '?'
  - attributes:
    - doc: Sampling rate, in Hz.
      dtype: float32
      name: rate
    - doc: Unit of measurement for time, which is fixed to 'seconds'.
      dtype: text
      name: unit
```

(continues on next page)

(continued from previous page)

```
    value: seconds
  doc: Timestamp of the first sample in seconds. When timestamps are uniformly spaced,
    the timestamp of the first sample can be specified and all subsequent ones calculated
    from the sampling rate attribute.
  dtype: float64
  name: starting_time
  quantity: '?'
- attributes:
  - doc: Value is '1'
    dtype: int32
    name: interval
    value: 1
  - doc: Unit of measurement for timestamps, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  dims:
  - num_times
  doc: Timestamps for samples stored in data, in seconds, relative to the common experiment
    master-clock stored in NWBFile.timestamps_reference_time.
  dtype: float64
  name: timestamps
  quantity: '?'
  shape:
  -
- dims:
  - num_times
  doc: Numerical labels that apply to each time point in data for the purpose of querying
    and slicing data by these values. If present, the length of this array should
    be the same size as the first dimension of data.
  dtype: uint8
  name: control
  quantity: '?'
  shape:
  -
- dims:
  - num_control_values
  doc: Description of each control value. Must be present if control is present. If
    present, control_description[0] should describe time points where control == 0.
  dtype: text
  name: control_description
  quantity: '?'
  shape:
  -
doc: "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.
```

(continues on next page)

(continued from previous page)

```
  The unit of data will indicate how to interpret SpatialSeries values."
groups:
- doc: 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.
  name: sync
  quantity: '?'
neurodata_type_def: SpatialSeries
neurodata_type_inc: TimeSeries
```

5.8.2 BehavioralEpochs

Extends: *NWBDataInterface*

Description: see Section 4.7.2

YAML Specification:

```yaml
default_name: BehavioralEpochs
doc: TimeSeries for storing behavioral 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.
groups:
- doc: IntervalSeries object containing start and stop times of epochs.
  neurodata_type_inc: IntervalSeries
  quantity: '*'
neurodata_type_def: BehavioralEpochs
neurodata_type_inc: NWBDataInterface
```

5.8.3 BehavioralEvents

Extends: *NWBDataInterface*

Description: see Section 4.7.3

YAML Specification:


```yaml
default_name: BehavioralEvents
doc: TimeSeries for storing behavioral events. See description of <a href="#BehavioralEpochs">BehavioralEpochs</a>
  for more details.
groups:
- doc: TimeSeries object containing behavioral events.
  neurodata_type_inc: TimeSeries
  quantity: '*'
neurodata_type_def: BehavioralEvents
neurodata_type_inc: NWBDataInterface
```

5.8.4 BehavioralTimeSeries

Extends: *NWBDataInterface*

Description: see Section 4.7.4

YAML Specification:

```yaml
default_name: BehavioralTimeSeries
doc: TimeSeries for storing Behavoioral time series data. See description of <a href="#BehavioralEpochs">BehavioralEpochs</a>
  for more details.
groups:
- doc: TimeSeries object containing continuous behavioral data.
  neurodata_type_inc: TimeSeries
  quantity: '*'
neurodata_type_def: BehavioralTimeSeries
neurodata_type_inc: NWBDataInterface
```

5.8.5 PupilTracking

Extends: *NWBDataInterface*

Description: see Section 4.7.5

YAML Specification:

```yaml
default_name: PupilTracking
doc: Eye-tracking data, representing pupil size.
groups:
- doc: TimeSeries object containing time series data on pupil size.
  neurodata_type_inc: TimeSeries
  quantity: +
neurodata_type_def: PupilTracking
neurodata_type_inc: NWBDataInterface
```

5.8.6 EyeTracking

Extends: *NWBDataInterface*

Description: see Section 4.7.6

YAML Specification:

```yaml
default_name: EyeTracking
doc: Eye-tracking data, representing direction of gaze.
groups:
- doc: SpatialSeries object containing data measuring direction of gaze.
  neurodata_type_inc: SpatialSeries
  quantity: '*'
neurodata_type_def: EyeTracking
neurodata_type_inc: NWBDataInterface
```

5.8.7 CompassDirection

Extends: *NWBDataInterface*

Description: see Section 4.7.7

YAML Specification:


```yaml
default_name: CompassDirection
doc: 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.
groups:
- doc: SpatialSeries object containing direction of gaze travel.
  neurodata_type_inc: SpatialSeries
  quantity: '*'
neurodata_type_def: CompassDirection
neurodata_type_inc: NWBDataInterface
```

5.8.8 Position

Extends: *NWBDataInterface*

Description: see Section 4.7.8

YAML Specification:


```yaml
default_name: Position
doc: Position data, whether along the x, x/y or x/y/z axis.
groups:
- doc: SpatialSeries object containing position data.
  neurodata_type_inc: SpatialSeries
  quantity: +
neurodata_type_def: Position
neurodata_type_inc: NWBDataInterface
```

5.9 Extracellular electrophysiology

This source module contains neurodata_types for extracellular electrophysiology data.

5.9.1 ElectricalSeries

Extends: *TimeSeries*

Description: see Section 4.8.1

YAML Specification:

```yaml
attributes:
- doc: Filtering applied to all channels of the data. For example, if this ElectricalSeries
    represents high-pass-filtered data (also known as AP Band), then this value could
    be "High-pass 4-pole Bessel filter at 500 Hz". If this ElectricalSeries represents
    low-pass-filtered LFP data and the type of filter is unknown, then this value
    could be "Low-pass filter at 300 Hz". If a non-standard filter type is used, provide
    as much detail about the filter properties as possible.
  dtype: text
  name: filtering
  required: false
- default_value: no description
  doc: Description of the time series.
  dtype: text
  name: description
  required: false
- default_value: no comments
  doc: 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.
  dtype: text
  name: comments
  required: false
datasets:
- attributes:
  - doc: Base unit of measurement for working with the data. This value is fixed to
      'volts'. Actual stored values are not necessarily stored in these units. To
      access the data in these units, multiply 'data' by 'conversion', followed by
      'channel_conversion' (if present), and then add 'offset'.
    dtype: text
    name: unit
    value: volts
  - default_value: 1.0
    doc: Scalar to multiply each element in data to convert it to the specified 'unit'.
      If the data are stored in acquisition system units or other units that require
      a conversion to be interpretable, multiply the data by 'conversion' to convert
      the data to the specified 'unit'. e.g. if the data acquisition system stores
      values in this object as signed 16-bit integers (int16 range -32,768 to 32,767)
      that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system
      gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition
      values to recorded volts is 2.5/32768/8000 = 9.5367e-9.
```

(continues on next page)

(continued from previous page)

```
    dtype: float32
    name: conversion
    required: false
  - default_value: 0.0
    doc: Scalar to add to the data after scaling by 'conversion' to finalize its coercion
      to the specified 'unit'. Two common examples of this include (a) data stored
      in an unsigned type that requires a shift after scaling to re-center the data,
      and (b) specialized recording devices that naturally cause a scalar offset with
      respect to the true units.
    dtype: float32
    name: offset
    required: false
  - default_value: -1.0
    doc: 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 -1.0.
    dtype: float32
    name: resolution
    required: false
  - doc: Optionally describe the continuity of the data. Can be "continuous", "instantaneous",
      or "step". For example, a voltage trace would be "continuous", because samples
      are recorded from a continuous process. An array of lick times would be "instantaneous",
      because the data represents distinct moments in time. Times of image presentations
      would be "step" because the picture remains the same until the next timepoint.
      This field is optional, but is useful in providing information about the underlying
      data. It may inform the way this data is interpreted, the way it is visualized,
      and what analysis methods are applicable.
    dtype: text
    name: continuity
    required: false
  dims:
  - - num_times
  - - num_times
    - num_channels
  - - num_times
    - num_channels
    - num_samples
  doc: Recorded voltage data.
  dtype: numeric
  name: data
  shape:
  - - 
  - - 
    - 
  - - 
    - 
    - 
- doc: DynamicTableRegion pointer to the electrodes that this time series was generated
    from.
  name: electrodes
```

(continues on next page)

(continued from previous page)

```yaml
    neurodata_type_inc: DynamicTableRegion
- attributes:
  - doc: The zero-indexed axis of the 'data' dataset that the channel-specific conversion
      factor corresponds to. This value is fixed to 1.
    dtype: int32
    name: axis
    value: 1
  dims:
  - num_channels
  doc: Channel-specific conversion factor. Multiply the data in the 'data' dataset
    by these values along the channel axis (as indicated by axis attribute) AND by
    the global conversion factor in the 'conversion' attribute of 'data' to get the
    data values in Volts, i.e, data in Volts = data * data.conversion * channel_conversion.
    This approach allows for both global and per-channel data conversion factors needed
    to support the storage of electrical recordings as native values generated by
    data acquisition systems. If this dataset is not present, then there is no channel-specific
    conversion factor, i.e. it is 1 for all channels.
  dtype: float32
  name: channel_conversion
  quantity: '?'
  shape:
  -
- attributes:
  - doc: Sampling rate, in Hz.
    dtype: float32
    name: rate
  - doc: Unit of measurement for time, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  doc: Timestamp of the first sample in seconds. When timestamps are uniformly spaced,
    the timestamp of the first sample can be specified and all subsequent ones calculated
    from the sampling rate attribute.
  dtype: float64
  name: starting_time
  quantity: '?'
- attributes:
  - doc: Value is '1'
    dtype: int32
    name: interval
    value: 1
  - doc: Unit of measurement for timestamps, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  dims:
  - num_times
  doc: Timestamps for samples stored in data, in seconds, relative to the common experiment
    master-clock stored in NWBFile.timestamps_reference_time.
```

(continues on next page)

(continued from previous page)

```yaml
  dtype: float64
  name: timestamps
  quantity: '?'
  shape:
  -
- dims:
  - num_times
  doc: Numerical labels that apply to each time point in data for the purpose of querying
    and slicing data by these values. If present, the length of this array should
    be the same size as the first dimension of data.
  dtype: uint8
  name: control
  quantity: '?'
  shape:
  -
- dims:
  - num_control_values
  doc: Description of each control value. Must be present if control is present. If
    present, control_description[0] should describe time points where control == 0.
  dtype: text
  name: control_description
  quantity: '?'
  shape:
  -
doc: A time series of acquired voltage data from extracellular recordings. The data
  field is an int or float array storing data in volts. The first dimension should
  always represent time. The second dimension, if present, should represent channels.
groups:
- doc: 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.
  name: sync
  quantity: '?'
neurodata_type_def: ElectricalSeries
neurodata_type_inc: TimeSeries
```

5.9.2 SpikeEventSeries

Extends: *ElectricalSeries*

Description: see Section 4.8.2

YAML Specification:

```yaml
attributes:
- doc: Filtering applied to all channels of the data. For example, if this ElectricalSeries
    represents high-pass-filtered data (also known as AP Band), then this value could
    be "High-pass 4-pole Bessel filter at 500 Hz". If this ElectricalSeries represents
    low-pass-filtered LFP data and the type of filter is unknown, then this value
    could be "Low-pass filter at 300 Hz". If a non-standard filter type is used, provide
    as much detail about the filter properties as possible.
  dtype: text
  name: filtering
  required: false
- default_value: no description
  doc: Description of the time series.
  dtype: text
  name: description
  required: false
- default_value: no comments
  doc: 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.
  dtype: text
  name: comments
  required: false
datasets:
- attributes:
  - doc: Unit of measurement for waveforms, which is fixed to 'volts'.
    dtype: text
    name: unit
    value: volts
  - default_value: 1.0
    doc: Scalar to multiply each element in data to convert it to the specified 'unit'.
      If the data are stored in acquisition system units or other units that require
      a conversion to be interpretable, multiply the data by 'conversion' to convert
      the data to the specified 'unit'. e.g. if the data acquisition system stores
      values in this object as signed 16-bit integers (int16 range -32,768 to 32,767)
      that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system
      gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition
      values to recorded volts is 2.5/32768/8000 = 9.5367e-9.
    dtype: float32
    name: conversion
    required: false
  - default_value: 0.0
    doc: Scalar to add to the data after scaling by 'conversion' to finalize its coercion
      to the specified 'unit'. Two common examples of this include (a) data stored
      in an unsigned type that requires a shift after scaling to re-center the data,
      and (b) specialized recording devices that naturally cause a scalar offset with
```

(continues on next page)

(continued from previous page)

```
      respect to the true units.
    dtype: float32
    name: offset
    required: false
  - default_value: -1.0
    doc: 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 -1.0.
    dtype: float32
    name: resolution
    required: false
  - doc: Optionally describe the continuity of the data. Can be "continuous",
↪"instantaneous",
      or "step". For example, a voltage trace would be "continuous", because samples
      are recorded from a continuous process. An array of lick times would be
↪"instantaneous",
      because the data represents distinct moments in time. Times of image presentations
      would be "step" because the picture remains the same until the next timepoint.
      This field is optional, but is useful in providing information about the underlying
      data. It may inform the way this data is interpreted, the way it is visualized,
      and what analysis methods are applicable.
    dtype: text
    name: continuity
    required: false
  dims:
  - - num_events
    - num_samples
  - - num_events
    - num_channels
    - num_samples
  doc: Spike waveforms.
  dtype: numeric
  name: data
  shape:
  - -
    -
  - -
    -
    -
- attributes:
  - doc: Value is '1'
    dtype: int32
    name: interval
    value: 1
  - doc: Unit of measurement for timestamps, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  dims:
  - num_times
  doc: Timestamps for samples stored in data, in seconds, relative to the common␣
↪experiment
```

(continues on next page)

(continued from previous page)

```
    master-clock stored in NWBFile.timestamps_reference_time. Timestamps are required
    for the events. Unlike for TimeSeries, timestamps are required for SpikeEventSeries
    and are thus re-specified here.
  dtype: float64
  name: timestamps
  shape:
  -
- doc: DynamicTableRegion pointer to the electrodes that this time series was generated
    from.
  name: electrodes
  neurodata_type_inc: DynamicTableRegion
- attributes:
  - doc: The zero-indexed axis of the 'data' dataset that the channel-specific conversion
      factor corresponds to. This value is fixed to 1.
    dtype: int32
    name: axis
    value: 1
  dims:
  - num_channels
  doc: Channel-specific conversion factor. Multiply the data in the 'data' dataset
    by these values along the channel axis (as indicated by axis attribute) AND by
    the global conversion factor in the 'conversion' attribute of 'data' to get the
    data values in Volts, i.e, data in Volts = data * data.conversion * channel_conversion.
    This approach allows for both global and per-channel data conversion factors needed
    to support the storage of electrical recordings as native values generated by
    data acquisition systems. If this dataset is not present, then there is no channel-specific
    conversion factor, i.e. it is 1 for all channels.
  dtype: float32
  name: channel_conversion
  quantity: '?'
  shape:
  -
- attributes:
  - doc: Sampling rate, in Hz.
    dtype: float32
    name: rate
  - doc: Unit of measurement for time, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  doc: Timestamp of the first sample in seconds. When timestamps are uniformly spaced,
    the timestamp of the first sample can be specified and all subsequent ones calculated
    from the sampling rate attribute.
  dtype: float64
  name: starting_time
  quantity: '?'
- dims:
  - num_times
  doc: Numerical labels that apply to each time point in data for the purpose of querying
    and slicing data by these values. If present, the length of this array should
```

(continues on next page)

(continued from previous page)

```yaml
    be the same size as the first dimension of data.
  dtype: uint8
  name: control
  quantity: '?'
  shape:
  -
- dims:
  - num_control_values
  doc: Description of each control value. Must be present if control is present. If
    present, control_description[0] should describe time points where control == 0.
  dtype: text
  name: control_description
  quantity: '?'
  shape:
  -
doc: 'Stores snapshots/snippets of recorded spike events (i.e., threshold crossings).
  This may also be raw data, as reported by ephys hardware. If so, the TimeSeries::description
  field should describe 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).'
groups:
- doc: 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.
  name: sync
  quantity: '?'
neurodata_type_def: SpikeEventSeries
neurodata_type_inc: ElectricalSeries
```

5.9.3 FeatureExtraction

Extends: *NWBDataInterface*

Description: see Section 4.8.3

YAML Specification:


```yaml
datasets:
- dims:
  - num_features
  doc: Description of features (eg, ''PC1'') for each of the extracted features.
  dtype: text
  name: description
  shape:
  -
- dims:
  - num_events
  - num_channels
  - num_features
  doc: Multi-dimensional array of features extracted from each event.
  dtype: float32
  name: features
  shape:
  -
  -
  -
- dims:
  - num_events
  doc: Times of events that features correspond to (can be a link).
  dtype: float64
  name: times
  shape:
  -
- doc: DynamicTableRegion pointer to the electrodes that this time series was generated
    from.
  name: electrodes
  neurodata_type_inc: DynamicTableRegion
default_name: FeatureExtraction
doc: Features, such as PC1 and PC2, that are extracted from signals stored in a SpikeEventSeries
  or other source.
neurodata_type_def: FeatureExtraction
neurodata_type_inc: NWBDataInterface
```

5.9.4 EventDetection

Extends: *NWBDataInterface*

Description: see Section 4.8.4

YAML Specification:

```yaml
datasets:
- doc: Description of how events were detected, such as voltage threshold, or dV/dT
    threshold, as well as relevant values.
  dtype: text
  name: detection_method
- dims:
  - num_events
  doc: Indices (zero-based) into source ElectricalSeries::data array corresponding
    to time of event. ''description'' should define what is meant by time of event
    (e.g., .25 ms before action potential peak, zero-crossing time, etc). The index
    points to each event from the raw data.
  dtype: int32
  name: source_idx
  shape:
  -
- attributes:
  - doc: Unit of measurement for event times, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  dims:
  - num_events
  doc: Timestamps of events, in seconds.
  dtype: float64
  name: times
  shape:
  -
default_name: EventDetection
doc: Detected spike events from voltage trace(s).
links:
- doc: Link to the 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 include that information here.
  name: source_electricalseries
  target_type: ElectricalSeries
neurodata_type_def: EventDetection
neurodata_type_inc: NWBDataInterface
```

5.9.5 EventWaveform

Extends: *NWBDataInterface*

Description: see Section 4.8.5

YAML Specification:


```yaml
default_name: EventWaveform
doc: 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.
groups:
- doc: SpikeEventSeries object(s) containing detected spike event waveforms.
  neurodata_type_inc: SpikeEventSeries
  quantity: '*'
neurodata_type_def: EventWaveform
neurodata_type_inc: NWBDataInterface
```

5.9.6 FilteredEphys

Extends: *NWBDataInterface*

Description: see Section 4.8.6

YAML Specification:

```yaml
default_name: FilteredEphys
doc: Electrophysiology 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. Filter properties
  should be noted in the ElectricalSeries 'filtering' attribute.
groups:
- doc: ElectricalSeries object(s) containing filtered electrophysiology data.
  neurodata_type_inc: ElectricalSeries
  quantity: +
neurodata_type_def: FilteredEphys
neurodata_type_inc: NWBDataInterface
```

5.9.7 LFP

Extends: *NWBDataInterface*

Description: see Section 4.8.7

YAML Specification:

```yaml
default_name: LFP
doc: 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 'filtering' attribute.
groups:
- doc: ElectricalSeries object(s) containing LFP data for one or more channels.
  neurodata_type_inc: ElectricalSeries
  quantity: +
neurodata_type_def: LFP
neurodata_type_inc: NWBDataInterface
```

5.9.8 ElectrodeGroup

Extends: *NWBContainer*

Description: see Section 4.8.8

YAML Specification:

```yaml
attributes:
- doc: Description of this electrode group.
  dtype: text
  name: description
- doc: Location of electrode group. Specify the area, layer, comments on estimation
    of area/layer, etc. Use standard atlas names for anatomical regions when possible.
  dtype: text
  name: location
datasets:
- doc: stereotaxic or common framework coordinates
  dtype:
  - doc: x coordinate
    dtype: float32
    name: x
  - doc: y coordinate
    dtype: float32
    name: y
  - doc: z coordinate
    dtype: float32
    name: z
  name: position
  quantity: '?'
doc: A physical grouping of electrodes, e.g. a shank of an array.
links:
- doc: Link to the device that was used to record from this electrode group.
  name: device
  target_type: Device
neurodata_type_def: ElectrodeGroup
neurodata_type_inc: NWBContainer
```

5.9.9 ClusterWaveforms

Extends: *NWBDataInterface*

Description: see Section 4.8.9

YAML Specification:

```yaml
datasets:
- doc: Filtering applied to data before generating mean/sd
  dtype: text
  name: waveform_filtering
- dims:
  - num_clusters
  - num_samples
  doc: 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)
  dtype: float32
  name: waveform_mean
  shape:
  -
  -
- dims:
  - num_clusters
  - num_samples
  doc: Stdev of waveforms for each cluster, using the same indices as in mean
  dtype: float32
  name: waveform_sd
  shape:
  -
  -
default_name: ClusterWaveforms
doc: DEPRECATED 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.
links:
- doc: Link to Clustering interface that was the source of the clustered data
  name: clustering_interface
  target_type: Clustering
neurodata_type_def: ClusterWaveforms
neurodata_type_inc: NWBDataInterface
```

5.9.10 Clustering

Extends: *NWBDataInterface*

Description: see Section 4.8.10

YAML Specification:


```yaml
datasets:
- doc: Description of clusters or clustering, (e.g. cluster 0 is noise, clusters curated
    using Klusters, etc)
  dtype: text
  name: description
- dims:
  - num_events
  doc: Cluster number of each event
  dtype: int32
  name: num
  shape:
  -
- dims:
  - num_clusters
  doc: Maximum ratio of waveform peak to RMS on any channel in the cluster (provides
    a basic clustering metric).
  dtype: float32
  name: peak_over_rms
  shape:
  -
- dims:
  - num_events
  doc: Times of clustered events, in seconds. This may be a link to times field in
    associated FeatureExtraction module.
  dtype: float64
  name: times
  shape:
  -
default_name: Clustering
doc: DEPRECATED Clustered spike data, whether from automatic clustering tools (e.g.,
  klustakwik) or as a result of manual sorting.
neurodata_type_def: Clustering
neurodata_type_inc: NWBDataInterface
```

5.10 Intracellular electrophysiology

This source module contains neurodata_types for intracellular electrophysiology data.

5.10.1 PatchClampSeries

Extends: *TimeSeries*

Description: see Section 4.9.1

YAML Specification:

```yaml
attributes:
- doc: Protocol/stimulus name for this patch-clamp dataset.
  dtype: text
  name: stimulus_description
- doc: Sweep number, allows to group different PatchClampSeries together.
  dtype: uint32
  name: sweep_number
  required: false
- default_value: no description
  doc: Description of the time series.
  dtype: text
  name: description
  required: false
- default_value: no comments
  doc: 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.
  dtype: text
  name: comments
  required: false
datasets:
- attributes:
  - doc: Base unit of measurement for working with the data. Actual stored values
      are not necessarily stored in these units. To access the data in these units,
      multiply 'data' by 'conversion' and add 'offset'.
    dtype: text
    name: unit
  - default_value: 1.0
    doc: Scalar to multiply each element in data to convert it to the specified 'unit'.
      If the data are stored in acquisition system units or other units that require
      a conversion to be interpretable, multiply the data by 'conversion' to convert
      the data to the specified 'unit'. e.g. if the data acquisition system stores
      values in this object as signed 16-bit integers (int16 range -32,768 to 32,767)
      that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system
      gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition
      values to recorded volts is 2.5/32768/8000 = 9.5367e-9.
    dtype: float32
    name: conversion
    required: false
  - default_value: 0.0
    doc: Scalar to add to the data after scaling by 'conversion' to finalize its coercion
```

(continues on next page)

(continued from previous page)

```
      to the specified 'unit'. Two common examples of this include (a) data stored
      in an unsigned type that requires a shift after scaling to re-center the data,
      and (b) specialized recording devices that naturally cause a scalar offset with
      respect to the true units.
    dtype: float32
    name: offset
    required: false
  - default_value: -1.0
    doc: 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 -1.0.
    dtype: float32
    name: resolution
    required: false
  - doc: Optionally describe the continuity of the data. Can be "continuous", "instantaneous",
      or "step". For example, a voltage trace would be "continuous", because samples
      are recorded from a continuous process. An array of lick times would be "instantaneous",
      because the data represents distinct moments in time. Times of image presentations
      would be "step" because the picture remains the same until the next timepoint.
      This field is optional, but is useful in providing information about the underlying
      data. It may inform the way this data is interpreted, the way it is visualized,
      and what analysis methods are applicable.
    dtype: text
    name: continuity
    required: false
  dims:
  - num_times
  doc: Recorded voltage or current.
  dtype: numeric
  name: data
  shape:
  -
- doc: Gain of the recording, in units Volt/Amp (v-clamp) or Volt/Volt (c-clamp).
  dtype: float32
  name: gain
  quantity: '?'
- attributes:
  - doc: Sampling rate, in Hz.
    dtype: float32
    name: rate
  - doc: Unit of measurement for time, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  doc: Timestamp of the first sample in seconds. When timestamps are uniformly spaced,
    the timestamp of the first sample can be specified and all subsequent ones calculated
    from the sampling rate attribute.
  dtype: float64
  name: starting_time
  quantity: '?'
```

(continues on next page)

(continued from previous page)

```
- attributes:
  - doc: Value is '1'
    dtype: int32
    name: interval
    value: 1
  - doc: Unit of measurement for timestamps, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  dims:
  - num_times
  doc: Timestamps for samples stored in data, in seconds, relative to the common experiment
    master-clock stored in NWBFile.timestamps_reference_time.
  dtype: float64
  name: timestamps
  quantity: '?'
  shape:
  -
- dims:
  - num_times
  doc: Numerical labels that apply to each time point in data for the purpose of querying
    and slicing data by these values. If present, the length of this array should
    be the same size as the first dimension of data.
  dtype: uint8
  name: control
  quantity: '?'
  shape:
  -
- dims:
  - num_control_values
  doc: Description of each control value. Must be present if control is present. If
    present, control_description[0] should describe time points where control == 0.
  dtype: text
  name: control_description
  quantity: '?'
  shape:
  -
doc: An abstract base class for patch-clamp data - stimulus or response, current or
  voltage.
groups:
- doc: 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.
  name: sync
  quantity: '?'
links:
- doc: Link to IntracellularElectrode object that describes the electrode that was
```

(continues on next page)

(continued from previous page)

```
    used to apply or record this data.
  name: electrode
  target_type: IntracellularElectrode
neurodata_type_def: PatchClampSeries
neurodata_type_inc: TimeSeries
```

5.10.2 CurrentClampSeries

Extends: *PatchClampSeries*

Description: see Section 4.9.2

YAML Specification:

```yaml
attributes:
- doc: Protocol/stimulus name for this patch-clamp dataset.
  dtype: text
  name: stimulus_description
- doc: Sweep number, allows to group different PatchClampSeries together.
  dtype: uint32
  name: sweep_number
  required: false
- default_value: no description
  doc: Description of the time series.
  dtype: text
  name: description
  required: false
- default_value: no comments
  doc: 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.
  dtype: text
  name: comments
  required: false
datasets:
- attributes:
  - doc: Base unit of measurement for working with the data. which is fixed to 'volts'.
      Actual stored values are not necessarily stored in these units. To access the
      data in these units, multiply 'data' by 'conversion' and add 'offset'.
    dtype: text
    name: unit
    value: volts
  - default_value: 1.0
    doc: Scalar to multiply each element in data to convert it to the specified 'unit'.
      If the data are stored in acquisition system units or other units that require
      a conversion to be interpretable, multiply the data by 'conversion' to convert
      the data to the specified 'unit'. e.g. if the data acquisition system stores
      values in this object as signed 16-bit integers (int16 range -32,768 to 32,767)
      that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system
      gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition
      values to recorded volts is 2.5/32768/8000 = 9.5367e-9.
    dtype: float32
    name: conversion
    required: false
  - default_value: 0.0
    doc: Scalar to add to the data after scaling by 'conversion' to finalize its coercion
      to the specified 'unit'. Two common examples of this include (a) data stored
      in an unsigned type that requires a shift after scaling to re-center the data,
      and (b) specialized recording devices that naturally cause a scalar offset with
      respect to the true units.
```

(continues on next page)

(continued from previous page)

```
    dtype: float32
    name: offset
    required: false
  - default_value: -1.0
    doc: 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 -1.0.
    dtype: float32
    name: resolution
    required: false
  - doc: Optionally describe the continuity of the data. Can be "continuous", "instantaneous",
      or "step". For example, a voltage trace would be "continuous", because samples
      are recorded from a continuous process. An array of lick times would be "instantaneous",
      because the data represents distinct moments in time. Times of image presentations
      would be "step" because the picture remains the same until the next timepoint.
      This field is optional, but is useful in providing information about the underlying
      data. It may inform the way this data is interpreted, the way it is visualized,
      and what analysis methods are applicable.
    dtype: text
    name: continuity
    required: false
  doc: Recorded voltage.
  name: data
- doc: Bias current, in amps.
  dtype: float32
  name: bias_current
  quantity: '?'
- doc: Bridge balance, in ohms.
  dtype: float32
  name: bridge_balance
  quantity: '?'
- doc: Capacitance compensation, in farads.
  dtype: float32
  name: capacitance_compensation
  quantity: '?'
- doc: Gain of the recording, in units Volt/Amp (v-clamp) or Volt/Volt (c-clamp).
  dtype: float32
  name: gain
  quantity: '?'
- attributes:
  - doc: Sampling rate, in Hz.
    dtype: float32
    name: rate
  - doc: Unit of measurement for time, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  doc: Timestamp of the first sample in seconds. When timestamps are uniformly spaced,
    the timestamp of the first sample can be specified and all subsequent ones calculated
    from the sampling rate attribute.
```

(continues on next page)

(continued from previous page)

```
  dtype: float64
  name: starting_time
  quantity: '?'
- attributes:
  - doc: Value is '1'
    dtype: int32
    name: interval
    value: 1
  - doc: Unit of measurement for timestamps, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  dims:
  - num_times
  doc: Timestamps for samples stored in data, in seconds, relative to the common experiment
    master-clock stored in NWBFile.timestamps_reference_time.
  dtype: float64
  name: timestamps
  quantity: '?'
  shape:
  -
- dims:
  - num_times
  doc: Numerical labels that apply to each time point in data for the purpose of querying
    and slicing data by these values. If present, the length of this array should
    be the same size as the first dimension of data.
  dtype: uint8
  name: control
  quantity: '?'
  shape:
  -
- dims:
  - num_control_values
  doc: Description of each control value. Must be present if control is present. If
    present, control_description[0] should describe time points where control == 0.
  dtype: text
  name: control_description
  quantity: '?'
  shape:
  -
doc: Voltage data from an intracellular current-clamp recording. A corresponding CurrentClampStimulusSeries
  (stored separately as a stimulus) is used to store the current injected.
groups:
- doc: 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.
```

(continues on next page)

(continued from previous page)

```
  name: sync
  quantity: '?'
links:
- doc: Link to IntracellularElectrode object that describes the electrode that was
    used to apply or record this data.
  name: electrode
  target_type: IntracellularElectrode
neurodata_type_def: CurrentClampSeries
neurodata_type_inc: PatchClampSeries
```

5.10.3 IZeroClampSeries

Extends: *CurrentClampSeries*

Description: see Section 4.9.3

YAML Specification:

```yaml
attributes:
- doc: An IZeroClampSeries has no stimulus, so this attribute is automatically set
    to "N/A"
  dtype: text
  name: stimulus_description
  value: N/A
- doc: Sweep number, allows to group different PatchClampSeries together.
  dtype: uint32
  name: sweep_number
  required: false
- default_value: no description
  doc: Description of the time series.
  dtype: text
  name: description
  required: false
- default_value: no comments
  doc: 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.
  dtype: text
  name: comments
  required: false
datasets:
- doc: Bias current, in amps, fixed to 0.0.
  dtype: float32
  name: bias_current
- doc: Bridge balance, in ohms, fixed to 0.0.
  dtype: float32
  name: bridge_balance
- doc: Capacitance compensation, in farads, fixed to 0.0.
  dtype: float32
  name: capacitance_compensation
- attributes:
  - doc: Base unit of measurement for working with the data. which is fixed to 'volts'.
      Actual stored values are not necessarily stored in these units. To access the
      data in these units, multiply 'data' by 'conversion' and add 'offset'.
    dtype: text
    name: unit
    value: volts
  - default_value: 1.0
    doc: Scalar to multiply each element in data to convert it to the specified 'unit'.
      If the data are stored in acquisition system units or other units that require
      a conversion to be interpretable, multiply the data by 'conversion' to convert
      the data to the specified 'unit'. e.g. if the data acquisition system stores
      values in this object as signed 16-bit integers (int16 range -32,768 to 32,767)
      that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system
```

(continues on next page)

(continued from previous page)

```
      gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition
      values to recorded volts is 2.5/32768/8000 = 9.5367e-9.
    dtype: float32
    name: conversion
    required: false
  - default_value: 0.0
    doc: Scalar to add to the data after scaling by 'conversion' to finalize its coercion
      to the specified 'unit'. Two common examples of this include (a) data stored
      in an unsigned type that requires a shift after scaling to re-center the data,
      and (b) specialized recording devices that naturally cause a scalar offset with
      respect to the true units.
    dtype: float32
    name: offset
    required: false
  - default_value: -1.0
    doc: 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 -1.0.
    dtype: float32
    name: resolution
    required: false
  - doc: Optionally describe the continuity of the data. Can be "continuous", "instantaneous",
      or "step". For example, a voltage trace would be "continuous", because samples
      are recorded from a continuous process. An array of lick times would be "instantaneous",
      because the data represents distinct moments in time. Times of image presentations
      would be "step" because the picture remains the same until the next timepoint.
      This field is optional, but is useful in providing information about the underlying
      data. It may inform the way this data is interpreted, the way it is visualized,
      and what analysis methods are applicable.
    dtype: text
    name: continuity
    required: false
  doc: Recorded voltage.
  name: data
- doc: Gain of the recording, in units Volt/Amp (v-clamp) or Volt/Volt (c-clamp).
  dtype: float32
  name: gain
  quantity: '?'
- attributes:
  - doc: Sampling rate, in Hz.
    dtype: float32
    name: rate
  - doc: Unit of measurement for time, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  doc: Timestamp of the first sample in seconds. When timestamps are uniformly spaced,
    the timestamp of the first sample can be specified and all subsequent ones calculated
    from the sampling rate attribute.
  dtype: float64
```

(continues on next page)

(continued from previous page)

```yaml
    name: starting_time
    quantity: '?'
  - attributes:
    - doc: Value is '1'
      dtype: int32
      name: interval
      value: 1
    - doc: Unit of measurement for timestamps, which is fixed to 'seconds'.
      dtype: text
      name: unit
      value: seconds
    dims:
    - num_times
    doc: Timestamps for samples stored in data, in seconds, relative to the common experiment
      master-clock stored in NWBFile.timestamps_reference_time.
    dtype: float64
    name: timestamps
    quantity: '?'
    shape:
    -
  - dims:
    - num_times
    doc: Numerical labels that apply to each time point in data for the purpose of querying
      and slicing data by these values. If present, the length of this array should
      be the same size as the first dimension of data.
    dtype: uint8
    name: control
    quantity: '?'
    shape:
    -
  - dims:
    - num_control_values
    doc: Description of each control value. Must be present if control is present. If
      present, control_description[0] should describe time points where control == 0.
    dtype: text
    name: control_description
    quantity: '?'
    shape:
    -
  doc: Voltage data from an intracellular recording 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.
  groups:
  - doc: 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
```

(continues on next page)

(continued from previous page)

```yaml
    mostly for archival purposes.
  name: sync
  quantity: '?'
links:
- doc: Link to IntracellularElectrode object that describes the electrode that was
    used to apply or record this data.
  name: electrode
  target_type: IntracellularElectrode
neurodata_type_def: IZeroClampSeries
neurodata_type_inc: CurrentClampSeries
```

5.10.4 CurrentClampStimulusSeries

Extends: *PatchClampSeries*

Description: see Section 4.9.4

YAML Specification:


```yaml
attributes:
- doc: Protocol/stimulus name for this patch-clamp dataset.
  dtype: text
  name: stimulus_description
- doc: Sweep number, allows to group different PatchClampSeries together.
  dtype: uint32
  name: sweep_number
  required: false
- default_value: no description
  doc: Description of the time series.
  dtype: text
  name: description
  required: false
- default_value: no comments
  doc: 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.
  dtype: text
  name: comments
  required: false
datasets:
- attributes:
  - doc: Base unit of measurement for working with the data. which is fixed to 'amperes'.
      Actual stored values are not necessarily stored in these units. To access the
      data in these units, multiply 'data' by 'conversion' and add 'offset'.
    dtype: text
    name: unit
    value: amperes
  - default_value: 1.0
    doc: Scalar to multiply each element in data to convert it to the specified 'unit'.
      If the data are stored in acquisition system units or other units that require
      a conversion to be interpretable, multiply the data by 'conversion' to convert
      the data to the specified 'unit'. e.g. if the data acquisition system stores
      values in this object as signed 16-bit integers (int16 range -32,768 to 32,767)
      that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system
      gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition
      values to recorded volts is 2.5/32768/8000 = 9.5367e-9.
    dtype: float32
    name: conversion
    required: false
  - default_value: 0.0
    doc: Scalar to add to the data after scaling by 'conversion' to finalize its coercion
      to the specified 'unit'. Two common examples of this include (a) data stored
      in an unsigned type that requires a shift after scaling to re-center the data,
      and (b) specialized recording devices that naturally cause a scalar offset with
      respect to the true units.
```


(continues on next page)

(continued from previous page)

```
    dtype: float32
    name: offset
    required: false
  - default_value: -1.0
    doc: 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 -1.0.
    dtype: float32
    name: resolution
    required: false
  - doc: Optionally describe the continuity of the data. Can be "continuous", "instantaneous",
      or "step". For example, a voltage trace would be "continuous", because samples
      are recorded from a continuous process. An array of lick times would be "instantaneous",
      because the data represents distinct moments in time. Times of image presentations
      would be "step" because the picture remains the same until the next timepoint.
      This field is optional, but is useful in providing information about the underlying
      data. It may inform the way this data is interpreted, the way it is visualized,
      and what analysis methods are applicable.
    dtype: text
    name: continuity
    required: false
  doc: Stimulus current applied.
  name: data
- doc: Gain of the recording, in units Volt/Amp (v-clamp) or Volt/Volt (c-clamp).
  dtype: float32
  name: gain
  quantity: '?'
- attributes:
  - doc: Sampling rate, in Hz.
    dtype: float32
    name: rate
  - doc: Unit of measurement for time, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  doc: Timestamp of the first sample in seconds. When timestamps are uniformly spaced,
    the timestamp of the first sample can be specified and all subsequent ones calculated
    from the sampling rate attribute.
  dtype: float64
  name: starting_time
  quantity: '?'
- attributes:
  - doc: Value is '1'
    dtype: int32
    name: interval
    value: 1
  - doc: Unit of measurement for timestamps, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
```

(continues on next page)

(continued from previous page)

```yaml
  dims:
  - num_times
  doc: Timestamps for samples stored in data, in seconds, relative to the common experiment
    master-clock stored in NWBFile.timestamps_reference_time.
  dtype: float64
  name: timestamps
  quantity: '?'
  shape:
  -
- dims:
  - num_times
  doc: Numerical labels that apply to each time point in data for the purpose of querying
    and slicing data by these values. If present, the length of this array should
    be the same size as the first dimension of data.
  dtype: uint8
  name: control
  quantity: '?'
  shape:
  -
- dims:
  - num_control_values
  doc: Description of each control value. Must be present if control is present. If
    present, control_description[0] should describe time points where control == 0.
  dtype: text
  name: control_description
  quantity: '?'
  shape:
  -
doc: Stimulus current applied during current clamp recording.
groups:
- doc: 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.
  name: sync
  quantity: '?'
links:
- doc: Link to IntracellularElectrode object that describes the electrode that was
    used to apply or record this data.
  name: electrode
  target_type: IntracellularElectrode
neurodata_type_def: CurrentClampStimulusSeries
neurodata_type_inc: PatchClampSeries
```

5.10.5 VoltageClampSeries

Extends: *PatchClampSeries*

Description: see Section 4.9.5

YAML Specification:


```yaml
attributes:
- doc: Protocol/stimulus name for this patch-clamp dataset.
  dtype: text
  name: stimulus_description
- doc: Sweep number, allows to group different PatchClampSeries together.
  dtype: uint32
  name: sweep_number
  required: false
- default_value: no description
  doc: Description of the time series.
  dtype: text
  name: description
  required: false
- default_value: no comments
  doc: 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.
  dtype: text
  name: comments
  required: false
datasets:
- attributes:
  - doc: Base unit of measurement for working with the data. which is fixed to 'amperes'.
      Actual stored values are not necessarily stored in these units. To access the
      data in these units, multiply 'data' by 'conversion' and add 'offset'.
    dtype: text
    name: unit
    value: amperes
  - default_value: 1.0
    doc: Scalar to multiply each element in data to convert it to the specified 'unit'.
      If the data are stored in acquisition system units or other units that require
      a conversion to be interpretable, multiply the data by 'conversion' to convert
      the data to the specified 'unit'. e.g. if the data acquisition system stores
      values in this object as signed 16-bit integers (int16 range -32,768 to 32,767)
      that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system
      gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition
      values to recorded volts is 2.5/32768/8000 = 9.5367e-9.
    dtype: float32
    name: conversion
    required: false
  - default_value: 0.0
    doc: Scalar to add to the data after scaling by 'conversion' to finalize its coercion
      to the specified 'unit'. Two common examples of this include (a) data stored
      in an unsigned type that requires a shift after scaling to re-center the data,
      and (b) specialized recording devices that naturally cause a scalar offset with
      respect to the true units.
```


(continues on next page)

(continued from previous page)

```yaml
    dtype: float32
    name: offset
    required: false
  - default_value: -1.0
    doc: 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 -1.0.
    dtype: float32
    name: resolution
    required: false
  - doc: Optionally describe the continuity of the data. Can be "continuous",
↪"instantaneous",
      or "step". For example, a voltage trace would be "continuous", because samples
      are recorded from a continuous process. An array of lick times would be
↪"instantaneous",
      because the data represents distinct moments in time. Times of image presentations
      would be "step" because the picture remains the same until the next timepoint.
      This field is optional, but is useful in providing information about the underlying
      data. It may inform the way this data is interpreted, the way it is visualized,
      and what analysis methods are applicable.
    dtype: text
    name: continuity
    required: false
  doc: Recorded current.
  name: data
- attributes:
  - doc: Unit of measurement for capacitance_fast, which is fixed to 'farads'.
    dtype: text
    name: unit
    value: farads
  doc: Fast capacitance, in farads.
  dtype: float32
  name: capacitance_fast
  quantity: '?'
- attributes:
  - doc: Unit of measurement for capacitance_fast, which is fixed to 'farads'.
    dtype: text
    name: unit
    value: farads
  doc: Slow capacitance, in farads.
  dtype: float32
  name: capacitance_slow
  quantity: '?'
- attributes:
  - doc: Unit of measurement for resistance_comp_bandwidth, which is fixed to 'hertz'.
    dtype: text
    name: unit
    value: hertz
  doc: Resistance compensation bandwidth, in hertz.
  dtype: float32
  name: resistance_comp_bandwidth
  quantity: '?'
```

(continues on next page)

(continued from previous page)

```yaml
- attributes:
  - doc: Unit of measurement for resistance_comp_correction, which is fixed to 'percent'.
    dtype: text
    name: unit
    value: percent
  doc: Resistance compensation correction, in percent.
  dtype: float32
  name: resistance_comp_correction
  quantity: '?'
- attributes:
  - doc: Unit of measurement for resistance_comp_prediction, which is fixed to 'percent'.
    dtype: text
    name: unit
    value: percent
  doc: Resistance compensation prediction, in percent.
  dtype: float32
  name: resistance_comp_prediction
  quantity: '?'
- attributes:
  - doc: Unit of measurement for whole_cell_capacitance_comp, which is fixed to 'farads'.
    dtype: text
    name: unit
    value: farads
  doc: Whole cell capacitance compensation, in farads.
  dtype: float32
  name: whole_cell_capacitance_comp
  quantity: '?'
- attributes:
  - doc: Unit of measurement for whole_cell_series_resistance_comp, which is fixed
      to 'ohms'.
    dtype: text
    name: unit
    value: ohms
  doc: Whole cell series resistance compensation, in ohms.
  dtype: float32
  name: whole_cell_series_resistance_comp
  quantity: '?'
- doc: Gain of the recording, in units Volt/Amp (v-clamp) or Volt/Volt (c-clamp).
  dtype: float32
  name: gain
  quantity: '?'
- attributes:
  - doc: Sampling rate, in Hz.
    dtype: float32
    name: rate
  - doc: Unit of measurement for time, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  doc: Timestamp of the first sample in seconds. When timestamps are uniformly spaced,
    the timestamp of the first sample can be specified and all subsequent ones calculated
    from the sampling rate attribute.
```

(continues on next page)

(continued from previous page)

```
  dtype: float64
  name: starting_time
  quantity: '?'
- attributes:
  - doc: Value is '1'
    dtype: int32
    name: interval
    value: 1
  - doc: Unit of measurement for timestamps, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  dims:
  - num_times
  doc: Timestamps for samples stored in data, in seconds, relative to the common experiment
    master-clock stored in NWBFile.timestamps_reference_time.
  dtype: float64
  name: timestamps
  quantity: '?'
  shape:
  -
- dims:
  - num_times
  doc: Numerical labels that apply to each time point in data for the purpose of querying
    and slicing data by these values. If present, the length of this array should
    be the same size as the first dimension of data.
  dtype: uint8
  name: control
  quantity: '?'
  shape:
  -
- dims:
  - num_control_values
  doc: Description of each control value. Must be present if control is present. If
    present, control_description[0] should describe time points where control == 0.
  dtype: text
  name: control_description
  quantity: '?'
  shape:
  -
doc: Current data from an intracellular voltage-clamp recording. A corresponding VoltageClampStimulusSeries
  (stored separately as a stimulus) is used to store the voltage injected.
groups:
- doc: 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.
```

(continues on next page)

(continued from previous page)

```
  name: sync
  quantity: '?'
links:
- doc: Link to IntracellularElectrode object that describes the electrode that was
    used to apply or record this data.
  name: electrode
  target_type: IntracellularElectrode
neurodata_type_def: VoltageClampSeries
neurodata_type_inc: PatchClampSeries
```

5.10.6 VoltageClampStimulusSeries

Extends: *PatchClampSeries*

Description: see Section 4.9.6

YAML Specification:

```yaml
attributes:
- doc: Protocol/stimulus name for this patch-clamp dataset.
  dtype: text
  name: stimulus_description
- doc: Sweep number, allows to group different PatchClampSeries together.
  dtype: uint32
  name: sweep_number
  required: false
- default_value: no description
  doc: Description of the time series.
  dtype: text
  name: description
  required: false
- default_value: no comments
  doc: 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.
  dtype: text
  name: comments
  required: false
datasets:
- attributes:
  - doc: Base unit of measurement for working with the data. which is fixed to 'volts'.
      Actual stored values are not necessarily stored in these units. To access the
      data in these units, multiply 'data' by 'conversion' and add 'offset'.
    dtype: text
    name: unit
    value: volts
  - default_value: 1.0
    doc: Scalar to multiply each element in data to convert it to the specified 'unit'.
      If the data are stored in acquisition system units or other units that require
      a conversion to be interpretable, multiply the data by 'conversion' to convert
      the data to the specified 'unit'. e.g. if the data acquisition system stores
      values in this object as signed 16-bit integers (int16 range -32,768 to 32,767)
      that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system
      gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition
      values to recorded volts is 2.5/32768/8000 = 9.5367e-9.
    dtype: float32
    name: conversion
    required: false
  - default_value: 0.0
    doc: Scalar to add to the data after scaling by 'conversion' to finalize its coercion
      to the specified 'unit'. Two common examples of this include (a) data stored
      in an unsigned type that requires a shift after scaling to re-center the data,
      and (b) specialized recording devices that naturally cause a scalar offset with
      respect to the true units.
```

(continues on next page)

(continued from previous page)

```yaml
    dtype: float32
    name: offset
    required: false
  - default_value: -1.0
    doc: 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 -1.0.
    dtype: float32
    name: resolution
    required: false
  - doc: Optionally describe the continuity of the data. Can be "continuous", "instantaneous",
      or "step". For example, a voltage trace would be "continuous", because samples
      are recorded from a continuous process. An array of lick times would be "instantaneous",
      because the data represents distinct moments in time. Times of image presentations
      would be "step" because the picture remains the same until the next timepoint.
      This field is optional, but is useful in providing information about the underlying
      data. It may inform the way this data is interpreted, the way it is visualized,
      and what analysis methods are applicable.
    dtype: text
    name: continuity
    required: false
  doc: Stimulus voltage applied.
  name: data
- doc: Gain of the recording, in units Volt/Amp (v-clamp) or Volt/Volt (c-clamp).
  dtype: float32
  name: gain
  quantity: '?'
- attributes:
  - doc: Sampling rate, in Hz.
    dtype: float32
    name: rate
  - doc: Unit of measurement for time, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  doc: Timestamp of the first sample in seconds. When timestamps are uniformly spaced,
    the timestamp of the first sample can be specified and all subsequent ones calculated
    from the sampling rate attribute.
  dtype: float64
  name: starting_time
  quantity: '?'
- attributes:
  - doc: Value is '1'
    dtype: int32
    name: interval
    value: 1
  - doc: Unit of measurement for timestamps, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
```

(continues on next page)

(continued from previous page)

```yaml
  dims:
  - num_times
  doc: Timestamps for samples stored in data, in seconds, relative to the common experiment
    master-clock stored in NWBFile.timestamps_reference_time.
  dtype: float64
  name: timestamps
  quantity: '?'
  shape:
  -
- dims:
  - num_times
  doc: Numerical labels that apply to each time point in data for the purpose of querying
    and slicing data by these values. If present, the length of this array should
    be the same size as the first dimension of data.
  dtype: uint8
  name: control
  quantity: '?'
  shape:
  -
- dims:
  - num_control_values
  doc: Description of each control value. Must be present if control is present. If
    present, control_description[0] should describe time points where control == 0.
  dtype: text
  name: control_description
  quantity: '?'
  shape:
  -
doc: Stimulus voltage applied during a voltage clamp recording.
groups:
- doc: 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.
  name: sync
  quantity: '?'
links:
- doc: Link to IntracellularElectrode object that describes the electrode that was
    used to apply or record this data.
  name: electrode
  target_type: IntracellularElectrode
neurodata_type_def: VoltageClampStimulusSeries
neurodata_type_inc: PatchClampSeries
```

5.10.7 IntracellularElectrode

Extends: *NWBContainer*

Description: see Section 4.9.7

YAML Specification:

```yaml
datasets:
- doc: unique ID of the cell
  dtype: text
  name: cell_id
  quantity: '?'
- doc: Description of electrode (e.g.,  whole-cell, sharp, etc.).
  dtype: text
  name: description
- doc: Electrode specific filtering.
  dtype: text
  name: filtering
  quantity: '?'
- doc: Initial access resistance.
  dtype: text
  name: initial_access_resistance
  quantity: '?'
- doc: Location of the electrode. Specify the area, layer, comments on estimation
    of area/layer, stereotaxic coordinates if in vivo, etc. Use standard atlas names
    for anatomical regions when possible.
  dtype: text
  name: location
  quantity: '?'
- doc: Electrode resistance, in ohms.
  dtype: text
  name: resistance
  quantity: '?'
- doc: Information about seal used for recording.
  dtype: text
  name: seal
  quantity: '?'
- doc: Information about slice used for recording.
  dtype: text
  name: slice
  quantity: '?'
doc: An intracellular electrode and its metadata.
links:
- doc: Device that was used to record from this electrode.
  name: device
  target_type: Device
neurodata_type_def: IntracellularElectrode
neurodata_type_inc: NWBContainer
```

5.10.8 SweepTable

Extends: DynamicTable

Description: see Section 4.9.8

YAML Specification:

```yaml
attributes:
- dims:
  - num_columns
  doc: The names of the columns in this table. This should be used to specify an order
    to the columns.
  dtype: text
  name: colnames
  shape:
  -
- doc: Description of what is in this dynamic table.
  dtype: text
  name: description
datasets:
- doc: Sweep number of the PatchClampSeries in that row.
  dtype: uint32
  name: sweep_number
  neurodata_type_inc: VectorData
- doc: The PatchClampSeries with the sweep number in that row.
  dtype:
    reftype: object
    target_type: PatchClampSeries
  name: series
  neurodata_type_inc: VectorData
- doc: Index for series.
  name: series_index
  neurodata_type_inc: VectorIndex
- dims:
  - num_rows
  doc: Array of unique identifiers for the rows of this dynamic table.
  dtype: int
  name: id
  neurodata_type_inc: ElementIdentifiers
  shape:
  -
- doc: Vector columns, including index columns, of this dynamic table.
  neurodata_type_inc: VectorData
  quantity: '*'
doc: '[DEPRECATED] Table used to group different PatchClampSeries. SweepTable is being
  replaced by IntracellularRecordingsTable and SimultaneousRecordingsTable tables.
  Additional SequentialRecordingsTable, RepetitionsTable, and ExperimentalConditions
  tables provide enhanced support for experiment metadata.'
neurodata_type_def: SweepTable
neurodata_type_inc: DynamicTable
```

5.10.9 IntracellularElectrodesTable

Extends: DynamicTable

Description: see Section 4.9.9

YAML Specification:


```yaml
attributes:
- doc: Description of what is in this dynamic table.
  dtype: text
  name: description
  value: Table for storing intracellular electrode related metadata.
- dims:
  - num_columns
  doc: The names of the columns in this table. This should be used to specify an order
    to the columns.
  dtype: text
  name: colnames
  shape:
  -
datasets:
- doc: Column for storing the reference to the intracellular electrode.
  dtype:
    reftype: object
    target_type: IntracellularElectrode
  name: electrode
  neurodata_type_inc: VectorData
- dims:
  - num_rows
  doc: Array of unique identifiers for the rows of this dynamic table.
  dtype: int
  name: id
  neurodata_type_inc: ElementIdentifiers
  shape:
  -
- doc: Vector columns, including index columns, of this dynamic table.
  neurodata_type_inc: VectorData
  quantity: '*'
doc: Table for storing intracellular electrode related metadata.
neurodata_type_def: IntracellularElectrodesTable
neurodata_type_inc: DynamicTable
```

5.10.10 IntracellularStimuliTable

Extends: DynamicTable

Description: see Section 4.9.10

YAML Specification:

```yaml
attributes:
- doc: Description of what is in this dynamic table.
  dtype: text
  name: description
  value: Table for storing intracellular stimulus related metadata.
- dims:
  - num_columns
  doc: The names of the columns in this table. This should be used to specify an order
    to the columns.
  dtype: text
  name: colnames
  shape:
  -
datasets:
- doc: Column storing the reference to the recorded stimulus for the recording (rows).
  name: stimulus
  neurodata_type_inc: TimeSeriesReferenceVectorData
- dims:
  - num_rows
  doc: Array of unique identifiers for the rows of this dynamic table.
  dtype: int
  name: id
  neurodata_type_inc: ElementIdentifiers
  shape:
  -
- doc: Vector columns, including index columns, of this dynamic table.
  neurodata_type_inc: VectorData
  quantity: '*'
doc: Table for storing intracellular stimulus related metadata.
neurodata_type_def: IntracellularStimuliTable
neurodata_type_inc: DynamicTable
```

5.10.11 IntracellularResponsesTable

Extends: DynamicTable

Description: see Section 4.9.11

YAML Specification:


```yaml
attributes:
- doc: Description of what is in this dynamic table.
  dtype: text
  name: description
  value: Table for storing intracellular response related metadata.
- dims:
  - num_columns
  doc: The names of the columns in this table. This should be used to specify an order
    to the columns.
  dtype: text
  name: colnames
  shape:
  -
datasets:
- doc: Column storing the reference to the recorded response for the recording (rows)
  name: response
  neurodata_type_inc: TimeSeriesReferenceVectorData
- dims:
  - num_rows
  doc: Array of unique identifiers for the rows of this dynamic table.
  dtype: int
  name: id
  neurodata_type_inc: ElementIdentifiers
  shape:
  -
- doc: Vector columns, including index columns, of this dynamic table.
  neurodata_type_inc: VectorData
  quantity: '*'
doc: Table for storing intracellular response related metadata.
neurodata_type_def: IntracellularResponsesTable
neurodata_type_inc: DynamicTable
```

5.10.12 IntracellularRecordingsTable

Extends: AlignedDynamicTable

Description: see Section 4.9.12

YAML Specification:

```yaml
attributes:
- doc: Description of the contents of this table. Inherited from AlignedDynamicTable
    and overwritten here to fix the value of the attribute.
  dtype: text
  name: description
  value: A table to group together a stimulus and response from a single electrode
    and a single simultaneous recording and for storing metadata about the intracellular
    recording.
- dims:
  - num_categories
  doc: The names of the categories in this AlignedDynamicTable. Each category is represented
    by one DynamicTable stored in the parent group. This attribute should be used
    to specify an order of categories and the category names must match the names
    of the corresponding DynamicTable in the group.
  dtype: text
  name: categories
  shape:
  -
- dims:
  - num_columns
  doc: The names of the columns in this table. This should be used to specify an order
    to the columns.
  dtype: text
  name: colnames
  shape:
  -
datasets:
- dims:
  - num_rows
  doc: Array of unique identifiers for the rows of this dynamic table.
  dtype: int
  name: id
  neurodata_type_inc: ElementIdentifiers
  shape:
  -
- doc: Vector columns, including index columns, of this dynamic table.
  neurodata_type_inc: VectorData
  quantity: '*'
doc: A table to group together a stimulus and response from a single electrode and
  a single simultaneous recording. Each row in the table represents a single recording
  consisting typically of a stimulus and a corresponding response. In some cases,
  however, only a stimulus or a response is recorded as part of an experiment. In
  this case, both the stimulus and response will point to the same TimeSeries while
  the idx_start and count of the invalid column will be set to -1, thus, indicating
  that no values have been recorded for the stimulus or response, respectively. Note,
```

(continues on next page)

(continued from previous page)

```yaml
  a recording MUST contain at least a stimulus or a response. Typically the stimulus
  and response are PatchClampSeries. However, the use of AD/DA channels that are not
  associated to an electrode is also common in intracellular electrophysiology, in
  which case other TimeSeries may be used.
groups:
- doc: Table for storing intracellular electrode related metadata.
  name: electrodes
  neurodata_type_inc: IntracellularElectrodesTable
- doc: Table for storing intracellular stimulus related metadata.
  name: stimuli
  neurodata_type_inc: IntracellularStimuliTable
- doc: Table for storing intracellular response related metadata.
  name: responses
  neurodata_type_inc: IntracellularResponsesTable
- doc: A DynamicTable representing a particular category for columns in the AlignedDynamicTable
    parent container. The table MUST be aligned with (i.e., have the same number of
    rows) as all other DynamicTables stored in the AlignedDynamicTable parent container.
    The name of the category is given by the name of the DynamicTable and its description
    by the description attribute of the DynamicTable.
  neurodata_type_inc: DynamicTable
  quantity: '*'
name: intracellular_recordings
neurodata_type_def: IntracellularRecordingsTable
neurodata_type_inc: AlignedDynamicTable
```

5.10.13 SimultaneousRecordingsTable

Extends: DynamicTable

Description: see Section 4.9.13

YAML Specification:


```yaml
attributes:
- dims:
  - num_columns
  doc: The names of the columns in this table. This should be used to specify an order
    to the columns.
  dtype: text
  name: colnames
  shape:
  -
- doc: Description of what is in this dynamic table.
  dtype: text
  name: description
datasets:
- attributes:
  - doc: Reference to the IntracellularRecordingsTable table that this table region
      applies to. This specializes the attribute inherited from DynamicTableRegion
      to fix the type of table that can be referenced here.
    dtype:
      reftype: object
      target_type: IntracellularRecordingsTable
    name: table
  doc: A reference to one or more rows in the IntracellularRecordingsTable table.
  name: recordings
  neurodata_type_inc: DynamicTableRegion
- doc: Index dataset for the recordings column.
  name: recordings_index
  neurodata_type_inc: VectorIndex
- dims:
  - num_rows
  doc: Array of unique identifiers for the rows of this dynamic table.
  dtype: int
  name: id
  neurodata_type_inc: ElementIdentifiers
  shape:
  -
- doc: Vector columns, including index columns, of this dynamic table.
  neurodata_type_inc: VectorData
  quantity: '*'
doc: A table for grouping different intracellular recordings from the IntracellularRecordingsTable
  table together that were recorded simultaneously from different electrodes.
name: simultaneous_recordings
neurodata_type_def: SimultaneousRecordingsTable
neurodata_type_inc: DynamicTable
```

5.10.14 SequentialRecordingsTable

Extends: DynamicTable

Description: see Section 4.9.14

YAML Specification:

```yaml
attributes:
- dims:
  - num_columns
  doc: The names of the columns in this table. This should be used to specify an order
    to the columns.
  dtype: text
  name: colnames
  shape:
  -
- doc: Description of what is in this dynamic table.
  dtype: text
  name: description
datasets:
- attributes:
  - doc: Reference to the SimultaneousRecordingsTable table that this table region
      applies to. This specializes the attribute inherited from DynamicTableRegion
      to fix the type of table that can be referenced here.
    dtype:
      reftype: object
      target_type: SimultaneousRecordingsTable
    name: table
  doc: A reference to one or more rows in the SimultaneousRecordingsTable table.
  name: simultaneous_recordings
  neurodata_type_inc: DynamicTableRegion
- doc: Index dataset for the simultaneous_recordings column.
  name: simultaneous_recordings_index
  neurodata_type_inc: VectorIndex
- doc: The type of stimulus used for the sequential recording.
  dtype: text
  name: stimulus_type
  neurodata_type_inc: VectorData
- dims:
  - num_rows
  doc: Array of unique identifiers for the rows of this dynamic table.
  dtype: int
  name: id
  neurodata_type_inc: ElementIdentifiers
  shape:
  -
- doc: Vector columns, including index columns, of this dynamic table.
  neurodata_type_inc: VectorData
  quantity: '*'
doc: A table for grouping different sequential recordings from the SimultaneousRecordingsTable
  table together. This is typically used to group together sequential recordings where
  a sequence of stimuli of the same type with varying parameters have been presented
```

(continues on next page)

(continued from previous page)

```
  in a sequence.
name: sequential_recordings
neurodata_type_def: SequentialRecordingsTable
neurodata_type_inc: DynamicTable
```

5.10.15 RepetitionsTable

Extends: DynamicTable

Description: see Section 4.9.15

YAML Specification:

```yaml
attributes:
- dims:
  - num_columns
  doc: The names of the columns in this table. This should be used to specify an order
    to the columns.
  dtype: text
  name: colnames
  shape:
  -
- doc: Description of what is in this dynamic table.
  dtype: text
  name: description
datasets:
- attributes:
  - doc: Reference to the SequentialRecordingsTable table that this table region applies
      to. This specializes the attribute inherited from DynamicTableRegion to fix
      the type of table that can be referenced here.
    dtype:
      reftype: object
      target_type: SequentialRecordingsTable
    name: table
  doc: A reference to one or more rows in the SequentialRecordingsTable table.
  name: sequential_recordings
  neurodata_type_inc: DynamicTableRegion
- doc: Index dataset for the sequential_recordings column.
  name: sequential_recordings_index
  neurodata_type_inc: VectorIndex
- dims:
  - num_rows
  doc: Array of unique identifiers for the rows of this dynamic table.
  dtype: int
  name: id
  neurodata_type_inc: ElementIdentifiers
  shape:
  -
- doc: Vector columns, including index columns, of this dynamic table.
  neurodata_type_inc: VectorData
  quantity: '*'
doc: A table for grouping different sequential intracellular recordings together.
  With each SequentialRecording typically representing a particular type of stimulus,
  the RepetitionsTable table is typically used to group sets of stimuli applied in
  sequence.
name: repetitions
neurodata_type_def: RepetitionsTable
neurodata_type_inc: DynamicTable
```

5.10.16 ExperimentalConditionsTable

Extends: DynamicTable

Description: see Section 4.9.16

YAML Specification:


```yaml
attributes:
- dims:
  - num_columns
  doc: The names of the columns in this table. This should be used to specify an order
    to the columns.
  dtype: text
  name: colnames
  shape:
  -
- doc: Description of what is in this dynamic table.
  dtype: text
  name: description
datasets:
- attributes:
  - doc: Reference to the RepetitionsTable table that this table region applies to.
      This specializes the attribute inherited from DynamicTableRegion to fix the
      type of table that can be referenced here.
    dtype:
      reftype: object
      target_type: RepetitionsTable
    name: table
  doc: A reference to one or more rows in the RepetitionsTable table.
  name: repetitions
  neurodata_type_inc: DynamicTableRegion
- doc: Index dataset for the repetitions column.
  name: repetitions_index
  neurodata_type_inc: VectorIndex
- dims:
  - num_rows
  doc: Array of unique identifiers for the rows of this dynamic table.
  dtype: int
  name: id
  neurodata_type_inc: ElementIdentifiers
  shape:
  -
- doc: Vector columns, including index columns, of this dynamic table.
  neurodata_type_inc: VectorData
  quantity: '*'
doc: A table for grouping different intracellular recording repetitions together that
  belong to the same experimental condition.
name: experimental_conditions
neurodata_type_def: ExperimentalConditionsTable
neurodata_type_inc: DynamicTable
```

5.11 Optogenetics

This source module contains neurodata_types for opto-genetics data.

5.11.1 OptogeneticSeries

Extends: *TimeSeries*

Description: see Section 4.10.1

YAML Specification:


```yaml
attributes:
- default_value: no description
  doc: Description of the time series.
  dtype: text
  name: description
  required: false
- default_value: no comments
  doc: 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.
  dtype: text
  name: comments
  required: false
datasets:
- attributes:
  - doc: Unit of measurement for data, which is fixed to 'watts'.
    dtype: text
    name: unit
    value: watts
  - default_value: 1.0
    doc: Scalar to multiply each element in data to convert it to the specified 'unit'.
      If the data are stored in acquisition system units or other units that require
      a conversion to be interpretable, multiply the data by 'conversion' to convert
      the data to the specified 'unit'. e.g. if the data acquisition system stores
      values in this object as signed 16-bit integers (int16 range -32,768 to 32,767)
      that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system
      gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition
      values to recorded volts is 2.5/32768/8000 = 9.5367e-9.
    dtype: float32
    name: conversion
    required: false
  - default_value: 0.0
    doc: Scalar to add to the data after scaling by 'conversion' to finalize its coercion
      to the specified 'unit'. Two common examples of this include (a) data stored
      in an unsigned type that requires a shift after scaling to re-center the data,
      and (b) specialized recording devices that naturally cause a scalar offset with
      respect to the true units.
    dtype: float32
    name: offset
    required: false
  - default_value: -1.0
```


(continues on next page)

(continued from previous page)

```
    doc: 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 -1.0.
    dtype: float32
    name: resolution
    required: false
  - doc: Optionally describe the continuity of the data. Can be "continuous", "instantaneous",
      or "step". For example, a voltage trace would be "continuous", because samples
      are recorded from a continuous process. An array of lick times would be "instantaneous",
      because the data represents distinct moments in time. Times of image presentations
      would be "step" because the picture remains the same until the next timepoint.
      This field is optional, but is useful in providing information about the underlying
      data. It may inform the way this data is interpreted, the way it is visualized,
      and what analysis methods are applicable.
    dtype: text
    name: continuity
    required: false
  dims:
  - num_times
  doc: Applied power for optogenetic stimulus, in watts.
  dtype: numeric
  name: data
  shape:
  -
- attributes:
  - doc: Sampling rate, in Hz.
    dtype: float32
    name: rate
  - doc: Unit of measurement for time, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  doc: Timestamp of the first sample in seconds. When timestamps are uniformly spaced,
    the timestamp of the first sample can be specified and all subsequent ones calculated
    from the sampling rate attribute.
  dtype: float64
  name: starting_time
  quantity: '?'
- attributes:
  - doc: Value is '1'
    dtype: int32
    name: interval
    value: 1
  - doc: Unit of measurement for timestamps, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  dims:
  - num_times
  doc: Timestamps for samples stored in data, in seconds, relative to the common
```

(continues on next page)

(continued from previous page)

```
↪experiment
      master-clock stored in NWBFile.timestamps_reference_time.
    dtype: float64
    name: timestamps
    quantity: '?'
    shape:
    -
  - dims:
    - num_times
    doc: Numerical labels that apply to each time point in data for the purpose of querying
      and slicing data by these values. If present, the length of this array should
      be the same size as the first dimension of data.
    dtype: uint8
    name: control
    quantity: '?'
    shape:
    -
  - dims:
    - num_control_values
    doc: Description of each control value. Must be present if control is present. If
      present, control_description[0] should describe time points where control == 0.
    dtype: text
    name: control_description
    quantity: '?'
    shape:
    -
  doc: An optogenetic stimulus.
  groups:
  - doc: 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.
    name: sync
    quantity: '?'
  links:
  - doc: Link to OptogeneticStimulusSite object that describes the site to which this
      stimulus was applied.
    name: site
    target_type: OptogeneticStimulusSite
  neurodata_type_def: OptogeneticSeries
  neurodata_type_inc: TimeSeries
```

5.11.2 OptogeneticStimulusSite

Extends: *NWBContainer*

Description: see Section 4.10.2

YAML Specification:

```yaml
datasets:
- doc: Description of stimulation site.
  dtype: text
  name: description
- doc: Excitation wavelength, in nm.
  dtype: float32
  name: excitation_lambda
- doc: Location of the stimulation site. Specify the area, layer, comments on estimation
    of area/layer, stereotaxic coordinates if in vivo, etc. Use standard atlas names
    for anatomical regions when possible.
  dtype: text
  name: location
doc: A site of optogenetic stimulation.
links:
- doc: Device that generated the stimulus.
  name: device
  target_type: Device
neurodata_type_def: OptogeneticStimulusSite
neurodata_type_inc: NWBContainer
```

5.12 Optical physiology

This source module contains neurodata_types for optical physiology data.

5.12.1 OnePhotonSeries

Extends: *ImageSeries*

Description: see Section 4.11.1

YAML Specification:

```yaml
attributes:
- doc: Photomultiplier gain.
  dtype: float32
  name: pmt_gain
  required: false
- doc: 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.
  dtype: float32
  name: scan_line_rate
  required: false
- doc: Exposure time of the sample; often the inverse of the frequency.
  dtype: float32
  name: exposure_time
  required: false
- doc: Amount of pixels combined into 'bins'; could be 1, 2, 4, 8, etc.
  dtype: uint8
  name: binning
  required: false
- doc: Power of the excitation in mW, if known.
  dtype: float32
  name: power
  required: false
- doc: Intensity of the excitation in mW/mm^2, if known.
  dtype: float32
  name: intensity
  required: false
- default_value: no description
  doc: Description of the time series.
  dtype: text
  name: description
  required: false
- default_value: no comments
  doc: 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.
  dtype: text
  name: comments
  required: false
datasets:
- attributes:
```

(continues on next page)

(continued from previous page)

```
- default_value: 1.0
  doc: Scalar to multiply each element in data to convert it to the specified 'unit'.
    If the data are stored in acquisition system units or other units that require
    a conversion to be interpretable, multiply the data by 'conversion' to convert
    the data to the specified 'unit'. e.g. if the data acquisition system stores
    values in this object as signed 16-bit integers (int16 range -32,768 to 32,767)
    that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system
    gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition
    values to recorded volts is 2.5/32768/8000 = 9.5367e-9.
  dtype: float32
  name: conversion
  required: false
- default_value: 0.0
  doc: Scalar to add to the data after scaling by 'conversion' to finalize its coercion
    to the specified 'unit'. Two common examples of this include (a) data stored
    in an unsigned type that requires a shift after scaling to re-center the data,
    and (b) specialized recording devices that naturally cause a scalar offset with
    respect to the true units.
  dtype: float32
  name: offset
  required: false
- default_value: -1.0
  doc: 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 -1.0.
  dtype: float32
  name: resolution
  required: false
- doc: Base unit of measurement for working with the data. Actual stored values
    are not necessarily stored in these units. To access the data in these units,
    multiply 'data' by 'conversion' and add 'offset'.
  dtype: text
  name: unit
- doc: Optionally describe the continuity of the data. Can be "continuous", "instantaneous",
    or "step". For example, a voltage trace would be "continuous", because samples
    are recorded from a continuous process. An array of lick times would be "instantaneous",
    because the data represents distinct moments in time. Times of image presentations
    would be "step" because the picture remains the same until the next timepoint.
    This field is optional, but is useful in providing information about the underlying
    data. It may inform the way this data is interpreted, the way it is visualized,
    and what analysis methods are applicable.
  dtype: text
  name: continuity
  required: false
dims:
- - frame
  - x
  - y
- - frame
  - x
```

(continues on next page)

(continued from previous page)

```
  - y
  - z
 doc: Binary data representing images across frames. If data are stored in an external
  file, this should be an empty 3D array.
 dtype: numeric
 name: data
 shape:
 - -
  -
  -
 - -
  -
  -
  -
- dims:
 - rank
 doc: Number of pixels on x, y, (and z) axes.
 dtype: int32
 name: dimension
 quantity: '?'
 shape:
 -
- attributes:
 - dims:
  - num_files
  doc: Each external image may contain one or more consecutive frames of the full
   ImageSeries. This attribute serves as an index to indicate which frames each
   file contains, to faciliate random access. The 'starting_frame' attribute, hence,
   contains a list of frame numbers within the full ImageSeries of the first frame
   of each file listed in the parent 'external_file' dataset. Zero-based indexing
   is used (hence, the first element will always be zero). For example, if the
   'external_file' dataset has three paths to files and the first file has 5 frames,
   the second file has 10 frames, and the third file has 20 frames, then this attribute
   will have values [0, 5, 15]. If there is a single external file that holds all
   of the frames of the ImageSeries (and so there is a single element in the 'external_file'
   dataset), then this attribute should have value [0].
  dtype: int32
  name: starting_frame
  shape:
  -
 dims:
 - num_files
 doc: Paths to one or more external file(s). The field is only present if format='external'.
  This is only relevant if the image series 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 NWB file and that file is linked to this file.
 dtype: text
 name: external_file
 quantity: '?'
```

(continues on next page)

(continued from previous page)

```yaml
  shape:
  -
- default_value: raw
  doc: Format of image. If this is 'external', then the attribute 'external_file'
    contains the path information to the image files. If this is 'raw', then the raw
    (single-channel) binary data is stored in the 'data' dataset. If this attribute
    is not present, then the default format='raw' case is assumed.
  dtype: text
  name: format
  quantity: '?'
- attributes:
  - doc: Sampling rate, in Hz.
    dtype: float32
    name: rate
  - doc: Unit of measurement for time, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  doc: Timestamp of the first sample in seconds. When timestamps are uniformly spaced,
    the timestamp of the first sample can be specified and all subsequent ones calculated
    from the sampling rate attribute.
  dtype: float64
  name: starting_time
  quantity: '?'
- attributes:
  - doc: Value is '1'
    dtype: int32
    name: interval
    value: 1
  - doc: Unit of measurement for timestamps, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  dims:
  - num_times
  doc: Timestamps for samples stored in data, in seconds, relative to the common experiment
    master-clock stored in NWBFile.timestamps_reference_time.
  dtype: float64
  name: timestamps
  quantity: '?'
  shape:
  -
- dims:
  - num_times
  doc: Numerical labels that apply to each time point in data for the purpose of querying
    and slicing data by these values. If present, the length of this array should
    be the same size as the first dimension of data.
  dtype: uint8
  name: control
  quantity: '?'
  shape:
```

(continues on next page)

(continued from previous page)

```yaml
  -
- dims:
  - num_control_values
  doc: Description of each control value. Must be present if control is present. If
    present, control_description[0] should describe time points where control == 0.
  dtype: text
  name: control_description
  quantity: '?'
  shape:
  -
doc: Image stack recorded over time from 1-photon microscope.
groups:
- doc: 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.
  name: sync
  quantity: '?'
links:
- doc: Link to ImagingPlane object from which this TimeSeries data was generated.
  name: imaging_plane
  target_type: ImagingPlane
- doc: Link to the Device object that was used to capture these images.
  name: device
  quantity: '?'
  target_type: Device
neurodata_type_def: OnePhotonSeries
neurodata_type_inc: ImageSeries
```

5.12.2 TwoPhotonSeries

Extends: *ImageSeries*

Description: see Section 4.11.2

YAML Specification:

```yaml
attributes:
- doc: Photomultiplier gain.
  dtype: float32
  name: pmt_gain
  required: false
- doc: 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.
  dtype: float32
  name: scan_line_rate
  required: false
- default_value: no description
  doc: Description of the time series.
  dtype: text
  name: description
  required: false
- default_value: no comments
  doc: 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.
  dtype: text
  name: comments
  required: false
datasets:
- dims:
  - - width|height
  - - width|height|depth
  doc: Width, height and depth of image, or imaged area, in meters.
  dtype: float32
  name: field_of_view
  quantity: '?'
  shape:
  - - 2
  - - 3
- attributes:
  - default_value: 1.0
    doc: Scalar to multiply each element in data to convert it to the specified 'unit'.
      If the data are stored in acquisition system units or other units that require
      a conversion to be interpretable, multiply the data by 'conversion' to convert
      the data to the specified 'unit'. e.g. if the data acquisition system stores
      values in this object as signed 16-bit integers (int16 range -32,768 to 32,767)
      that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system
      gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition
      values to recorded volts is 2.5/32768/8000 = 9.5367e-9.
    dtype: float32
    name: conversion
```

(continues on next page)

(continued from previous page)

```yaml
    required: false
  - default_value: 0.0
    doc: Scalar to add to the data after scaling by 'conversion' to finalize its coercion
      to the specified 'unit'. Two common examples of this include (a) data stored
      in an unsigned type that requires a shift after scaling to re-center the data,
      and (b) specialized recording devices that naturally cause a scalar offset with
      respect to the true units.
    dtype: float32
    name: offset
    required: false
  - default_value: -1.0
    doc: 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 -1.0.
    dtype: float32
    name: resolution
    required: false
  - doc: Base unit of measurement for working with the data. Actual stored values
      are not necessarily stored in these units. To access the data in these units,
      multiply 'data' by 'conversion' and add 'offset'.
    dtype: text
    name: unit
  - doc: Optionally describe the continuity of the data. Can be "continuous", "instantaneous",
      or "step". For example, a voltage trace would be "continuous", because samples
      are recorded from a continuous process. An array of lick times would be "instantaneous",
      because the data represents distinct moments in time. Times of image presentations
      would be "step" because the picture remains the same until the next timepoint.
      This field is optional, but is useful in providing information about the underlying
      data. It may inform the way this data is interpreted, the way it is visualized,
      and what analysis methods are applicable.
    dtype: text
    name: continuity
    required: false
  dims:
  - - frame
    - x
    - y
  - - frame
    - x
    - y
    - z
  doc: Binary data representing images across frames. If data are stored in an external
    file, this should be an empty 3D array.
  dtype: numeric
  name: data
  shape:
  - -
    -
    -
  - -
```

(continues on next page)

(continued from previous page)

```
    -
    -
    -
- dims:
  - rank
  doc: Number of pixels on x, y, (and z) axes.
  dtype: int32
  name: dimension
  quantity: '?'
  shape:
  -
- attributes:
  - dims:
    - num_files
    doc: Each external image may contain one or more consecutive frames of the full
      ImageSeries. This attribute serves as an index to indicate which frames each
      file contains, to faciliate random access. The 'starting_frame' attribute, hence,
      contains a list of frame numbers within the full ImageSeries of the first frame
      of each file listed in the parent 'external_file' dataset. Zero-based indexing
      is used (hence, the first element will always be zero). For example, if the
      'external_file' dataset has three paths to files and the first file has 5 frames,
      the second file has 10 frames, and the third file has 20 frames, then this␣
↪attribute
      will have values [0, 5, 15]. If there is a single external file that holds all
      of the frames of the ImageSeries (and so there is a single element in the
↪'external_file'
      dataset), then this attribute should have value [0].
    dtype: int32
    name: starting_frame
    shape:
    -
  dims:
  - num_files
  doc: Paths to one or more external file(s). The field is only present if format=
↪'external'.
    This is only relevant if the image series 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 NWB file and that file is linked to this file.
  dtype: text
  name: external_file
  quantity: '?'
  shape:
  -
- default_value: raw
  doc: Format of image. If this is 'external', then the attribute 'external_file'
    contains the path information to the image files. If this is 'raw', then the raw
    (single-channel) binary data is stored in the 'data' dataset. If this attribute
    is not present, then the default format='raw' case is assumed.
  dtype: text
  name: format
  quantity: '?'
- attributes:
```

(continues on next page)

(continued from previous page)

```
  - doc: Sampling rate, in Hz.
    dtype: float32
    name: rate
  - doc: Unit of measurement for time, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  doc: Timestamp of the first sample in seconds. When timestamps are uniformly spaced,
    the timestamp of the first sample can be specified and all subsequent ones calculated
    from the sampling rate attribute.
  dtype: float64
  name: starting_time
  quantity: '?'
- attributes:
  - doc: Value is '1'
    dtype: int32
    name: interval
    value: 1
  - doc: Unit of measurement for timestamps, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  dims:
  - num_times
  doc: Timestamps for samples stored in data, in seconds, relative to the common experiment
    master-clock stored in NWBFile.timestamps_reference_time.
  dtype: float64
  name: timestamps
  quantity: '?'
  shape:
  -
- dims:
  - num_times
  doc: Numerical labels that apply to each time point in data for the purpose of querying
    and slicing data by these values. If present, the length of this array should
    be the same size as the first dimension of data.
  dtype: uint8
  name: control
  quantity: '?'
  shape:
  -
- dims:
  - num_control_values
  doc: Description of each control value. Must be present if control is present. If
    present, control_description[0] should describe time points where control == 0.
  dtype: text
  name: control_description
  quantity: '?'
  shape:
  -
doc: Image stack recorded over time from 2-photon microscope.
```

(continues on next page)

(continued from previous page)

```
groups:
- doc: 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.
  name: sync
  quantity: '?'
links:
- doc: Link to ImagingPlane object from which this TimeSeries data was generated.
  name: imaging_plane
  target_type: ImagingPlane
- doc: Link to the Device object that was used to capture these images.
  name: device
  quantity: '?'
  target_type: Device
neurodata_type_def: TwoPhotonSeries
neurodata_type_inc: ImageSeries
```

5.12.3 RoiResponseSeries

Extends: *TimeSeries*

Description: see Section 4.11.3

YAML Specification:


```yaml
attributes:
- default_value: no description
  doc: Description of the time series.
  dtype: text
  name: description
  required: false
- default_value: no comments
  doc: 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.
  dtype: text
  name: comments
  required: false
datasets:
- attributes:
  - default_value: 1.0
    doc: Scalar to multiply each element in data to convert it to the specified 'unit'.
      If the data are stored in acquisition system units or other units that require
      a conversion to be interpretable, multiply the data by 'conversion' to convert
      the data to the specified 'unit'. e.g. if the data acquisition system stores
      values in this object as signed 16-bit integers (int16 range -32,768 to 32,767)
      that correspond to a 5V range (-2.5V to 2.5V), and the data acquisition system
      gain is 8000X, then the 'conversion' multiplier to get from raw data acquisition
      values to recorded volts is 2.5/32768/8000 = 9.5367e-9.
    dtype: float32
    name: conversion
    required: false
  - default_value: 0.0
    doc: Scalar to add to the data after scaling by 'conversion' to finalize its coercion
      to the specified 'unit'. Two common examples of this include (a) data stored
      in an unsigned type that requires a shift after scaling to re-center the data,
      and (b) specialized recording devices that naturally cause a scalar offset with
      respect to the true units.
    dtype: float32
    name: offset
    required: false
  - default_value: -1.0
    doc: 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 -1.0.
    dtype: float32
    name: resolution
    required: false
  - doc: Base unit of measurement for working with the data. Actual stored values
      are not necessarily stored in these units. To access the data in these units,
      multiply 'data' by 'conversion' and add 'offset'.
```


(continues on next page)

(continued from previous page)

```yaml
    dtype: text
    name: unit
  - doc: Optionally describe the continuity of the data. Can be "continuous",
↪"instantaneous",
      or "step". For example, a voltage trace would be "continuous", because samples
      are recorded from a continuous process. An array of lick times would be
↪"instantaneous",
      because the data represents distinct moments in time. Times of image presentations
      would be "step" because the picture remains the same until the next timepoint.
      This field is optional, but is useful in providing information about the underlying
      data. It may inform the way this data is interpreted, the way it is visualized,
      and what analysis methods are applicable.
    dtype: text
    name: continuity
    required: false
  dims:
  - - num_times
  - - num_times
    - num_ROIs
  doc: Signals from ROIs.
  dtype: numeric
  name: data
  shape:
  - -
  - -
    -
- doc: DynamicTableRegion referencing into an ROITable containing information on the
    ROIs stored in this timeseries.
  name: rois
  neurodata_type_inc: DynamicTableRegion
- attributes:
  - doc: Sampling rate, in Hz.
    dtype: float32
    name: rate
  - doc: Unit of measurement for time, which is fixed to 'seconds'.
    dtype: text
    name: unit
    value: seconds
  doc: Timestamp of the first sample in seconds. When timestamps are uniformly spaced,
    the timestamp of the first sample can be specified and all subsequent ones calculated
    from the sampling rate attribute.
  dtype: float64
  name: starting_time
  quantity: '?'
- attributes:
  - doc: Value is '1'
    dtype: int32
    name: interval
    value: 1
  - doc: Unit of measurement for timestamps, which is fixed to 'seconds'.
    dtype: text
    name: unit
```

(continues on next page)

(continued from previous page)

```yaml
    value: seconds
  dims:
  - num_times
  doc: Timestamps for samples stored in data, in seconds, relative to the common␣
↪experiment
    master-clock stored in NWBFile.timestamps_reference_time.
  dtype: float64
  name: timestamps
  quantity: '?'
  shape:
  -
- dims:
  - num_times
  doc: Numerical labels that apply to each time point in data for the purpose of querying
    and slicing data by these values. If present, the length of this array should
    be the same size as the first dimension of data.
  dtype: uint8
  name: control
  quantity: '?'
  shape:
  -
- dims:
  - num_control_values
  doc: Description of each control value. Must be present if control is present. If
    present, control_description[0] should describe time points where control == 0.
  dtype: text
  name: control_description
  quantity: '?'
  shape:
  -
doc: ROI responses over an imaging plane. The first dimension represents time. The
  second dimension, if present, represents ROIs.
groups:
- doc: 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.
  name: sync
  quantity: '?'
neurodata_type_def: RoiResponseSeries
neurodata_type_inc: TimeSeries
```

5.12.4 DfOverF

Extends: *NWBDataInterface*

Description: see Section 4.11.4

YAML Specification:


```yaml
default_name: DfOverF
doc: dF/F information about a region of interest (ROI). Storage hierarchy of dF/F
  should be the same as for segmentation (i.e., same names for ROIs and for image
  planes).
groups:
- doc: RoiResponseSeries object(s) containing dF/F for a ROI.
  neurodata_type_inc: RoiResponseSeries
  quantity: +
neurodata_type_def: DfOverF
neurodata_type_inc: NWBDataInterface
```

5.12.5 Fluorescence

Extends: *NWBDataInterface*

Description: see Section 4.11.5

YAML Specification:


```yaml
default_name: Fluorescence
doc: 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).
groups:
- doc: RoiResponseSeries object(s) containing fluorescence data for a ROI.
  neurodata_type_inc: RoiResponseSeries
  quantity: +
neurodata_type_def: Fluorescence
neurodata_type_inc: NWBDataInterface
```

5.12.6 ImageSegmentation

Extends: *NWBDataInterface*

Description: see Section 4.11.6

YAML Specification:


```yaml
default_name: ImageSegmentation
doc: 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.
groups:
- doc: Results from image segmentation of a specific imaging plane.
  neurodata_type_inc: PlaneSegmentation
  quantity: +
neurodata_type_def: ImageSegmentation
neurodata_type_inc: NWBDataInterface
```

5.12.7 PlaneSegmentation

Extends: DynamicTable

Description: see Section 4.11.7

YAML Specification:

```yaml
attributes:
- dims:
  - num_columns
  doc: The names of the columns in this table. This should be used to specify an order
    to the columns.
  dtype: text
  name: colnames
  shape:
  -
- doc: Description of what is in this dynamic table.
  dtype: text
  name: description
datasets:
- dims:
  - - num_roi
    - num_x
    - num_y
  - - num_roi
    - num_x
    - num_y
    - num_z
  doc: ROI masks for each ROI. Each image mask is the size of the original imaging
    plane (or volume) and members of the ROI are finite non-zero.
  name: image_mask
  neurodata_type_inc: VectorData
  quantity: '?'
  shape:
  - -
    -
    -
  - -
    -
    -
    -
- doc: Index into pixel_mask.
  name: pixel_mask_index
  neurodata_type_inc: VectorIndex
  quantity: '?'
- doc: 'Pixel masks for each ROI: a list of indices and weights for the ROI. Pixel
    masks are concatenated and parsing of this dataset is maintained by the
↪PlaneSegmentation'
  dtype:
  - doc: Pixel x-coordinate.
    dtype: uint32
    name: x
  - doc: Pixel y-coordinate.
```

(continues on next page)

(continued from previous page)

```yaml
    dtype: uint32
    name: y
  - doc: Weight of the pixel.
    dtype: float32
    name: weight
  name: pixel_mask
  neurodata_type_inc: VectorData
  quantity: '?'
- doc: Index into voxel_mask.
  name: voxel_mask_index
  neurodata_type_inc: VectorIndex
  quantity: '?'
- doc: 'Voxel masks for each ROI: a list of indices and weights for the ROI. Voxel
    masks are concatenated and parsing of this dataset is maintained by the␣
↪PlaneSegmentation'
  dtype:
  - doc: Voxel x-coordinate.
    dtype: uint32
    name: x
  - doc: Voxel y-coordinate.
    dtype: uint32
    name: y
  - doc: Voxel z-coordinate.
    dtype: uint32
    name: z
  - doc: Weight of the voxel.
    dtype: float32
    name: weight
  name: voxel_mask
  neurodata_type_inc: VectorData
  quantity: '?'
- dims:
  - num_rows
  doc: Array of unique identifiers for the rows of this dynamic table.
  dtype: int
  name: id
  neurodata_type_inc: ElementIdentifiers
  shape:
  -
- doc: Vector columns, including index columns, of this dynamic table.
  neurodata_type_inc: VectorData
  quantity: '*'
doc: Results from image segmentation of a specific imaging plane.
groups:
- doc: Image stacks that the segmentation masks apply to.
  groups:
  - doc: One or more image stacks that the masks apply to (can be one-element stack).
    neurodata_type_inc: ImageSeries
    quantity: '*'
  name: reference_images
links:
- doc: Link to ImagingPlane object from which this data was generated.
```

(continues on next page)

(continued from previous page)

```yaml
    name: imaging_plane
    target_type: ImagingPlane
  neurodata_type_def: PlaneSegmentation
  neurodata_type_inc: DynamicTable
```

5.12.8 ImagingPlane

Extends: *NWBContainer*

Description: see Section 4.11.8

YAML Specification:

```yaml
datasets:
- doc: Description of the imaging plane.
  dtype: text
  name: description
  quantity: '?'
- doc: Excitation wavelength, in nm.
  dtype: float32
  name: excitation_lambda
- doc: Rate that images are acquired, in Hz. If the corresponding TimeSeries is present,
    the rate should be stored there instead.
  dtype: float32
  name: imaging_rate
  quantity: '?'
- doc: Calcium indicator.
  dtype: text
  name: indicator
- doc: Location of the imaging plane. Specify the area, layer, comments on estimation
    of area/layer, stereotaxic coordinates if in vivo, etc. Use standard atlas names
    for anatomical regions when possible.
  dtype: text
  name: location
- attributes:
  - default_value: 1.0
    doc: Scalar to multiply each element in data to convert it to the specified 'unit'.
      If the data are stored in acquisition system units or other units that require
      a conversion to be interpretable, multiply the data by 'conversion' to convert
      the data to the specified 'unit'. e.g. if the data acquisition system stores
      values in this object as pixels from x = -500 to 499, y = -500 to 499 that correspond
      to a 2 m x 2 m range, then the 'conversion' multiplier to get from raw data
      acquisition pixel units to meters is 2/1000.
    dtype: float32
    name: conversion
    required: false
  - default_value: meters
    doc: Base unit of measurement for working with the data. The default value is
      'meters'.
    dtype: text
    name: unit
    required: false
  dims:
  - - height
    - width
    - x, y, z
  - - height
    - width
```

(continues on next page)

(continued from previous page)

```yaml
    - depth
    - x, y, z
  doc: DEPRECATED Physical position of each pixel. 'xyz' represents the position of
    the pixel relative to the defined coordinate space. Deprecated in favor of origin_
↪coords
    and grid_spacing.
  dtype: float32
  name: manifold
  quantity: '?'
  shape:
  - -
    -
    - 3
  - -
    -
    -
    - 3
- attributes:
  - default_value: meters
    doc: Measurement units for origin_coords. The default value is 'meters'.
    dtype: text
    name: unit
    required: false
  dims:
  - - x, y
  - - x, y, z
  doc: Physical location of the first element of the imaging plane (0, 0) for 2-D
    data or (0, 0, 0) for 3-D data. See also reference_frame for what the physical
    location is relative to (e.g., bregma).
  dtype: float32
  name: origin_coords
  quantity: '?'
  shape:
  - - 2
  - - 3
- attributes:
  - default_value: meters
    doc: Measurement units for grid_spacing. The default value is 'meters'.
    dtype: text
    name: unit
    required: false
  dims:
  - - x, y
  - - x, y, z
  doc: Space between pixels in (x, y) or voxels in (x, y, z) directions, in the specified
    unit. Assumes imaging plane is a regular grid. See also reference_frame to interpret
    the grid.
  dtype: float32
  name: grid_spacing
  quantity: '?'
  shape:
  - - 2
```

(continues on next page)

(continued from previous page)

```
  - - 3
- doc: Describes reference frame of origin_coords and grid_spacing. For example, this
    can be a text description of the anatomical location and orientation of the grid
    defined by origin_coords and grid_spacing or the vectors needed to transform or
    rotate the grid to a common anatomical axis (e.g., AP/DV/ML). This field is necessary
    to interpret origin_coords and grid_spacing. If origin_coords and grid_spacing
    are not present, then this field is not required. For example, if the microscope
    takes 10 x 10 x 2 images, where the first value of the data matrix (index (0,
    0, 0)) corresponds to (-1.2, -0.6, -2) mm relative to bregma, the spacing between
    pixels is 0.2 mm in x, 0.2 mm in y and 0.5 mm in z, and larger numbers in x means
    more anterior, larger numbers in y means more rightward, and larger numbers in
    z means more ventral, then enter the following -- origin_coords = (-1.2, -0.6,
    -2) grid_spacing = (0.2, 0.2, 0.5) reference_frame = "Origin coordinates are relative
    to bregma. First dimension corresponds to anterior-posterior axis (larger index
    = more anterior). Second dimension corresponds to medial-lateral axis (larger
    index = more rightward). Third dimension corresponds to dorsal-ventral axis (larger
    index = more ventral)."
  dtype: text
  name: reference_frame
  quantity: '?'
doc: An imaging plane and its metadata.
groups:
- doc: An optical channel used to record from an imaging plane.
  neurodata_type_inc: OpticalChannel
  quantity: +
links:
- doc: Link to the Device object that was used to record from this electrode.
  name: device
  target_type: Device
neurodata_type_def: ImagingPlane
neurodata_type_inc: NWBContainer
```

5.12.9 OpticalChannel

Extends: *NWBContainer*

Description: see Section 4.11.9

YAML Specification:

```yaml
datasets:
- doc: Description or other notes about the channel.
  dtype: text
  name: description
- doc: Emission wavelength for channel, in nm.
  dtype: float32
  name: emission_lambda
doc: An optical channel used to record from an imaging plane.
neurodata_type_def: OpticalChannel
neurodata_type_inc: NWBContainer
```

5.12.10 MotionCorrection

Extends: *NWBDataInterface*

Description: see Section 4.11.10

YAML Specification:

```yaml
default_name: MotionCorrection
doc: '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).'
groups:
- doc: Reuslts from motion correction of an image stack.
  neurodata_type_inc: CorrectedImageStack
  quantity: +
neurodata_type_def: MotionCorrection
neurodata_type_inc: NWBDataInterface
```

5.12.11 CorrectedImageStack

Extends: *NWBDataInterface*

Description: see Section 4.11.11

YAML Specification:

```yaml
doc: Reuslts from motion correction of an image stack.
groups:
- doc: Image stack with frames shifted to the common coordinates.
  name: corrected
  neurodata_type_inc: ImageSeries
- doc: Stores the x,y delta necessary to align each frame to the common coordinates,
    for example, to align each frame to a reference image.
  name: xy_translation
  neurodata_type_inc: TimeSeries
links:
- doc: Link to ImageSeries object that is being registered.
  name: original
  target_type: ImageSeries
neurodata_type_def: CorrectedImageStack
neurodata_type_inc: NWBDataInterface
```

5.13 Retinotopy

This source module contains neurodata_type for retinotopy data.

5.13.1 ImagingRetinotopy

Extends: *NWBDataInterface*

Description: see Section 4.12.1

YAML Specification:

```yaml
datasets:
- attributes:
  - dims:
    - num_rows, num_cols
    doc: 'Number of rows and columns in the image. NOTE: row, column representation
      is equivalent to height, width.'
    dtype: int32
    name: dimension
    shape:
    - 2
  - dims:
    - height, width
    doc: Size of viewing area, in meters.
    dtype: float32
    name: field_of_view
    shape:
    - 2
  - doc: Unit that axis data is stored in (e.g., degrees).
    dtype: text
    name: unit
  dims:
  - num_rows
  - num_cols
  doc: Phase response to stimulus on the first measured axis.
  dtype: float32
  name: axis_1_phase_map
  shape:
  -
  -
- attributes:
  - dims:
    - num_rows, num_cols
    doc: 'Number of rows and columns in the image. NOTE: row, column representation
      is equivalent to height, width.'
    dtype: int32
    name: dimension
    shape:
    - 2
  - dims:
    - height, width
    doc: Size of viewing area, in meters.
```

(continues on next page)

(continued from previous page)

```
    dtype: float32
    name: field_of_view
    shape:
    - 2
  - doc: Unit that axis data is stored in (e.g., degrees).
    dtype: text
    name: unit
  dims:
  - num_rows
  - num_cols
  doc: 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.
  dtype: float32
  name: axis_1_power_map
  quantity: '?'
  shape:
  -
  -
- attributes:
  - dims:
    - num_rows, num_cols
    doc: 'Number of rows and columns in the image. NOTE: row, column representation
      is equivalent to height, width.'
    dtype: int32
    name: dimension
    shape:
    - 2
  - dims:
    - height, width
    doc: Size of viewing area, in meters.
    dtype: float32
    name: field_of_view
    shape:
    - 2
  - doc: Unit that axis data is stored in (e.g., degrees).
    dtype: text
    name: unit
  dims:
  - num_rows
  - num_cols
  doc: Phase response to stimulus on the second measured axis.
  dtype: float32
  name: axis_2_phase_map
  shape:
  -
  -
- attributes:
  - dims:
    - num_rows, num_cols
    doc: 'Number of rows and columns in the image. NOTE: row, column representation
      is equivalent to height, width.'
    dtype: int32
```

(continues on next page)

(continued from previous page)

```
    name: dimension
    shape:
    - 2
  - dims:
    - height, width
    doc: Size of viewing area, in meters.
    dtype: float32
    name: field_of_view
    shape:
    - 2
  - doc: Unit that axis data is stored in (e.g., degrees).
    dtype: text
    name: unit
  dims:
  - num_rows
  - num_cols
  doc: 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.
  dtype: float32
  name: axis_2_power_map
  quantity: '?'
  shape:
  -
  -
- dims:
  - axis_1, axis_2
  doc: Two-element array describing the contents of the two response axis fields.
    Description should be something like ['altitude', 'azimuth'] or '['radius', 'theta'].
  dtype: text
  name: axis_descriptions
  shape:
  - 2
- attributes:
  - doc: Number of bits used to represent each value. This is necessary to determine
      maximum (white) pixel value.
    dtype: int32
    name: bits_per_pixel
  - dims:
    - num_rows, num_cols
    doc: 'Number of rows and columns in the image. NOTE: row, column representation
      is equivalent to height, width.'
    dtype: int32
    name: dimension
    shape:
    - 2
  - dims:
    - height, width
    doc: Size of viewing area, in meters.
    dtype: float32
    name: field_of_view
    shape:
    - 2
```

(continues on next page)

(continued from previous page)

```yaml
  - doc: Focal depth offset, in meters.
    dtype: float32
    name: focal_depth
  - doc: Format of image. Right now only 'raw' is supported.
    dtype: text
    name: format
  dims:
  - num_rows
  - num_cols
  doc: 'Gray-scale image taken with same settings/parameters (e.g., focal depth, wavelength)
    as data collection. Array format: [rows][columns].'
  dtype: uint16
  name: focal_depth_image
  quantity: '?'
  shape:
  -
  -
- attributes:
  - dims:
    - num_rows, num_cols
    doc: 'Number of rows and columns in the image. NOTE: row, column representation
      is equivalent to height, width.'
    dtype: int32
    name: dimension
    shape:
    - 2
  - dims:
    - height, width
    doc: Size of viewing area, in meters.
    dtype: float32
    name: field_of_view
    shape:
    - 2
  dims:
  - num_rows
  - num_cols
  doc: Sine of the angle between the direction of the gradient in axis_1 and axis_2.
  dtype: float32
  name: sign_map
  quantity: '?'
  shape:
  -
  -
- attributes:
  - doc: Number of bits used to represent each value. This is necessary to determine
      maximum (white) pixel value
    dtype: int32
    name: bits_per_pixel
  - dims:
    - num_rows, num_cols
    doc: 'Number of rows and columns in the image. NOTE: row, column representation
```

(continues on next page)

(continued from previous page)

```yaml
      is equivalent to height, width.'
    dtype: int32
    name: dimension
    shape:
    - 2
  - dims:
    - height, width
    doc: Size of viewing area, in meters.
    dtype: float32
    name: field_of_view
    shape:
    - 2
  - doc: Format of image. Right now only 'raw' is supported.
    dtype: text
    name: format
  dims:
  - num_rows
  - num_cols
  doc: 'Gray-scale anatomical image of cortical surface. Array structure: [rows][columns]'
  dtype: uint16
  name: vasculature_image
  shape:
  -
  -
default_name: ImagingRetinotopy
doc: '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. This group
  does not store the raw responses imaged during retinotopic mapping or the stimuli
  presented, but rather the resulting phase and power maps after applying a Fourier
  transform on the averaged responses. 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).'
neurodata_type_def: ImagingRetinotopy
neurodata_type_inc: NWBDataInterface
```

Part III

History and Legal

CHAPTER

SIX

MAKING A PULL REQUEST

Actions to take on each PR that modifies the schema and does not prepare the schema for a public release (this is also in the GitHub PR template):

If the current schema version on "dev" is a public release, then:

1. Update the version string in `docs/format/source/conf.py` and `common/namespace.yaml` to the next version with the suffix "-alpha"
2. Add a new section in the release notes for the new version with the date "Upcoming"

Always:

1. Add release notes for the PR to `docs/format/source/format_release_notes.rst`

Documentation or internal changes to the repo (i.e., changes that do not affect the schema files) do not need to be accompanied with a version bump or addition to the release notes.

CHAPTER

SEVEN

MERGING PRS AND MAKING RELEASES

Public release: a tagged release of the schema. The version string MUST NOT have a suffix indicating a pre-release, such as "-alpha". The current "dev" branch of PyNWB and all PyNWB releases MUST point to a public release of nwb-schema. All schema that use nwb-schema as a submodule MUST also point only to public releases.

Internal release: a state of the schema "dev" branch where the version string ends with "-alpha".

The default branch of nwb-schema is "dev". **The "dev" branch holds the bleeding edge version of the nwb-schema specification.**

PRs should be made to "dev". Every PR should include an update to `docs/format/source/format_release_notes.rst`. If the current version is a public release, then the PR should also update the version of the schema in two places: `docs/format/source/conf.py` and `core/nwb.namespace.yaml`. The new version should be the next bugfix/minor/major version of the schema with the suffix "-alpha". For example, if the current schema on "dev" has version "2.2.0", then a PR implementing a bug fix should update the schema version from "2.2.0" to "2.2.1-alpha". Appending the "-alpha" suffix ensures that any person or API accessing the default "dev" branch of the repo containing an internal release of the schema receives the schema with a version string that is distinct from public releases of the schema. If the current schema on "dev" is already an internal release, then the version string does not need to be updated unless the PR requires an upgrade in the version (e.g., from bugfix to minor).

PyNWB should contain a branch and PR that tracks the "dev" branch of nwb-schema. Before a public release of nwb-schema is made, this PyNWB branch should be checked to ensure that when the new release is made, the branch can be merged without issue.

Immediately prior to making a new public release, the version of the schema should be updated to remove the "-alpha" suffix and the documentation and release notes should be updated as needed (see next section).

The current "dev" branch of PyNWB and all PyNWB releases MUST always point to a public release of nwb-schema. If a public release contains an internally released version of nwb-schema, e.g., from an untagged commit on the "dev" branch, then it will be difficult to find the version (commit) of nwb-schema that was used to create an NWB file when the schema is not cached.

CHAPTER

EIGHT

MAKING A RELEASE CHECKLIST

Before merging:

1. Update requirements versions as needed
2. Update legal file dates and information in `Legal.txt`, `license.txt`, `README.md`, `docs/format/source/conf.py`, and any other locations as needed
3. Update `README.rst` as needed
4. Update the version string in `docs/format/source/conf.py`, `core/nwb.namespace.yaml`, and `/core/nwb.file.yaml` (remove "-alpha" suffix)
5. Update `docs/format/source/conf.py` as needed
6. Update release notes (set release date) in `docs/format/source/format_release_notes.rst` and any other docs as needed
7. Test docs locally (`cd docs/format; make fulldoc`) where the nwb-schema submodule in the local version of PyNWB is fully up-to-date with the head of the dev branch.
8. Push changes to a new PR and make sure all PRs to be included in this release have been merged. Add `?template=release.md` to the PR URL to auto-populate the PR with this checklist.
9. Check that the readthedocs build for this PR succeeds (build latest to pull the new branch, then activate and build docs for new branch): https://readthedocs.org/projects/nwb-schema/builds/

After merging:

1. Create a new git tag. Pull the latest dev branch locally, run `git tag [version] --sign`, copy and paste the release notes into the tag message, and run `git push --tags`.
2. On the GitHub tags page, click "..." -> "Create release" for the new tag on the right side of the page. Copy and paste the release notes into the release message, update the formatting if needed (reST to Markdown), and set the title to the version string.
3. Check that the readthedocs "latest" and "stable" builds run and succeed. Delete the readthedocs build for the merged PR. https://readthedocs.org/projects/nwb-schema/builds/
4. Update the nwb-schema submodule in the PyNWB branch corresponding to this schema version to point to the tagged commit.

This checklist can also be found in the GitHub release PR template.

The time between merging this PR and creating a new public release should be minimized.

CHAPTER NINE

THE “DEV” BRANCH AND RELEASES

The default branch is “dev”. The “dev” branch will hold the bleeding edge version of the NWB format specification, language specification, and storage specification. PRs should be made to “dev”. Every PR should include an update to the corresponding format/language/storage release notes. If the PR involves a change to the NWB format, the PR should also update the version of the format in three places: */docs/format/conf.py*, */core/nwb.namespace.yaml* and */core/nwb.file.yaml*. The new version string should be the next bugfix/minor/major version of the format with the suffix “a” (for “alpha”). For example, if the format is currently in version “2.2.0” and the format is then updated / released internally with a bug fix, then the PR for that bug fix should update the format version from “2.2.0” to “2.2.1a”. Appending the “a” suffix ensures that any person or API accessing the default dev branch of the repo containing an internally released version of the schema receives the schema with a version string that is distinct from the full public released versions of the schema.

Before merging a PR on nwb-schema, developers should take care to test their changes locally with the latest version of PyNWB and MatNWB to ensure compatibility. If the APIs require changes to work with the PR, those changes should be implemented and tested locally before merging the PR to ensure that the API changes can be implemented and no further changes to the schema are required.

When a new public release is ready, the branches of the APIs, PyNWB and MatNWB, that track nwb-schema should be checked to ensure that when the new release is made, the branches in the APIs can be merged without issue.

Immediately prior to a new public release, the version of the format should be updated to remove the any alphabetic, suffixes, e.g., “a”, “b”, and “rc”, and the documentation and release notes should be checked and updated (see release process documents).

It is important that all public releases of nwb-schema contain a publicly released version of hdmf-common-schema. If a public release contains an internally released version of hdmf-common-schema, e.g., from an untagged commit on the “dev” branch, then tracking the identity of the included hdmf-common-schema becomes difficult and the same version string could point to two different versions of the schema.

For the same reason, it is important that all public releases of the APIs, PyNWB and MatNWB, contain a publicly released version of nwb-schema. Starting with nwb-schema version 2.2.0, the dev branch and all public releases of PyNWB and MatNWB include only publicly released versions of nwb-schema. For more details, see the [PyNWB software process documentation](https://pynwb.readthedocs.io/en/stable/software_process.html).

The [NWB Extensions Versioning Guidelines](https://nwb-extensions.github.io/versioning_guidelines) are used to guide versioning of the NWB core schema, as well as extensions to NWB.

CHAPTER TEN

RELEASE NOTES

10.1 2.8.0 (Upcoming)

10.1.1 Minor changes

- Added support to set boundary metadata for `SpatialSeries`. (#524)
- Added columns `waveform_mean_index` and `waveform_sd_index` to `Units` to make the `waveform_mean` and `waveform_sd` columns ragged. This allows for a different number of waveform means/SDs per unit which is useful when each unit is associated with a different number of electrodes and there is a waveform mean/SD for each electrode and unit. (#576)

10.2 2.7.0 (February 7, 2024)

10.2.1 Minor changes

- Fixed typos in docstrings. (#560)
- Deprecated `ImagingRetinotopy` neurodata type. (#565)
- Modified `OptogeneticSeries` to allow 2D data, primarily in extensions of `OptogeneticSeries`. (#564)
- Added optional `stimulus_template` column to `IntracellularStimuliTable` as part of the `IntracellularRecordingsTable`. (#545)
- Added support for `NWBDataInterface` and `DynamicTable` in `NWBFile.stimulus`. (#559)

10.3 2.6.0 (January 17, 2023)

10.3.1 Minor changes

- Added OnePhotonSeries. (#523)
- `Subject.age` has a new optional attribute, `reference`, which can take a value of “birth” (default) or “gestational”. (#525)
- Added “in seconds” to the doc of Units.spike_times. (#530)

10.4 2.5.0 (June 14, 2022)

10.4.1 Major changes

- Shape of SpatialSeries.data is more restrictive to prevent > 3 columns. (#510)

10.4.2 Minor changes

- The elements *x*, *y*, *z*, *imp* and *filtering* are now optional instead of required. (#506)
- Added an `offset` attribute to all `TimeSeries` objects to allow enhanced translation to scientific units. (#494)
- Allowed `NWBFile/stimulus/templates` to contain `Images` objects. (#459)
- Added new optional "order_of_images" dataset to `Images` that contains an ordered list of object references to `Image` objects stored in the same `Images` object. This dataset must be used if the images are referred to by index, e.g., from an `IndexSeries` object. Created new neurodata type `ImageReferences` which should be used for this dataset. (#459, #518, #519, #520)
- Overhauled `IndexSeries` type (#459): - Fixed dtype of `data` dataset of `IndexSeries` (int32 -> uint32). - Updated `unit` attribute of `data` to have fixed value "N/A". - Updated docstrings for the `conversion`, `resolution`, and `offset` attributes of `data` to indicate that these fields are not used. - Added link to an `Images` object, which contains an ordered collection of images. - Discouraged use of the `indexed_timeseries` link to an `ImageSeries`.
- Updated `TimeIntervals` to use the new `TimeSeriesReferenceVectorData` type. This does not alter the overall structure of `TimeIntervals` in a major way aside from changing the value of the `neurodata_type` attribute in the file from `VectorData` to `` `TimeSeriesReferenceVectorData ``. This change replaces the existing `TimeIntervals.timeseries` column with a `TimeSeriesReferenceVectorData` type column of the same name and overall schema. This change facilitates creating common functionality around `TimeSeriesReferenceVectorData`. This change affects all existing `TimeIntervals` tables as part of the `intervals/` group, i.e., `intervals/epochs`, `intervals/trials`, and `intervals/invalid_times`. (#486)
- Clarified the doc string for the `reference` column of the electrodes table. (#498)
- Added `cell_id` field to `IntracellularElectrode`. (#512)

10.5 2.4.0 (Aug. 11, 2021)

10.5.1 Major changes

- Added new `TimeSeriesReferenceVectorData` type for referencing ranges of `TimeSeries` from a `VectorData` column (#470)
- Integrated the intracellular electrophysiology experiment metadata table structure developed as part of the ndx-icephys-meta extension project with NWB (#470). This includes the following new types:
  - `IntracellularRecordingsTable` is an `AlignedDynamicTable` for managing individual intracellular recordings and to group together a stimulus and response from a single electrode recording. The table contains the following category tables:
    * `IntracellularElectrodesTable`; a `DynamicTable` for storing metadata about the `IntracellularElectrode` used
    * `IntracellularStimuliTable`; a `DynamicTable` for storing metadata about the recorded stimulus `TimeSeries` using the new `TimeSeriesReferenceVectorData` type to reference `TimeSeries`

    * IntracellularResponsesTable; a DynamicTable for storing metadata about the recorded response TimeSeries using the new TimeSeriesReferenceVectorData type to reference TimeSeries

  - SimultaneousRecordingsTable is a DynamicTable for grouping different intracellular recordings from the IntracellularRecordingsTable together that were recorded simultaneously from different electrodes and for storing metadata about simultaneous recordings
  - SequentialRecordingsTable is a DynamicTable for grouping different sequential recordings from the SimultaneousRecordingsTable together and storing metadata about sequential recordings
  - RepetitionsTable a DynamicTable for grouping different sequential intracellular recordings from the SequentialRecordingsTable together and storing metadata about repetitions
  - ExperimentalConditionsTable is a DynamicTable for grouping different intracellular recording repetitions from the RepetitionsTable together and storing metadata about experimental conditions

- Added the new intracellular electrophysiology metadata tables to /general/intracellular_ephys as part of NWBFile (#470)

10.5.2 Deprecations

- SweepTable has been deprecated in favor of the new intracellular electrophysiology metadata tables. Use of SweepTable is still possible but no longer recommended. (#470)
- /general/intracellular_ephys/filtering has been deprecated in favor of IntracellularElectrode.filtering (#470)

10.5.3 Bug Fixes

- Fixed incorrect dtype for electrodes table column "filtering" (float -> text) (#478)
- Removed quantity: * from the type definitions of OptogeneticStimulusSite and ImagingPlane. This change improves clarity of the schema to follow best practices. It has no functional effect on the schema. (#472)
- Updated ImageSeries to have its data dataset be required. Since ImageSeries is a TimeSeries and TimeSeries.data is required, ImageSeries.data should also be a required dataset. Otherwise this creates problems for inheritance and validation. If ImageSeries data are stored in an external file, then ImageSeries.data should be set to an empty 3D array. (#481)

10.6 2.3.0 (May 12, 2021)

- Add optional waveforms column to the Units table.
- Add optional strain field to Subject.
- Add to DecompositionSeries an optional DynamicTableRegion called source_channels.
- Add to ImageSeries an optional link to Device.
- Add optional continuity field to TimeSeries.
- Add optional filtering attribute to ElectricalSeries.
- Clarify documentation for electrode impedance and filtering.
- Add description of extra fields.
- Set the stimulus_description for IZeroCurrentClamp to have the fixed value N/A.

- Update hdmf-common-schema from 1.1.3 to version 1.5.0. - The HDMF-experimental namespace was added, which includes the `ExternalResources` and `EnumData` data types. Schema in the HDMF-experimental namespace are experimental and subject to breaking changes at any time. - Added experimental data type `ExternalResources` for storing ontology information / external resource references. - Added experimental data type `EnumData` to store data from a set of fixed values. - Changed dtype for datasets within `CSRMatrix` from 'int' to 'uint' and added missing `data_type_inc: Container` to the `CSRMatrix` type. - Added data type `SimpleMultiContainer`, a Container for storing other Container and Data objects together. - Added data type `AlignedDynamicTable`, a DynamicTable type with support for categories (or sub-headings) each described by a separate DynamicTable. - Fixed missing dtype for `VectorIndex`. - `VectorIndex` now extends `VectorData` instead of `Index`. - Removed unused and non-functional `Index` data type. - See https://hdmf-common-schema.readthedocs.io/en/latest/format_release_notes.html for full release notes.

10.7 2.2.5 (May 29, 2020)

- Add schema validation CI.
- Fix incorrect dims and shape for `ImagingPlane.origin_coords` and `ImagingPlane.grid_spacing`, and fix incorrect dims for `TwoPhotonSeries.field_of_view`.

10.8 2.2.4 (April 14, 2020)

- Fix typo in `nwb.ophys.yaml` that prevents proper parsing of the schema.

10.9 2.2.3 (April 13, 2020)

- Move nested type definitions to root of YAML files. This does not functionally change the schema but simplifies parsing of the schema and extensions by APIs.
- Make `ImagingPlane.imaging_rate` optional to handle cases where an imaging plane is associated with multiple time series with different rates.
- Add release process documentation.

10.10 2.2.2 (March 2, 2020)

- Fix shape and dims of `OpticalSeries.data` for color images
- Allow more than one `OpticalChannel` object in `ImagingPlane`
- Update hdmf-common-schema to 1.1.3. This fixes missing 'shape' and 'dims' key for types `VectorData`, `VectorIndex`, and `DynamicTableRegion`.
- Revert changes to `nwb.retinotopy.yaml` in 2.1.0 which break backward compatibility and were not supported by the APIs in any case. Changes will be revisited in a future version.

10.11 2.2.1 (January 14, 2020)

- Fixed incorrect version numbers in `nwb.file.yaml` and `hdmf-common-schema/common/namespace.yaml`.

10.12 2.2.0 (January 6, 2020)

- Moved common data structures such as Container and DynamicTable to hdmf-common-schema.
  - The hdmf-common-schema repo is now included as a submodule
  - See https://github.com/NeurodataWithoutBorders/nwb-schema/pull/307 for details
- Added "channel_conversion" dataset to ElectricalSeries to represent per-channel conversion factors.
- Added "sampling_rate" and "unit" attributes to "waveform_mean" and "waveform_sd" datasets/columns in Units table.
- Added "description" and "manufacturer" attributes to Device.
- Deprecated ImagingPlane "manifold" in favor of "origin_coords" and "grid_spacing"
- Use "text" data type for all DynamicTable "colnames". Previously, only ASCII was allowed.
- Use "text" data type for electrode table columns "location" and "group_name". Previously, only ASCII was allowed.
- Added to description to make electrode x,y,z consistent with CCF reference. https://allensdk.readthedocs.io/en/latest/reference_space.html
- Added "position" dataset with compound data type x,y,z in ElectrodeGroup.
- Avoid enforcing "uint64" for sweep numbers for better compatibility. Use uint instead which is 32bit.
- Set *dtype* for *Image* and its subtypes to *numeric*. (note: technically this breaks backwards compatibility, in the schema, but the *pynwb* API has always enforced that Images have a numeric type, and realistically we do not think users are storing strings in an *Image* dataset.)
- Added "resolution" attribute to "spike_times" column of Units.
- Changed the "quantity" key of attribute Units.resolution to "required" for schema language compliance.
- Removed "required" key from dataset ImageSeries.field_of_view for schema language compliance.
- Replaced "required" keys with "quantity" keys for ImagingPlane.origin_coords and ImagingPlane.grid_spacing for schema language compliance.
- Refactored ImagingRetinotopy type to reduce redundancy.
- Added "doc" key to ImagingRetinotopy.axis_2_power_map for schema language compliance.
- Fixed makefiles for generating documentation on Windows.
- Added optional "reference" column in "electrodes" table.
- Changed dims of ImageSeries from (frame, y, x) to (frame, x, y) and (frame, z, y, x) to (frame, x, y, z) to be consistent with the dimension ordering in PlaneSegmentation.
- Changed dims of Image from (y, x) to (x, y). (note: as far as we know, users of NWB 2.0 that use the Image type encode their data as (x, y)) to be consistent with the dimension ordering in ImageSeries.
- Updated hdmf-common-schema to version 1.1.0 which includes:
  - The 'colnames' attribute of `DynamicTable` changed from data type 'ascii' to 'text'.

  - Improved documentation and type docstrings.

10.13 2.1.0 (September 2019)

- Improved documentation in "doc" attribute of many types
- Removed "help" attribute
  - Now that the schema is cached in an NWB file by default, this attribute is redundant, confusing, used inconsistently, clutters the file and documentation, and adds substantial boilerplate to writing extensions
  - See https://github.com/NeurodataWithoutBorders/nwb-schema/issues/270 for details
- Removed static "description" attribute from some types
  - These were intended to be a "help" attribute, which has now been removed
  - For example, TimeIntervals dataset "start_time" attribute "description" had a fixed value that is now removed
- Reordered keys
  - This standardizes the order of keys across types and makes the schema more readable
  - See https://github.com/NeurodataWithoutBorders/nwb-schema/issues/274 for details
- Added "dims" attribute for datasets where "shape" was specified without "dims"
  - The "dims" attribute describes the data along each dimension of the dataset and is helpful to provide alongside "shape"
  - For example, NWBFile dataset "keywords" has attribute "shape" has one entry: "null". The attribute "dims" was added with one entry: "num_keywords"
- Removed redundant specifications that are inherited from a parent type
- ElectrodeGroup link "device": optional -> required
  - This was previously required by PyNWB
  - See https://github.com/NeurodataWithoutBorders/pynwb/issues/1025 for details
- Matched default and fixed values of datasets and attributes with the documentation and intended use
  - IZeroClampSeries dataset "bias_current" unspecified value -> fixed value 0.0
  - IZeroClampSeries dataset "bridge_balance" unspecified value -> fixed value 0.0
  - IZeroClampSeries dataset "capacitance_compensation" unspecified value -> fixed value 0.0
  - TimeSeries dataset "resolution" default value: 0.0 -> -1.0
  - ImagingRetinotopy dataset "axis_descriptions" attribute "shape": null -> 2
  - DecompositionSeries dataset "data" attribute "unit" default value unspecified -> default value "no unit"
  - VoltageClampStimulusSeries, CurrentClampSeries, IZeroClampSeries attribute "unit" has fixed value "volts"
  - CurrentClampStimulusSeries, VoltageClampSeries, attribute "unit" has fixed value "amperes"
- NWBFile dataset "experimenter" and "related_publications" change from scalar to 1-D, unlimited arrays
  - This allows the "experimenter" and "related_publications" dataset to encode multiple values

  - See https://github.com/NeurodataWithoutBorders/pynwb/issues/985 and https://github.com/NeurodataWithoutBorders/nwb-schema/issues/299 for details
- Standardized units to be plural, lower-case, SI units
  - TimeSeries dataset "starting_time" attribute "unit" fixed value: "Seconds" -> "seconds"
  - TimeSeries dataset "timestamps" attribute "unit" fixed value: "Seconds" -> "seconds"
  - ElectricalSeries dataset "data" attribute "unit" default value (previously optional): "volt" -> fixed value "volts"
  - SpikeEventSeries dataset "data" attribute "unit" default value (previously optional): "volt" -> fixed value "volts"
  - SpikeEventSeries dataset "timestamps" attribute "unit" fixed value (previously optional): "Seconds" -> fixed value "seconds"
  - EventDetection dataset "times" attribute "unit" default value: "Seconds" -> "seconds"
  - VoltageClampSeries dataset "capacitance_fast" attribute "unit" default value "Farad" -> fixed value "farads"
  - VoltageClampSeries dataset "capacitance_slow" attribute "unit" default value "Farad" -> fixed value "farads"
  - VoltageClampSeries dataset "resistance_comp_bandwidth" attribute "unit" default value "Hz" -> fixed value "hertz"
  - VoltageClampSeries dataset "resistance_comp_correction" attribute "unit" default value "percent" -> fixed value "percent"
  - VoltageClampSeries dataset "resistance_comp_prediction" attribute "unit" default value "percent" -> fixed value "percent"
  - VoltageClampSeries dataset "whole_cell_capacitance_comp" attribute "unit" default value "Farad" -> fixed value "farads"
  - VoltageClampSeries dataset "whole_cell_series_resistance_comp" attribute "unit" default value "Ohm" -> fixed value "ohms"
  - OptogeneticSeries dataset "data" attribute "unit" default value "watt" -> fixed value "watts"
  - ImagingPlane dataset "manifold" attribute "unit" default value "Meter" -> default value "meters"
  - see https://github.com/NeurodataWithoutBorders/nwb-schema/issues/277 for details
- Made Units table column "waveform_mean" and "waveform_sd" have shape num_units x num_samples x num_electrodes
  - See https://github.com/NeurodataWithoutBorders/pynwb/pull/1008 for details
- Made CorrectedImageStack and ImagingRetinotopy inherit from the more specific NWBDataInterface instead of NWBContainer
- Added a scratch space for saving arbitrary datasets to an NWBFile
  - NWB is cumbersome as a format for day-to-day work. There is a lot of overheard to save one-off analysis results to an NWB file. To save new datasets, a user has write an extension. This is a lot of work for a result that may just be tossed out.
  - "scratch" is now an optional top-level group under NWBFile that can hold NWBContainer groups and ScratchData datasets

- The scratch space is explicitly for non-standardized data that is not intended for reuse by others. Standard NWB types, and extensions if required, should always be used for any data that you intend to share. As such, published data should not include scratch data and a user should be able to ignore any data stored in scratch to use a file.
- See https://github.com/NeurodataWithoutBorders/nwb-schema/issues/286 for details

- Set the default value for the dataset "format" to "raw" and clarified the documentation for ImageSeries
  - See https://github.com/NeurodataWithoutBorders/nwb-schema/pull/308/files for details
- Add required attribute `object_id` to all NWB Groups and Datasets with an assigned neurodata_type
  - See https://nwb-schema.readthedocs.io/en/latest/format_description.html#common-attributes for details

Backwards compatibility: The PyNWB and MatNWB APIs can read 2.0 files with the 2.1 schema.

10.14 2.0.2 (June 2019)

- Updated authors
- Removed NWBFile subgroup "specifications" because schema is now cached
  - See https://github.com/NeurodataWithoutBorders/pynwb/pull/953 for details
- Made DecompositionSeries link "source_timeseries" optional
  - See https://github.com/NeurodataWithoutBorders/pynwb/pull/955 for details

10.15 2.0.1 (March 2019)

Change: Added `doc` and `title` descriptions for the individual source files included in the main namespace.

Reason: Enhance documentation of the namespace and facilitate presentation of types in autogenerated docs by making it easier to sort `neurodata_types` into meaningful categories (i.e., sections) with approbriate tiles and descriptions.

Backwards compatibility: No changes to the actual specification of the format are made. 2.0.1 is fully compatible with 2.0.0.

10.16 2.0.0 (January 2019)

Main release: November 2017 (Beta), November 2018 (RC), January 2019 (final)

10.16.1 Added new base data types: `NWBContainer`, `NWBData`, `NWBDataInterface`

Change: Added common base types for Groups, Datasets, and for Groups storing primary experiment data

Reason Collect common functionality and ease future evolution of the standard

Specific Changes

- *NWBContainer* defines a common base type for all Groups with a `neurodata_type` and is now the base type of all main data group types in the NWB format, including *TimeSeries*. This also means that all group types now inherit the required `help` and `source` attribute from `NWBContainer`. A number of neurodata_types have been updated to add the missing `help` (see https://github.com/NeurodataWithoutBorders/nwb-schema/pull/37/files for details)

- *NWBDataInterface* extends *NWBContainer* and replaces `Interface` from NWB 1.x. It has been renamed to ease intuition. *NWBDataInterface* serves as base type for primary data (e.g., experimental or analysis data) and is used to distinguish in the schema between non-metadata data containers and metadata containers. (see https://github.com/NeurodataWithoutBorders/nwb-schema/pull/116/files for details)
- *NWBData* defines a common base type for all Datasets with a `neurodata_type` and serves a similar function to *NWBContainer* only for Datasets instead of Groups.

10.16.2 Support general data structures for data tables and vector data

10.16.2.1 Support row-based and column-based tables

Change: Add support for storing tabular data via row-based and column-based table structures.

Reason: Simplify storage of complex metadata. Simplify storage of dynamic and variable-length metadata.

Format Changes:

- **Row-based tables:** are implemented via a change in the specification language through support for compound data types The advantage of row-based tables is that they 1) allow referencing of sets of rows via region-references to a single dataset (e.g., a set of electrodes), 2) make it easy to add rows by appending to a single dataset, 3) make it easy to read individual rows of a table (but require reading the full table to extract the data of a single column). Row-based tables are used to simplify, e.g., the organization of electrode-metadata in NWB 2 (see above). (See the specification language release notes for details about the addition of compound data types in the schema).
  - *Referencing rows in a row-based tables:* Subsets of rows can referenced directly via a region-reference to the row-based table. Subsets
  - *Referencing columns in a row-based table:* This is currently not directly supported, but could be implemented via a combination of an object-reference to the table and a list of the labels of columns.
- **Column-based tables:** are implemented via the new neurodata_type DynamicTable. A DynamicTable is simplified-speaking just a collection of an arbitrary number of VectorData table column datasets (all with equal length) and a dataset storing row ids and a dataset storing column names. The advantage of the column-based store is that it 1) makes it easy to add new columns to the table without the need for extensions and 2) the column-based storage makes it easy to read individual columns efficiently (while reading full rows requires reading from multiple datasets). DynamicTable is used, e.g., to enhance storage of trial data. (See https://github.com/NeurodataWithoutBorders/pynwb/pull/536/files)
  - *Referencing rows in column-based tables:* As DynamicTable consist of multiple datasets (compared to row-based tables which consists of a single 1D dataset with a compound datatype) is not possible to reference a set of rows with a single region reference. To address this issue, NWB defines DynamicTableRegion (added later in PR634 (PyNWB)) dataset type, which stores a list of integer indices (row index) and also has an attribute `table` with the object reference to the corresponding DynamicTable.
  - *Referencing columns in a columns-based table:* As each column is a separate dataset, columns of a column-based DynamicTable can be directly references via links, object-references and region-references.

10.16.2.2 Enable efficient storage of large numbers of vector data elements

Change Introduce neurodata_types VectorData , VectorIndex, ElementIdentifiers

Reason To efficiently store spike data as part of UnitTimes a new, more efficient data structure was required. This builds the general, reusable types to define efficient data storage for large numbers of data vectors in efficient, consolidated arrays, which enable more efficient read, write, and search (see *Improved storage of unit-based data*).

Format Changes

- VectorData : Data values from a series of data elements are concatenated into a single array. This allows all elements to be stored efficiently in a single data array.
- VectorIndex : 1D dataset of exclusive stop-indices selecting subranges in VectorData. In addition, the `target` attribute stores an object reference to the corresponding VectorData dataset. With this we can efficiently access single sub-vectors associated with single elements from the VectorData collection. An alternative approach would be store region-references as part of the VectorIndex. We opted for stop-indices mainly because they are more space-efficient and are easier to use for introspection of index values than region references.
- ElementIdentifiers : 1D array for storing unique identifiers for the elements in a VectorIndex.

See *Improved storage of unit-based data* for an illustration and specific example use in practice. See also I116 (nwb-schema) and PR382 (PyNWB) for further details.

10.16.3 Use new table and vector data structures to improve data organization

10.16.3.1 Improved organization of electrode metadata in `/general/extracellular_ephys`

Change: Consolidate metadata from related electrodes (e.g., from a single device) in a single location.

Example: Previous versions of the format specified in `/general/extracellular_ephys` for each electrode a group `<electrode_group_X>` that stores 3 text datasets with a description, device name, and location, respectively. The main `/general/extracellular_ephys group` then contained in addition the following datasets:

- `electrode_group` text array describing for each electrode_group (implicitly referenced by index) which device (shank, probe, tetrode, etc.) was used,
- `electrode_map` array with the x,y,z locations of each electrode
- `filtering`, i.e., a single string describing the filtering for all electrodes (even though each electrode might be from different devices), and iv),
- `impedance`, i.e., a single text array for impedance (i.e., the user has to know which format the string has, e.g., a float or a tuple of floats for impedance ranges).

Reason:

- Avoid explosion of the number of groups and datasets. For example, in the case of an ECoG grid with 128 channels one had to create 128 groups and corresponding datasets to store the required metadata about the electrodes using the original layout.
- Simplify access to related metadata. E.g., access to metadata from all electrodes of a single device requires resolution of a potentially large number of implicit links and access to a large number of groups (one per electrode) and datasets.
- Improve performance of metadata access operations. E.g., to access the `location` of all electrodes corresponding to a single recording in an `<ElectricalSeries>` in the original layout required iterating over a potentially large number of groups and datasets (one per electrode), hence, leading to a large number of small, independent read/write/seek operations, causing slow performance on common data accesses. Using the new layout, these kind of common data accesses can often be resolved via a single read/write

- Ease maintenance, use, and development through consolidation of related metadata

Format Changes

- Added specification of a new neurodata type <ElectrodeGroup> group. Each <ElectrodeGroup> contains the following datasets to describe the metadata of a set of related electrodes (e.g., all electrodes from a single device):
  - `description` : text dataset (for the group)
  - `device`: Soft link to the device in `/general/devices/`
  - `location`: Text description of the location of the device
- Added table-like dataset `electrodes` that consolidates all electrode-specific metadata. This is a DynamicTable describing for each electrode:
  - `id` : a user-specified unique identifier
  - `x`, `y`, `z` : The floating point coordinate for the electrode
  - `imp` : the impedance of the channel
  - `location` : The location of channel within the subject e.g. brain region
  - `filtering` : Description of hardware filtering
  - `group` : Object reference to the `ElectrodeGroup` object
  - `group_name` : The name of the `ElectrodeGroup`
- Updated `/general/extracellular_ephys` as follows:
  - Replaced `/general/extracellular_ephys/<electrode_group_X>` group (and all its contents) with the new <ElectrodeGroup>
  - Removed `/general/extracellular_ephys/electrode_map` dataset. This information is now stored in the `ElectrodeTable`.
  - Removed `/general/extracellular_ephys/electrode_group` dataset. This information is now stored in <ElectrodeGroup>/device.
  - Removed `/general/extracellular_ephys/impedance` This information is now stored in the `ElectrodeTable`.
  - Removed `/general/extracellular_ephys/filtering` This information is now stored in the `ElectrodeTable`.

Note: In NWB 2.0Beta the refactor originally used a row-based table for the `ElectrodeTable` based on a compound data type as described in #I6 (new-schema), i.e., `electrodes` was a 1D compound dataset. This was later changed to a column-based DynamicTable (see *Support row-based and column-based tables*). The main reason for this later change was mainly to avoid the need for large numbers of user-extensions to add electrode metadata (see #I623 (PyNWB) and PR634 (PyNWB) for details.) This change also removed the optional `description` column as it can be added easily by the user to the DynamicTable if required.

10.16.3.2 Improved storage of lab-specific meta-data

Reason: Labs commonly have specific meta-data associated with sessions, and we need a good way to organize this within NWB.

Changes: The datatype `LabMetaData` has been added to the schema within /general so that an extension can be added to /general by inheriting from LabMetaData.

For further details see I19 (nwb-schema) and PR751 (PyNWB).

10.16.3.3 Improved storage of Spectral Analyses (Signal Decomposition)

Reason: Labs commonly use analyses that involve frequency decomposition or bandpass filtering of neural or behavioral data, and it is difficult to standardize this data and meta-data across labs.

Changes: A new datatype, *DecompositionSeries* has been introduced to offer a common interface for labs to exchange the result of time-frequency analysis. The new type offers a DynamicTable to allow users to flexibly add features of bands, and a place to directly link to the *TimeSeries* that was used.

For further details see #I46 (nwb-schema) and #PR764 (PyNWB)

10.16.3.4 Improved storage of Images

Reason:

- **Improve consistency of schema:** Previously there was a reference to `Image` from `ImageSeries`, however `Image` was not defined in the schema
- **Support different static image types**

Changes: *Image* was added as a base type, and subtypes were defined: *GrayscaleImage*, *RGBImage*, and *RGBAImage* (The "A" in "RGBA" is for alpha, i.e., opacity).

10.16.3.5 Improved storage of ROIs

Reason:

- **Improve efficiency:** Similar to epochs, in NWB 1.x ROIs were stored as a single group per ROI. This structure is inefficient for storing large numbers of ROIs.
- **Make links explicit:** The relationship of `RoiResponseSeries` to `ROI` objects was implicit (i.e. ROI was specified by a string), so one had to know a priori which `ImageSegmentation` and `ImagingPlane` was used to produce the ROIs.
- **Support 3D ROIs:** Allow users to add 3D ROIs collected from a multi-plane image.

Changes: The main types for storing ROIs in NWB 2 are *ImageSegmentation* which stores 0 or more *PlaneSegmentation*. *PlaneSegmentation* is a DynamicTable for managing image segmentation results of a specific imaging plane. The ROIs are referenced by *RoiResponseSeries* which stores the ROI responses over an imaging plane. During the development of NWB 2 the management of ROIs has been improved several times. Here we outline the main changes (several of which were ultimately merged together in the *PlaneSegmentation* type).

1. Added neurodata_type `ImageMasks` replacing `ROI.img_mask` (from NWB 1.x) with **(a)** a 3D dataset with shape [num_rois, num_x_pixels, num_y_pixels] (i.e. an array of planar image masks) or **(b)** a 4D dataset with shape [num_rois, num_x_pixels, num_y_pixels, num_z_pixels] (i.e. an array of volumetric image masks) `ImageMasks` was subsequently merged with *PlaneSegmentation* and is represented by the VectorData table column `image_mask` in the table.

2. Added neurodata_type `PixelMasks` which replaces ROI.pix_mask/ROI.pix_mask_weight (from NWB 1.x) with a table that has columns "x", "y", and "weight" (i.e. combining ROI.pix_mask and ROI.pix_mask_weight into a single table). `PixelMasks` was subsequently merged with *PlaneSegmentation* and is represented by the VectorData dataset `pixel__mask` that is referenced from the table via the VectorIndex column `pixel_mask_index`.
3. Added analogous neurodata_type `VoxelMasks` with a table that has columns "x", "y", "z", and "weight" for 3D ROIs. `VoxelMasks` was subsequently merged with *PlaneSegmentation* and is represented by the VectorData dataset `voxel_mask` that is referenced from the table via the VectorIndex column `voxel_mask_index`.
4. Added neurodata_type `ROITable` which defines a table for storing references to the image mask and pixel mask for each ROI (see item 1,2). The `ROITable` type was subsequently merged with the *PlaneSegmentation* type and as such does no longer appear as a separate type in the NWB 2 schema but *PlaneSegmentation* takes the function of `ROITable`.
5. Added neurodata_type `ROITableRegion` for referencing a subset of elements in an ROITable. Subsequently `ROITableRegion` has been replaced by DynamicTableRegion as the `ROITable` changed to a DynamicTable and was merged with *PlaneSegmentation* (see 8.)
6. Replaced `RoiResponseSeries.roi_names` with `RoiResponseSeries.rois`, which is a DynamicTableRegion into the *PlaneSegmentation* table of ROIs (see items 3,4). (Before ROITable was converted from a row-based to a column-based table, *RoiResponseSeries.rois*` had been changed to a `ROITableRegion` which was then subsequently changed to a corresponding DynamicTableRegion)
7. Removed `RoiResponseSeries.segmentation_interface`. This information is available through `RoiResponseSeries.rois` (described above in 5.)
8. Assigned neurodata_type *PlaneSegmentation* to the image_plan group in *ImageSegmentation* and updated it to use the `ROITable`, `ImageMasks`, `PixelMasks`, and :`VoxelMasks` (see items 1-4 above). Specifically, *PlaneSegmentation* has been changed to be a DynamicTable and `ROITable`, `ImageMasks`, `PixelMasks`, and `VoxelMasks` have been merged into the *PlaneSegmentation* table, resulting in the removal of the `ROITable`, `ROITableRegion`, `ImageMasks`, `PixelMasks`, and `VoxelMasks` types.

For additional details see also:

- PR391 (PyNWB) and I118 (nwb-schema) for details on the main refactoring of ROI storage,
- PR665 (PyNWB) and I663 (PyNWB) (and previous issue I643 (PyNWB)) for details on the subsequent refactor using DynamicTable, and
- PR688 (PyNWB) and I554 (nwb-schema) for details on 3D ROIs,

10.16.3.6 Improved storage of unit-based data

In NWB 1.0.x data about spike units was stored across a number of different neurodata_types, specifically `UnitTimes`, `ClusterWaveforms`, and `Clustering`. This structure had several critical shortcomings, which were addressed in three main phases during the development of NWB 2.

Problem 1: Efficiency: In NWB 1.x each unit was stored as a separate group `unit_n` containing the `times` and `unit_description` for unit with index `n`. In cases where users have a very large number of units, this was problematic with regard to performance. To address this challenge `UnitTimes` has been restructured in NWB 2 to use the new VectorData , VectorIndex, ElementIdentifiers data structures (see *Enable efficient storage of large numbers of vector data elements*).Specifically, NWB 2 replaced `unit_n` (from NWB 1.x, also referred to by neurodata_type `SpikeUnit` in NWB 2beta) groups in `UnitTimes` with the following data:

- `unit_ids` : ElementIdentifiers dataset for storing unique ids for each element
- `spike_times_index`: VectorIndex dataset with region references into the spike times dataset
- `spike_times`: VectorData dataset storing the actual spike times data of all units in a single data array (for efficiency).

See also I116 (nwb-schema) and PR382 (PyNWB) for further details.

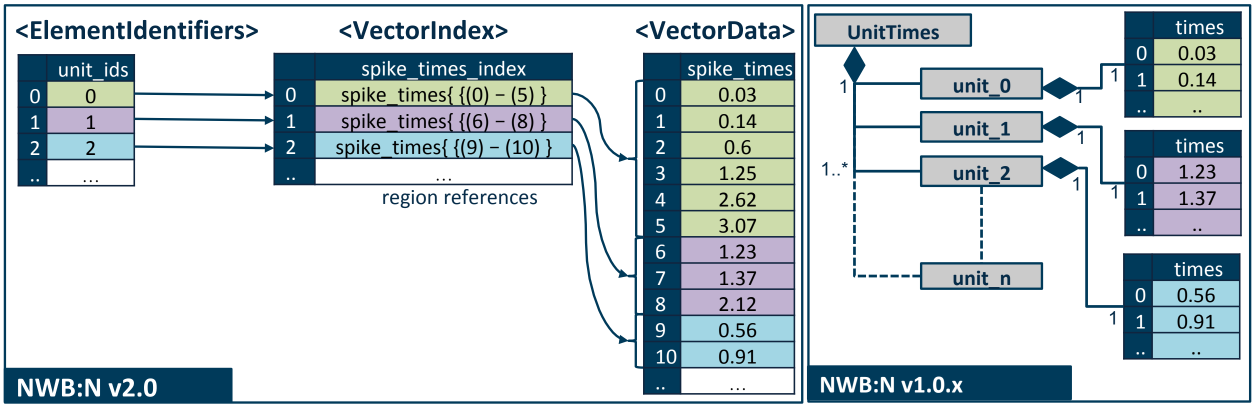


Fig. 10.1: Overview of the basic data structure for storing `UnitTimes` using the VectorData (`spike_times`), VectorIndex (`spike_times_index`), and ElementIdentifiers (`unit_ids`) data structures.

Problem 2: Dynamic Metadata: Users indicated that it was not easy to store user-defined metadata about units. To address this challenge, NWB 2 added an optional top-level group `units/` (which was subsequently moved to `/intervals/units`) which is a DynamicTable with `id` and `description` columns and optional additional user-defined table columns. See PR597 on PyNWB for detailed code changes. See the PyNWB docs for a short tutorial on how to use unit metadata. See *NWBFile Groups: /units* for an overview of the unit schema.

Problem 3: Usability: Finally, users found that storing unit data was challenging due to the fact that the information was distributed across a number of different types. To address this challenge, NWB 2.0 integrates `UnitTimes`, `ClusterWaveforms`, and `Clustering` (deprecated) into the new column-based table `units/` (i.e., `intervals/units`) (which still uses the optimized vector data storage to efficiently store spike times). See for discussions and I674 on PyNWB (and related I675 on PyNWB) and the pull request PR684 on PyNWB for detailed changes.

Together these changes have resulted in the following improved structure for storing unit data and metadata in NWB 2.0.

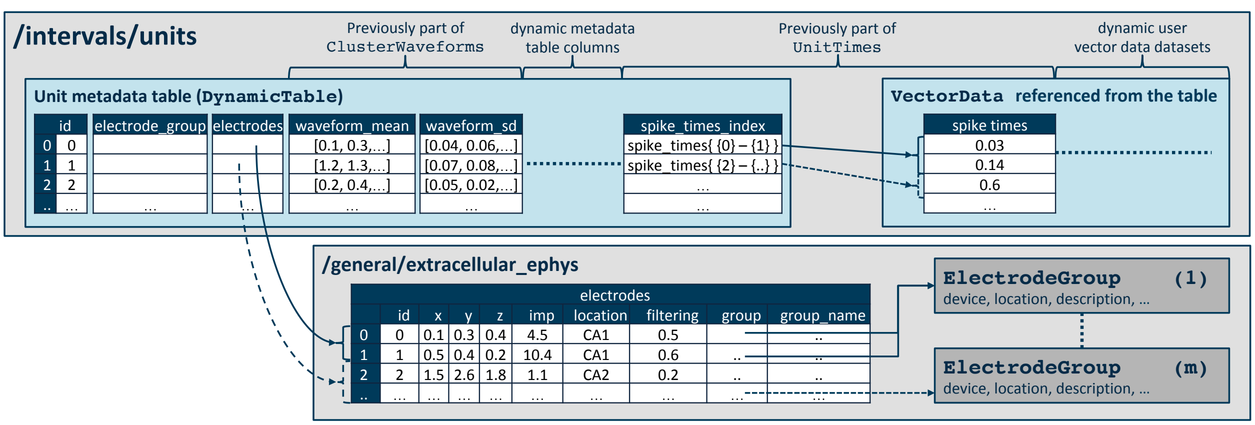


Fig. 10.2: Overview of the data structure for storing spiking unit data and metadata in NWB 2.0.

In addition to `spike_times`, the units table includes the following optional columns:

- `obs_intervals`: intervals indicating the time intervals over which this unit was recorded.
- `electrodes`: range references to the electrodes table indicating which electrodes from which this unit was recorded.

- `electrode_group`: may be used instead of `electrodes` if mutually exclusive electrode groups are sufficient.
- `waveform_mean`: mean waveform across all recorded spikes.
- `waveform_sd`: standard deviation from the mean across all recorded spikes.

10.16.3.7 Improved support for sweep-based information

Changes Added *SweepTable* type stored in `/general/intracellular_ephys`

Reason: In Icephys it is common to have sweeps (i.e., a group of PatchClampSeries belonging together, were up to two TimeSeries are from one electrode, including other TimeSeries not related to an electrode (aka TTL channels)). NWB 1.0.x did not support the concept of sweeps, so it was not possible to link different TimeSeries for sweeps. The goal of this change is to allow users to find the TimeSeries which are from one sweep without having to iterate over all present TimeSeries.

Format Changes Added neurodata_type *SweepTable* to `/general/intracellular_ephys`. SweepTable is a *DynamicTable <sec-DynamicTable>* storing for each sweep a the `sweep_number` and the `series_index`. The later is a VectorIndex pointing to a VectorData dataset describing belonging *PatchClampSeries* to the sweeps. See I499 (PyNWB) and PR701 (PyNWB) for further details.

10.16.4 Improved specification of reference time stamp(s)

To improve the specification of reference time, NWB adopts ISO8061 for storing datetimes and adds `timestamps_reference_time` as explicit zero for all timestamps in addition to the `session_start_time`.

10.16.4.1 Improve standardization of reference time specification using ISO8061

Changes: Modify `session_start_time` an `file_create_date` to enforce use of ISO 8601 datetime strings

Reason: Standardize the specification of timestamps to ensure consistent programmatic and human interpretation

Format Changes: Updated `session_start_time` and `file_create_date` to use `dtype: isodatetime` that was added as dedicated dtype to the specification language. For details see PR641 (PyNWB) and I50 (nwb-schema).

10.16.4.2 Improved specification of reference time

Change: Add field `timestamps_reference_time`, allowing users to explicitly specify a date and time corresponding to time zero for all timestamps in the nwb file.

Reason: Previously `session_start_time` served both as the indicator for the start time of a session as well as the global reference time for a file. Decoupling the two makes the global reference time explicit and enables users to use times relative to the session start as well as other reference time frames, e.g., using POSIX time. This also makes the specification easier to develop against, since this will explicitly specify the offset to obtain relative timestamps, eliminating the need for APIs to guess based on range.

Format Changes: Added top-level field `timestamps_reference_time`. See PR709 (PyNWB) and I49 (nwb-schema) for further details.

10.16.5 Improved storage of time intervals

10.16.5.1 Improved storage of epoch data

Change: Store epoch data as a table to improve efficiency, usability and extensibility.

Reason: In NWB 1.x Epochs are stored as a single group per Epoch. Within each Epoch, the index into each TimeSeries that the Epoch applies to was stored as a single group. This structure is inefficient for storing large numbers of Epochs.

Format Changes: In NWB 2 epochs are stored via a *TimeIntervals* table (i.e., a DynamicTable for storing time intervals) that is stored in the group `/intervals/epochs`. Over the course of the development of NWB 2 the epoch storage has been refined in several phases:

- First, we create a new neurodata_type `Epochs` which was included in *NWBFile* as the group `epochs`. This simplified the extension of the epochs structure. `/epochs` at that point contained a compound (row-based) table with neurodata_type `EpochTable` that described the start/stop times, tags, and a region reference into the `TimeSeriesIndex` to identify the timeseries parts the epoch applies to. Note, the types `Epochs`, `EpochTable` and `TimeSeriesIndex` have been removed/superseded in subsequent changes. (See PR396 (PyNWB) and I119 (nwb-schema)).
- Later, an additional DynamicTable for storing dynamic metadata about epochs was then added to the `Epochs` neurodata_type to support storage of dynamic metadata about epochs without requiring users to create custom extensions (see PR536 (PyNWB)).
- Subsequently the epoch table was then fully converted to a DynamicTable (see PR682 (PyNWB) and I664 (PyNWB))
- Finally, the EpochTable was then moved to `/intervals/epochs` and the EpochTable type was replaced by the more general type *TimeIntervals*. This also led to removal of the `Epochs` type (see PR690 (PyNWB) and I683 (PyNWB))

10.16.5.2 Improved support for trial-based data

Change: Add dedicated concept for storing trial data.

Reason: Users indicated that it was not easy to store trial data in NWB 1.x.

Format Changes: Added optional group `/intervals/trials/` which is a DynamicTable with `id`, `start_time`, and `stop_time` columns and optional additional user-defined table columns. See PR536 on PyNWB for detailed code changes. See the PyNWB docs for a short tutorial on how to use trials. See *NWBFile Groups: /trials* for an overview of the trial schema. **Note:** Originally trials was added a top-level group trials which was then later moved to `/intervals/trials` as part of the generalization of time interval storage as part of PR690 (PyNWB) .

10.16.5.3 Generalized storage of time interval

Change: Create general type *TimeIntervals* (which is a generalization of the previous EpochTable type) and create top-level group `/intervals` for organizing time interval data.

Reason: Previously all time interval data was stored in either `epochs` or `trials`. To facilitate reuse and extensibility this has been generalized to enable users to create arbitrary types of intervals in addition to the predefined types, i.e., epochs or trials.

Format Changes: See PR690 (PyNWB) and I683 (PyNWB) for details:

- Renamed `EpochTable` type to the more general type *TimeIntervals* to facilitate reuse.
- Created top-level group `/intervals` for organizing time interval data.
  - Moved `/epochs` to `/intervals/epochs` and reused the TimeIntervals type

- Moved /trials to /intervals/trials and reused the TimeIntervals type
- Allow users to add arbitrary TimeIntervals tables to /intervals
- Add optional *TimeIntervals* object named invalid_times in ''/intervals``, which specifies time intervals that contain artifacts. See I224 (nwb-schema) and PR731 (PyNWB) for details.

10.16.6 Replaced Implicit Links/Data-Structures with Explicit Links

Change Replace implicit links with explicit soft-links to the corresponding HDF5 objects where possible, i.e., use explicit HDF5 mechanisms for expressing basic links between data rather than implicit ones that require users/developers to know how to use the specific data. In addition to links, NWB 2 adds support for object- and region references, enabling the creation of datasets (i.e., arrays) that store links to other data objects (groups or datasets) or regions (i.e., subsets) of datasets.

Reason: In several places datasets containing arrays of either 1) strings with object names, 2) strings with paths, or 3) integer indexes are used that implicitly point to other locations in the file. These forms of implicit links are not self-describing (e.g., the kind of linking, target location, implicit size and numbering assumptions are not easily identified). This hinders human interpretation of the data as well as programmatic resolution of these kind of links.

Format Changes:

- Text dataset image_plane of <TwoPhotonSeries> is now a link to the corresponding <ImagingPlane> (which is stored in /general/optophysiology)
- Text dataset image_plane_name of <ImageSegmentation> is now a link to the corresponding <ImagingPlane> (which is stored in /general/optophysiology). The dataset is also renamed to image_plane for consistency with <TwoPhotonSeries>
- Text dataset electrode_name of <PatchClampSeries> is now a link to the corresponding <IntracellularElectrode> (which is stored in /general/intracellular_ephys). The dataset is also renamed to electrode for consistency.
- Text dataset site in <OptogeneticSeries> is now a link to the corresponding <StimulusSite> (which is stored in /general/optogenetics).
- Integer dataset electrode_idx of FeatureExtraction is now a dataset electrodes of type DynamicTableRegion pointing to a region of the ElectrodeTable stored in /general/extracellular_ephys/electrodes.
- Integer array dataset electrode_idx of <ElectricalSeries> is now a dataset electrodes of type DynamicTableRegion pointing to a region of the ElectrodeTable stored in /general/extracellular_ephys/electrodes.
- Text dataset /general/extracellular_ephys/<electrode_group_X>/device is now a link <ElectrodeGroup>/device
- The Epochs , Unit, Trial and other dynamic tables in NWB 2 also support (and use) region and object references to explicitly reference other data (e.g., vector data as part of the unit tables).

10.16.7 Improved consistency, identifiably, and readability

10.16.7.1 Improved identifiably of objects

Change: All groups and datasets are now required to either have a unique `name` or a unique `neurodata_type` defined.

Reason: This greatly simplifies the unique identification of objects with variable names.

Format Changes: Defined missing neurodata_types for a number of objects, e.g.,:

- Group `/general/optophysiology/<imaging_plane_X>` now has the neurodata type `ImagingPlane`
- Group `/general/intracellular_ephys/<electrode_X>` now has the neurodata type `IntracellularElectrode`
- Group `/general/optogenetics/<site_X>` now has the neurodata type `StimulusSite`
- …

To enable identification of the type of objects, the `neurodata_type` is stored in HDF5 files as an attribute on the corresponding object (i.e., group or dataset). Also information about the `namespace` (e.g., the name and version) are stored as attributed to allow unique identification of the specification for storage objects.

10.16.7.2 Simplified extension of subject metadata

Specific Change: Assigned `neurodata_type` to `/general/subject` to enable extension of the subject container directly without having to extend `NWBFile` itself. (see https://github.com/NeurodataWithoutBorders/nwb-schema/issues/120 and https://github.com/NeurodataWithoutBorders/nwb-schema/pull/121 for details)

10.16.7.3 Reduce requirement for potentially empty groups

Change: Make several previously required fields optional

Reason: Reduce need for empty groups.

Format Changes: The following groups/datasets have been made optional:

- `/epochs` : not all experiments may require epochs.
- `/general/optogenetics` : not all experiments may use optogenetic data
- `device` in *IntracellularElectrode*
-

10.16.7.4 Added missing metadata

Change: Add a few missing metadata attributes/datasets.

Reason: Ease data interpretation, improve format consistency, and enable storage of additional metadata

Format Changes:

- `/general/devices` text dataset becomes group with neurodata type `Device` to enable storage of more complex and structured metadata about devices (rather than just a single string)
- Added attribute `unit=Seconds` to `<EventDetection>/times` dataset to explicitly describe time units and improve human and programmatic data interpretation
- Added `filtering` dataset to type `<IntracellularElectrode>` (i.e., `/general/intracellular_ephys/<electrode_X>`) to enable specification of per-electrode filtering data

- Added default values for <TimeSeries>/description and <TimeSeries>/comments

10.16.7.5 Improved Consistency

Change: Rename objects, add missing objects, and refine types

Reason: Improve consistency in the naming of data objects that store similar types of information in different places and ensure that the same kind of information is available.

Format Changes:

- Added missing `help` attribute for <BehavioralTimeSeries> to improve consistency with other types as well as human data interpretation
- Renamed dataset `image_plan_name` in <ImageSegmentation> to `image_plane` to ensure consistency in naming with <TwoPhotonSeries>
- Renamed dataset `electrode_name` in <PatchClampSeries> to `electrode` for consistency (and since the dataset is now a link, rather than a text name).
- Renamed dataset `electrode_idx` in <FeatureExtraction> to `electrode_group` for consistency (and since the dataset is now a link to the <ElectrodeGroup>)
- Renamed dataset `electrode_idx` in <ElectricalSeries> to `electrode_group` for consistency (and since the dataset is now a link to the <ElectrodeGroup>)
- Changed `imaging_rate` field in *ImagingPlane* from text to float. See PR697 (PyNWB) and I136 (nwb-schema) for details

10.16.7.6 Added `keywords` field

Change: Added keywords fields to `/general`

Reason: Data archive and search tools often rely on user-defined keywords to facilitate discovery. This enables users to specify keywords for a file. (see PR620 (PyNWB))

10.16.7.7 Removed `source` field

Change: Remove required attribute `source` from all neurodata_types

Reason: In NWB 1.0.x the attribute `source` was defined as a free text entry intended for storage of provenance information. In practice, however, this attribute was often either ignored, contained no useful information, and/or was misused to encode custom metadata (that should have been defined via extensions).

Specific Change: Removed attribute `source` from the core base neurodata_types which effects a large number of the types throughout the NWB schema. For further details see PR695 (PyNWB))

10.16.7.8 Removed `ancestry` field

Change: Removed the explicit specification of ancestry as an attribute as part of the format specification

Reason: 1) avoid redundant information as the ancestry is encoded in the inheritance of types, 2) ease maintenance, and 3) avoid possible inconsistencies between the ancestry attribute and the true ancestry (i.e., inheritance hierarchy) as defined by the spec.

Note The new specification API as part of PyNWB/HDMF makes the ancestry still easily accessible to users. As the ancestry can be easily extracted from the spec, we currently do not write a separate ancestry attribute but this could be easily added if needed. (see also PR707 (PyNWB), I24 (nwb-schema))

10.16.8 Improved organization of processed and acquisition data

10.16.8.1 Improved organization of processed data

Change: Relaxed requirements and renamed and refined core types used for storage of processed data.

Reason: Ease user intuition and provide greater flexibility for users.

Specific Changes: The following changes have been made to the organization of processed data:

- ***Module* has been renamed to *ProcessingModule* to avoid possible confusion**
  and to clarify its purpose. Also *ProcessingModule* may now contain any *NWBDataInterface*.
- With *NWBDataInterface* now being a general base class of *TimeSeries*, this means that it is now possible to define data processing types that directly inherit from *TimeSeries*, which was not possible in NWB 1.x.
- *Interface* has been renamed to *NWBDataInterface* to avoid confusion and ease intuition (see above)
- All *Interface* types in the original format had fixed names. The fixed names have been replaced by specification of corresponding default names. This change enables storage of multiple instances of the same analysis type in the same *ProcessingModule* by allowing users to customize the name of the data processing types, whereas in version 1.0.x only a single instance of each analysis could be stored in a *ProcessingModule* due to the requirement for fixed names.

10.16.8.2 Simplified organization of acquisition data

Specific Changes:

- `/acquisition` may now store any primary data defined via an *NWBDataInterface* type (not just TimeSeries).
- `/acquisition/timeseries` and `/acquisition/images` have been removed
- Created a new neurodata_type *Images* for storing a collection of images to replace `acquisition/images` and provide a more general container for use elsewhere in NWB (i.e., this is not meant to replace *ImageSeries*)

10.16.9 Other changes:

- **PR765 made the timestamps in**
  SpikeEventSeries required

10.16.10 Improved governance and accessibility

Change: Updated release and documentation mechanisms for the NWB format specification

Reason: Improve governance, ease-of-use, extensibility, and accessibility of the NWB format specification

Specific Changes

- The NWB format specification is now released in separate Git repository
- Format specifications are released as YAML files (rather than via Python .py file included in the API)
- Organized core types into a set of smaller YAML files to ease overview and maintenance
- Converted all documentation documents to Sphinx reStructuredText to improve portability, maintainability, deployment, and public access
- Sphinx documentation for the format are auto-generated from the YAML sources to ensure consistency between the specification and documentation
- The PyNWB API now provides dedicated data structured to interact with NWB specifications, enabling users to programmatically access and generate format specifications

10.16.11 Specification language changes

Change: Numerous changes have been made to the specification language itself in NWB 2.0. Most changes to the specification language effect mainly how the format is specified, rather than the actual structure of the format. The changes that have implications on the format itself are described next. For an overview and discussion of the changes to the specification language see specification language release notes.

10.16.11.1 Specification of dataset dimensions

Change: Updated the specification of the dimensions of dataset

Reason: To simplify the specification of dimension of datasets and attribute

Format Changes:

- The shape of various dataset is now specified explicitly for several datasets via the new `shape` key
- The `unit` for values in a dataset are specified via an attribute on the dataset itself rather than via `unit` definitions in structs that are available only in the specification itself but not the format.
- In some cases the length of a dimension was implicitly described by the length of structs describing the components of a dimension. This information is now explicitly described in the `shape` of a dataset.

10.16.11.2 Added `Link` type

Change Added new type for links

Reason:

- Links are usually a different type than datasets on the storage backend (e.g., HDF5)
- Make links more readily identifiable
- Avoid special type specification in datasets

Format Changes: The format itself is not affected by this change aside from the fact that datasets that were links are now explicitly declared as links.

10.16.11.3 Removed datasets defined via autogen

Change Support for `autogen` has been removed from the specification language. After review of all datasets that were produced via autogen it was decided that all autogen datasets should be removed from the format.

Reason The main reasons for removal of autogen dataset is to ease use and maintenance of NWB files by 1) avoiding redundant storage of information (i.e., improve normalization of data) and 2) avoiding dependencies between data (i.e., datasets having to be updated due to changes in other locations in a file).

Format Changes

- Datasets/Attributes that have been removed due to redundant storage of the path of links stored in the same group:
  - IndexSeries/indexed_timeseries_path
  - RoiResponseSeries/segmentation_interface_path
  - ImageMaskSeries/masked_imageseries_path
  - ClusterWaveforms/clustering_interface_path
  - EventDetection/source_electricalseries_path
  - MotionCorrection/image_stack_name/original_path
  - NWBFile/epochs/epoch_X.links
- Datasets//Attributes that have been removed because they stored only a list of groups/datasets (of a given type or property) in the current group.
  - Module.interfaces (now ProcessingModule)
  - ImageSegmentation/image_plane/roi_list
  - UnitTimes/unit_list
  - TimeSeries.extern_fields
  - TimeSeries.data_link
  - TimeSeries.timestamp_link
  - TimeSeries.missing_fields
- Other datasets/attributes that have been removed to ease use and maintenance because the data stored is redundant and can be easily extracted from the file:
  - NWBFile/epochs.tags
  - TimeSeries/num_samples
  - Clustering/cluster_nums

10.16.11.4 Removed `'neurodata\_type=Custom'`

Change The `'neurodata\_type=Custom'` has been removed.

Reason All additions of data should be governed by extensions. Custom datasets can be identified based on the specification, i.e., any objects that are not part of the specification are custom.

10.17 1.0.x (09/2015 - 04/2017)

NWB 1.0.x has been deprecated. For documents relating to the 1.0.x schema please see https://github.com/NeurodataWithoutBorders/specification_nwbn_1_0_x.

10.17.1 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.

10.17.2 1.0.5i_beta, Dec 6, 2016

Removed some comments. Modify author string in info section.

10.17.3 1.0.5h_beta, Nov 30, 2016

Add dimensions to /acquisition/images/<image_X>

10.17.4 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 was done to make the specification language more consistent by having these group properties specified in one place (`"\_properties"` dictionary).

10.17.5 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`.

10.17.6 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.

10.17.7 1.0.5d_beta, Sept 6, 2016

- Changed ImageSeries img_mask dimensions to: `"dimensions": ["num\_y","num\_x"]` to match description.

10.17.8 1.0.5c_beta, Aug 17, 2016

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

10.17.9 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.

10.17.10 1.0.5a_beta, Aug 10, 2016

Expand class of Ids allowed in `TimeSeries` `missing\_fields` attribute to allow custom uses.

10.17.11 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`

10.17.12 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.

10.17.13 1.0.3 April, 2016

- Renamed "ISI_Retinotopy" to "ISIRetinotopy"
- Change ImageSeries external_file to an array. Added attribute starting_frame.
- Added IZeroClampSeries.

10.17.14 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

10.17.15 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/

```

(continues on next page)

(continued from previous page)

```
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.

10.17.16 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.

CHAPTER

ELEVEN

CREDITS

11.1 Acknowledgments

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

11.2 Authors

11.2.1 NWB: Version 2.0.0 and later

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

11.2.2 NWB: Version 1.0.x and earlier

Version 1.0.5g (and earlier) of the NWB file format were created by Jeff Teeters et al. as part of the first NWB pilot project. The documents for NWB 2 have been adopted from the final version of format docs released by the original NWB pilot project.

CHAPTER TWELVE

LEGAL

12.1 Copyright

If you have questions about your rights to use or distribute this software, please contact Berkeley Lab's Innovation & Partnerships Office at IPO@lbl.gov.

NOTICE. This Software was developed under funding from the U.S. Department of Energy and the U.S. Government consequently retains certain rights. As such, the U.S. Government has been granted for itself and others acting on its behalf a paid-up, nonexclusive, irrevocable, worldwide license in the Software to reproduce, distribute copies to the public, prepare derivative works, and perform publicly and display publicly, and to permit other to do so.

12.2 Licence

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

(1) Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

(2) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

(3) Neither the name of the University of California, Lawrence Berkeley National Laboratory, U.S. Dept. of Energy nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.