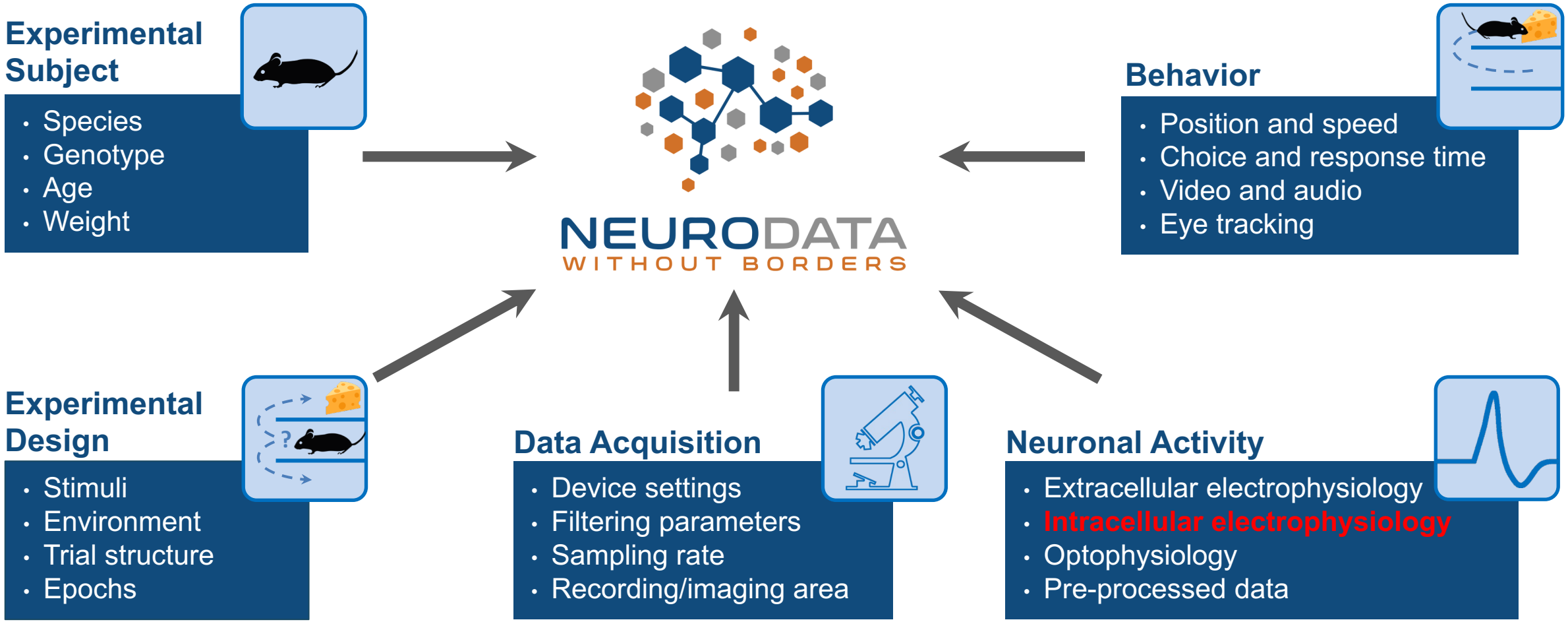
Intracellular Electrophysiology Data in NWB

Oliver Rübel

Machine Learning & Analytics Group
Computational Biosciences Group
Scientific Data Division
Lawrence Berkeley National Laboratory

BERKELEY LAB
Bringing Science Solutions to the World

NWB for neurophysiology data and metadata



Overview

1. Intracellular Electrophysiology Data
2. Intracellular Electrophysiology Experiment Metadata
3. Creating Intracellular Electrophysiology NWB Files Using PyNWB and MatNWB
4. Querying Intracellular Electrophysiology Experiment Metadata Using PyNWB and pandas
5. The Role of Neurodata Extension (NDX) in Advancing Neurophysiology Data Standardization

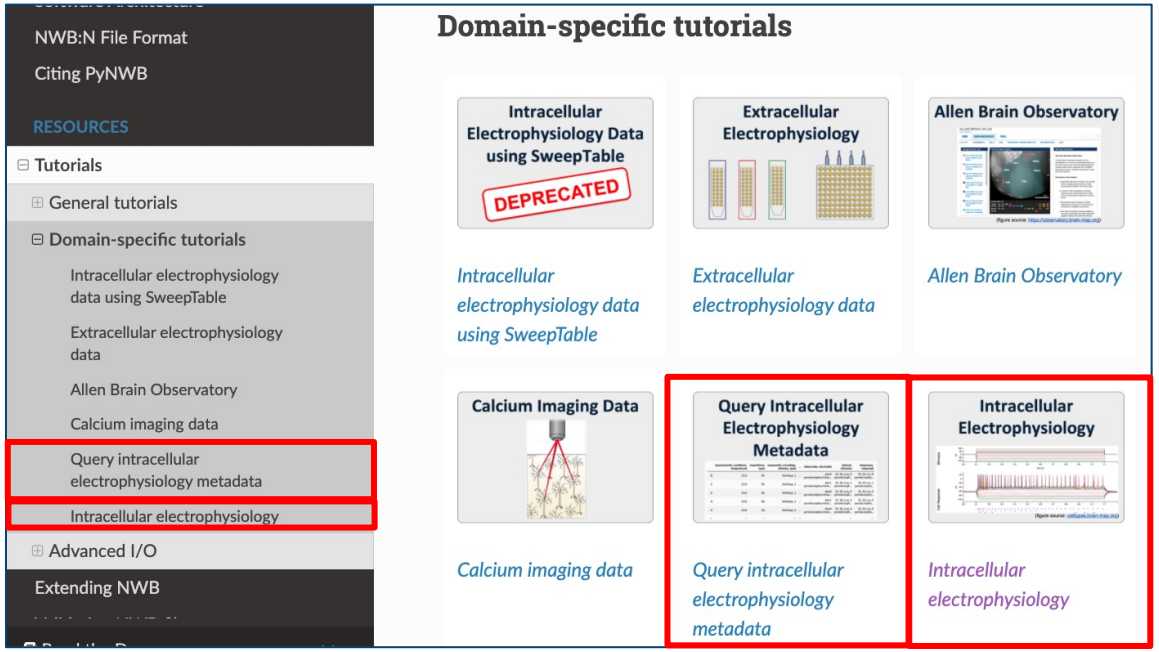
Python Resources

- **Install PyNWB**

```
$ pip install -U pynwb
```

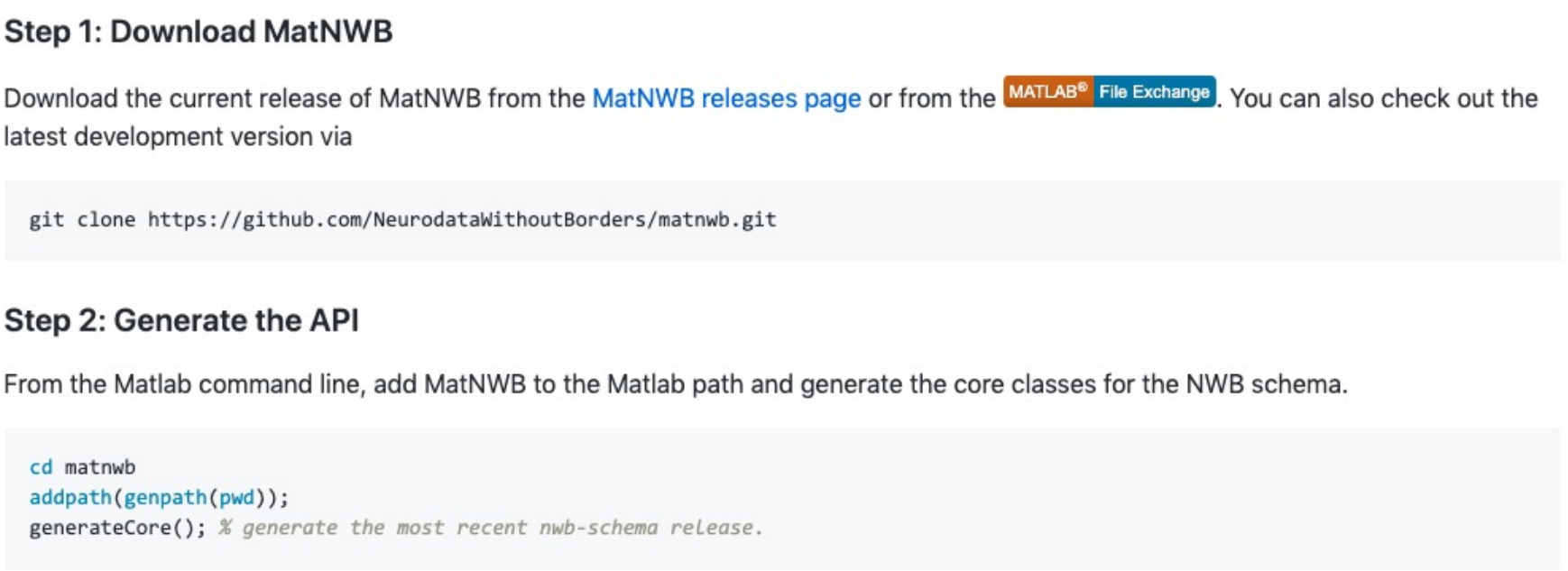
```
$ conda install -c conda-forge pynwb
```

- **PyNWB Tutorials:** https://pynwb.readthedocs.io/en/stable/tutorials



MATLAB Resources

- **Install MatNWB:** https://neurodatawithoutborders.github.io/matnwb



Step 1: Download MatNWB

Download the current release of MatNWB from the MatNWB releases page or from the MATLAB File Exchange. You can also check out the latest development version via

```
git clone https://github.com/NeurodataWithoutBorders/matnwb.git
```

Step 2: Generate the API

From the Matlab command line, add MatNWB to the Matlab path and generate the core classes for the NWB schema.

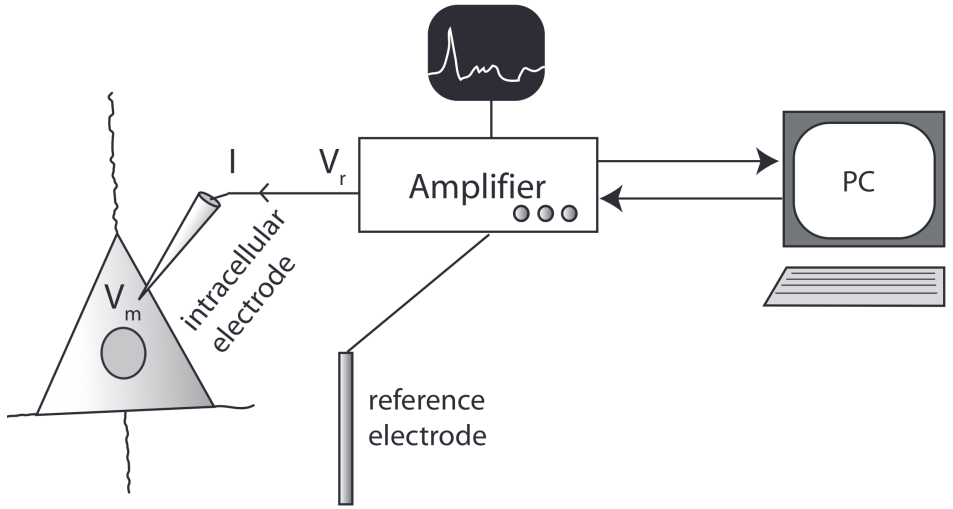
```
cd matnwb
addpath(genpath(pwd));
generateCore(); % generate the most recent nwb-schema release.
```

- **MatNWB Tutorial:**
  - `matnwb/tutorials/icephys.mlx`
  - https://neurodatawithoutborders.github.io/matnwb/tutorials/html/icephys.html

Intracellular Electrophysiology Data

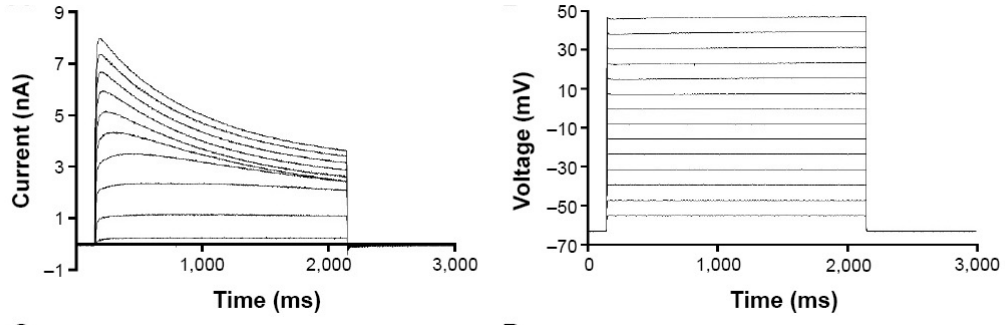
Intracellular Recording

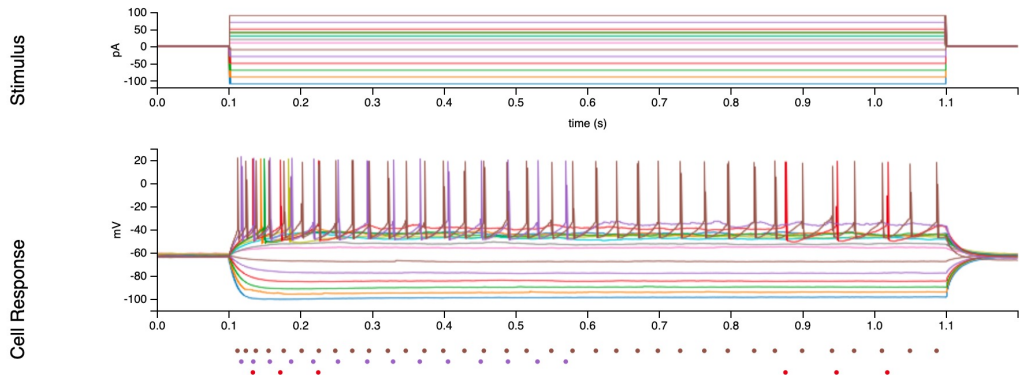
- Measurement of voltage or current across the cell membrane
- Allows measurement of subthreshold activity and estimation of membrane properties and conductances



Intracellular Recording

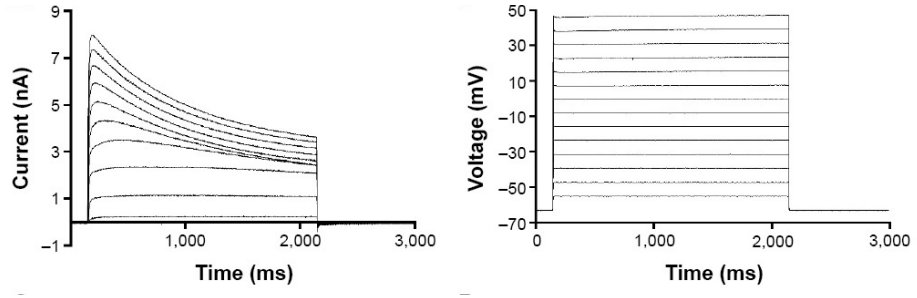
- Measurement of voltage or current across the cell membrane
- Allows measurement of subthreshold activity and estimation of membrane properties and conductances
- Two modes:
  - Voltage Clamp
  - Current Clamp

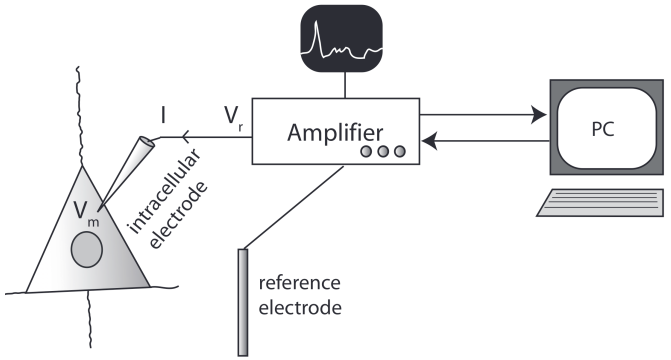


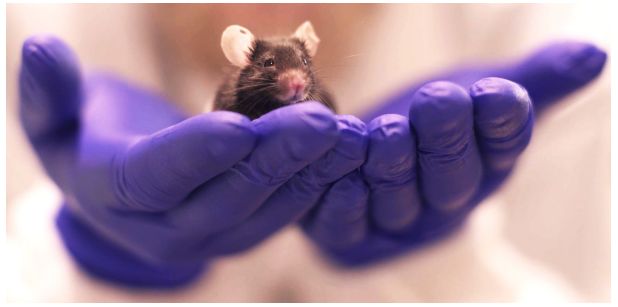


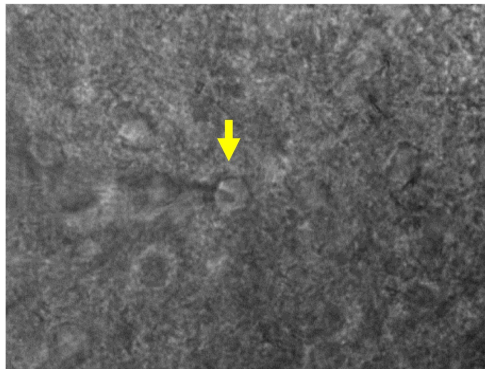
Data and Metadata for Intracellular Recording

- TimeSeries to store Stimulus and Response traces
- Device metadata
  - Electrode
  - Headstage
  - Amplifier
- Preparation metadata
  - Subject
  - Slice

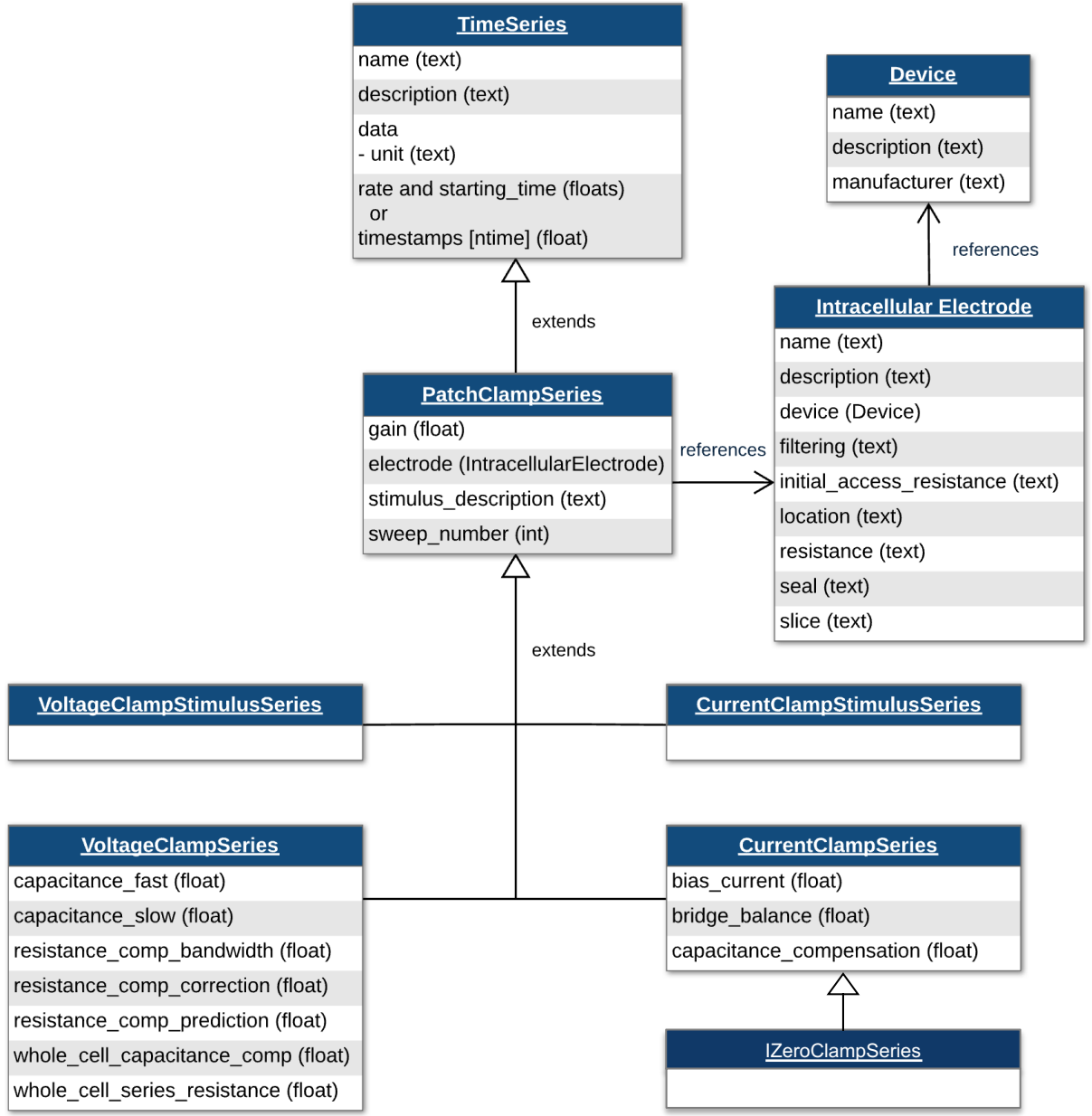






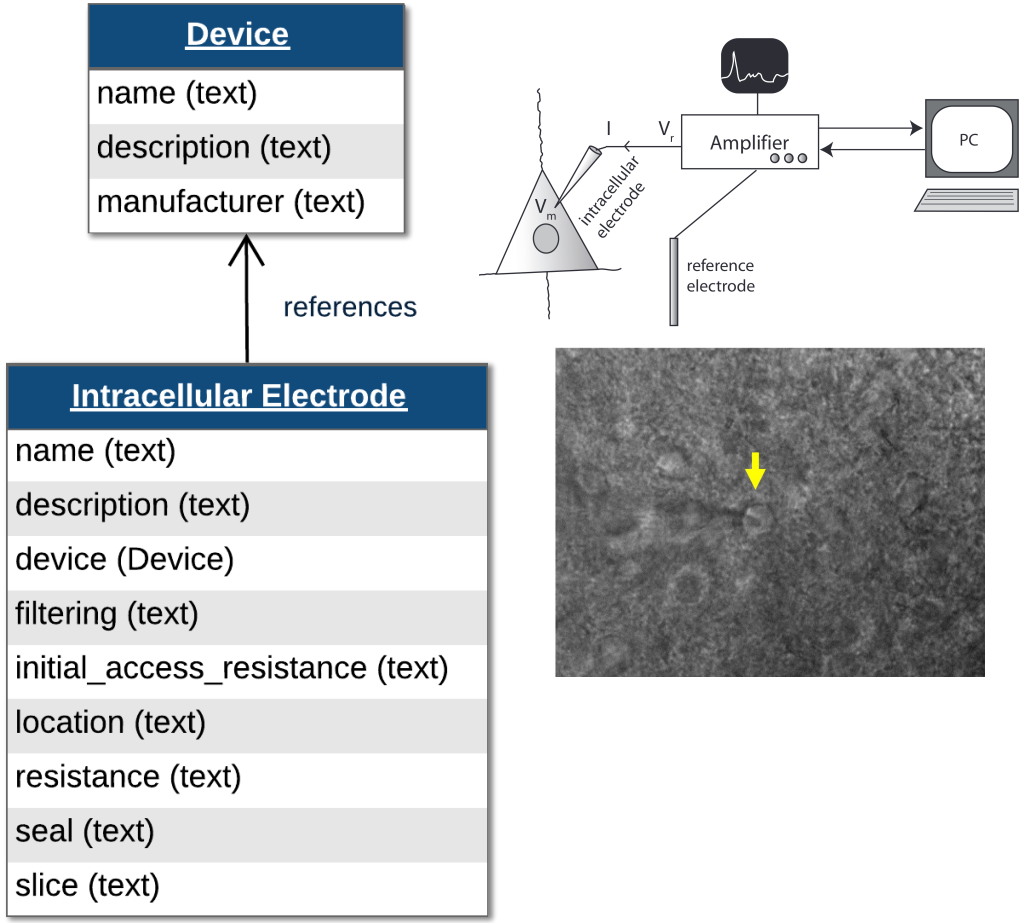


Intracellular Electrophysiology Class Diagram



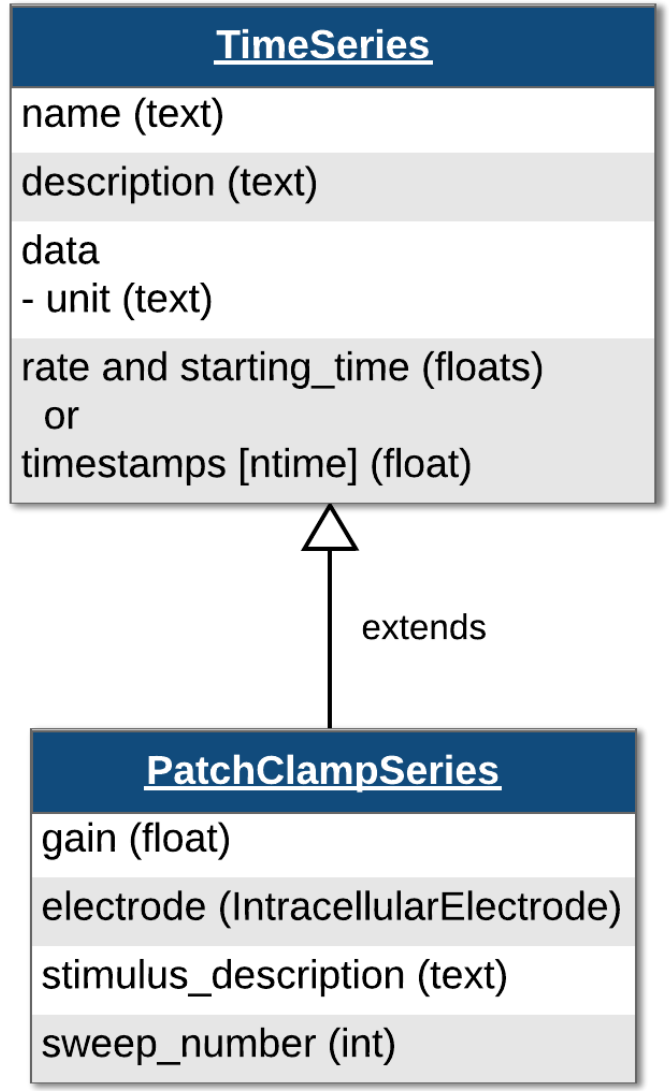
IntracellularElectrode

description	Description of electrode (e.g., whole-cell, sharp, etc.)
device	Device that was used to record from this electrode, e.g. amplifier, recording system. (required)
filtering	Electrode specific filtering.
initial_access_resistance	Initial access resistance (expected units: ohms)
location	Specify the area/layer, stereotaxic coordinates, anatomical atlas references, etc.
resistance	Electrode resistance (expected units: ohms).
seal	Seal resistance.
slice	Information about the slice preparation.



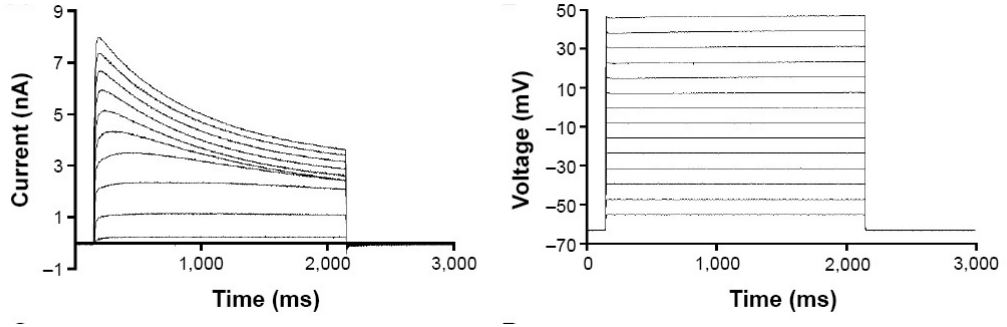
PatchClampSeries

electrode	Link to Intracellular Electrode with electrode metadata. (required)
stimulus_description	Protocol/stimulus name for this recording.
gain	Gain of the recording.
sweep_number	Sweep number, allows grouping of multiple PatchClampSeries together.



VoltageClampSeries

Current data from an intracellular voltage-clamp recording. A corresponding *VoltageClampStimulusSeries* (stored separately) is used to store the command voltage trace used to generate this response.

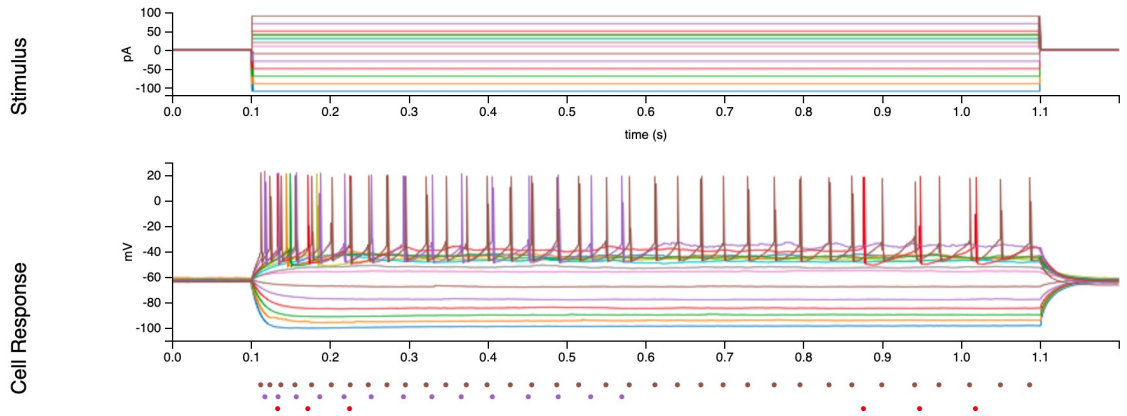


capacitance_fast	Fast capacitance (farads)
capacitance_slow	Slow capacitance (farads)
resistance_comp_bandwidth	Resistance compensation bandwidth (Hz)
resistance_comp_correction	Resistance compensation correction (%)
resistance_comp_prediction	Resistance compensation prediction (%)
whole_cell_capacitance_comp	Whole cell capacitance compensation (farads)
whole_cell_series_resistance	Whole cell series resistance compensation (ohms)

VoltageClampSeries
capacitance_fast (float)
capacitance_slow (float)
resistance_comp_bandwidth (float)
resistance_comp_correction (float)
resistance_comp_prediction (float)
whole_cell_capacitance_comp (float)
whole_cell_series_resistance (float)

CurrentClampSeries

Voltage data from an intracellular current-clamp recording. A corresponding *CurrentClampStimulusSeries* (stored separately) is used to store the injected current trace used to generate this response.



bias_current	Bias current (amperes).
bridge_balance	Bridge balance (ohms).
capacitance_compensation	Capacitance compensation (farads).

CurrentClampSeries
bias_current (float)
bridge_balance (float)
capacitance_compensation (float)

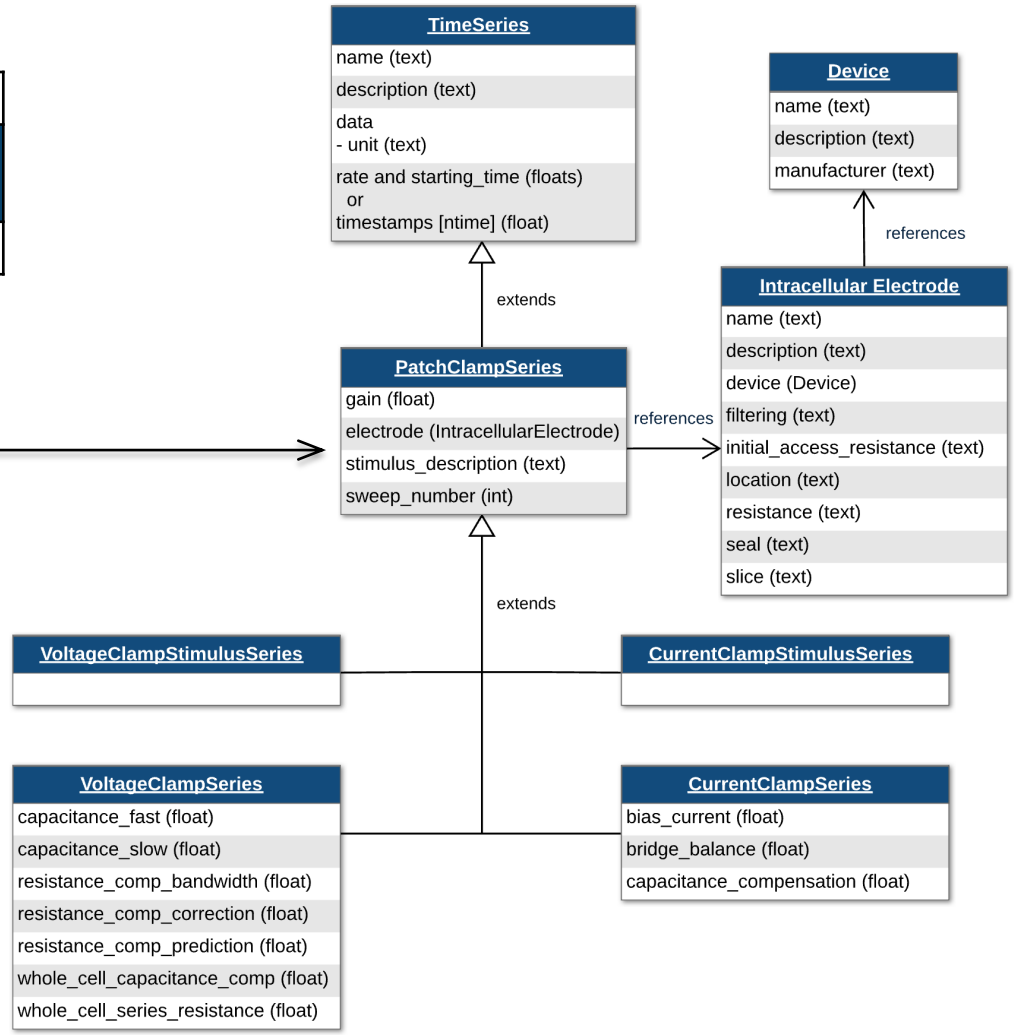
IZeroClampSeries is a subtype of CurrentClampSeries used for voltage data from intracellular recordings when all current and amplifier settings are off and no stimulus is present. I.e., the bias_current, bridge_balance, and capacitance_compensation fields are set to zero and there is no corresponding CurrentClampStimulusSeries.

Relating simultaneously recorded PatchClampSeries

NWB v(2 – 2.3): SweepTable

SweepTable			
id	sweep_number uint32	series PatchClampSeries	…
…	…	…	…

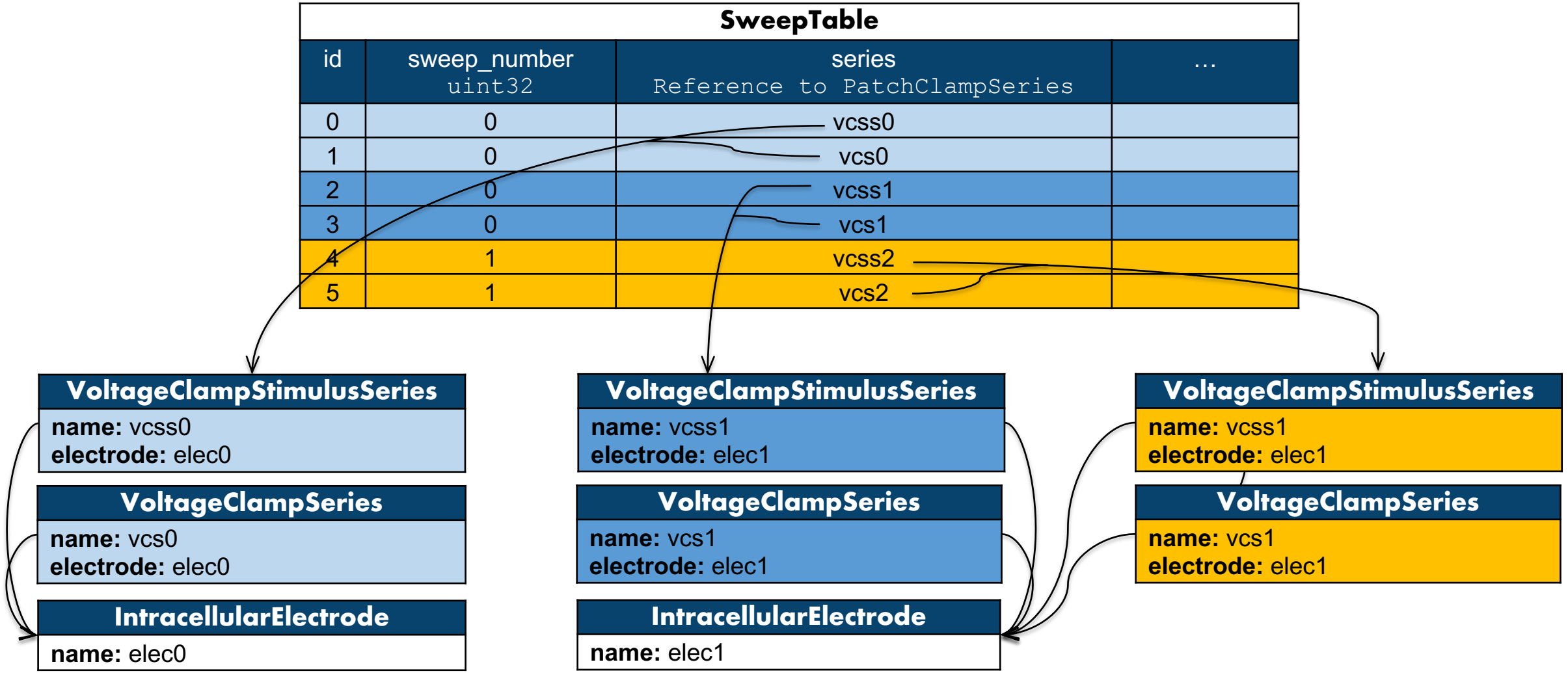
references



SweepTable allows grouping **PatchClampSeries** that stem from the same *sweep*, i.e., a group of simultaneously recorded PatchClampSeries (with the same starting point in time).

Example: Relating PatchClampSeries via the SweepTable

NWB v(2 – 2.3): SweepTable

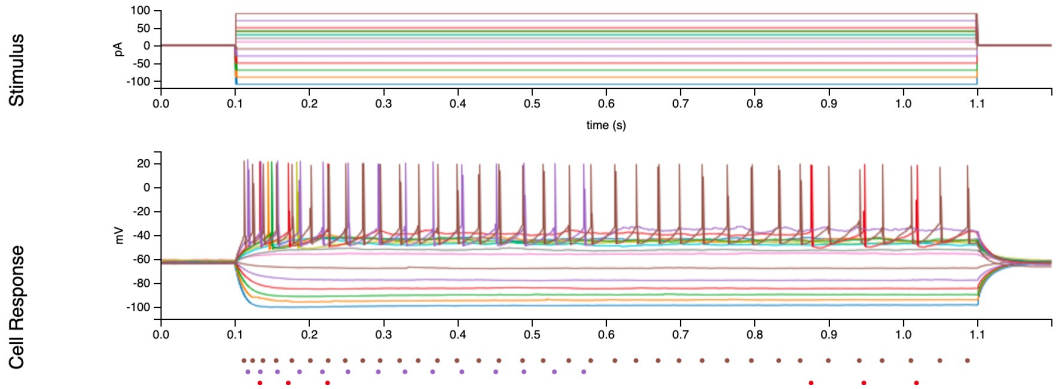


Intracellular Electrophysiology Experiment Metadata

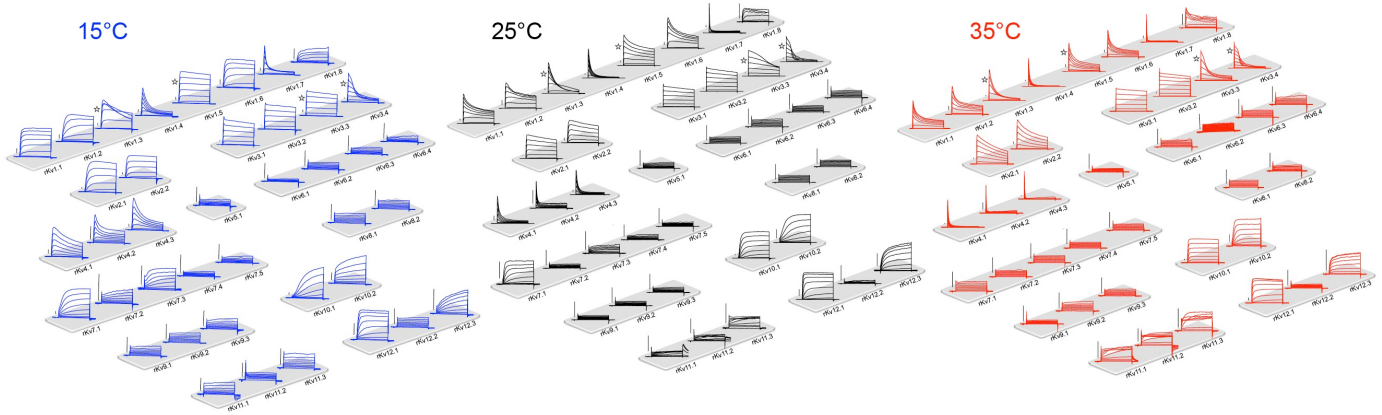
Motivation

How can we describe the complex organization of intracellular experiments?

- Intracellular electrophysiology experiments typically consist of large collections of recordings with varying parameters
- Experiments are typically organized hierarchically by time



Example set of sequentially recorded stimuli with varying amplitude and corresponding response.
(source: celltypes.brain-map.org)



Example set of activation stimuli R. Ranjan et al.
(source: doi: 10.3389/fncel.2019.00358, channelpedia.epfl.ch)

Levels in the intracellular recording hierarchy

1. **Intracellular Recordings** each consisting of:
    - **Intracellular Electrode,** i.e., an **IntracellularElectrode** as defined in NWB
    - **Stimulus**, i.e., a **TimeSeries** representing voltage or current stimulation with a particular set of parameters
    - **Response,** i.e., a **TimeSeries** representing voltage or current recorded from a single cell using a single intracellular electrode
2. **Simultaneous Recordings** (a.k.a. sweep) each consisting of one or more *intracellular recordings*. I.e., a group of stimuli presented and responses recorded simultaneously, possibly using multiple electrodes
3. **Sequential Recordings,** each consisting of one or more *simultaneous recordings*. This typically represents a set of stimuli that are applied in sequence, often of the same stimulus type.
4. **Repetitions,** each consisting of one or more *sequential recordings.* This typically represents sets of stimuli that are applied in sequence.
5. **Experimental Conditions**, each consisting of one ore more *repetitions*. This represents a collection of recordings that were acquired in sequence under the same experimental conditions.

Goals

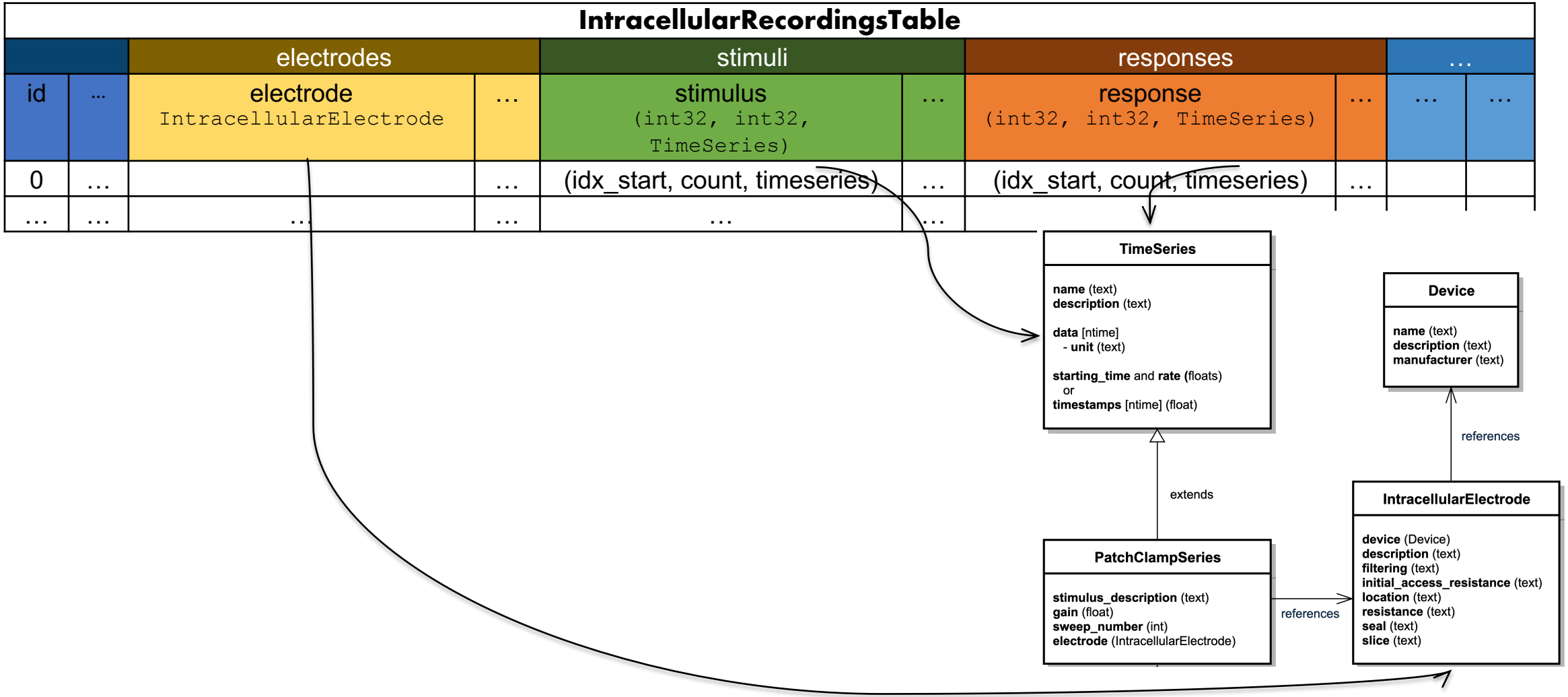
- Allow metadata to be associated explicitly with different levels of the hierarchy
- Relationships between the different levels should be explicit
- Relationships between different TimeSeries should be explicit
- Data should not be replicated in an NWB file to reduce the risk of errors
- The data model should be extensible to allow further standardization of icephys experiment metadata
- The data model should be backward compatible with NWB v[2.0 - 2.3] (using SweepTable, which was deprecated in NWB 2.4)

The Intracellular Electrophysiology Experiment Metadata Data Model

NEURODATA
WITHOUT BORDERS

Intracellular Recordings Table

Overview



Intracellular Recordings Table

Features

IntracellularRecordingsTable									
		electrodes		stimuli		responses		…	
id	…	electrode IntracellularElectrode	…	stimulus (int32, int32, TimeSeries)	…	response (int32, int32, TimeSeries)	…	…	…
0	…		…	(idx_start, count, timeseries)	…	(idx_start, count, timeseries)	…		
…	…	…	…	…	…	…	…		

- Supports dynamic metadata for each type of data (i.e., electrodes, stimuli, and responses) as well as across an intracellular recording
- Allows definition of DynamicTable data categories
- Makes the relationship between stimulus and response pairs explicit in the table (i.e., row index)
- Allows referencing of temporal sub-ranges in **TimeSeries** (similar to **TimeIntervals** in NWB)
- Compatible with current PatchClampSeries types
  - sweep_number is no longer used here (but still required in PatchClampSeries)
  - Electrode redundant in table and PatchClampSeries
  - SweepTable no longer needed

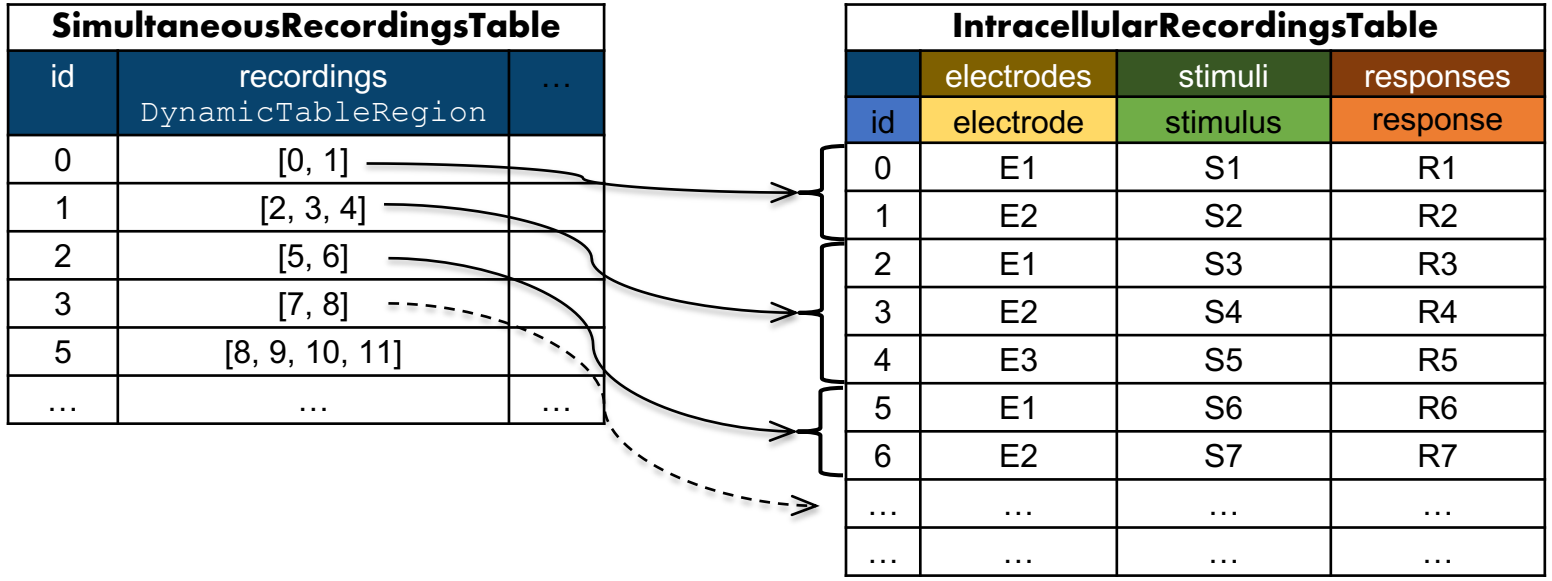
ICEphys Metadata Hierarchy

Intracellular Recordings

IntracellularRecordingsTable			
	electrodes	stimuli	responses
id	electrode	stimulus	response
0	E1	S1	R1
1	E2	S2	R2
2	E1	S3	R3
3	E2	S4	R4
4	E3	S5	R5
5	E1	S6	R6
6	E2	S7	R7
…	…	…	…
…	…	…	…

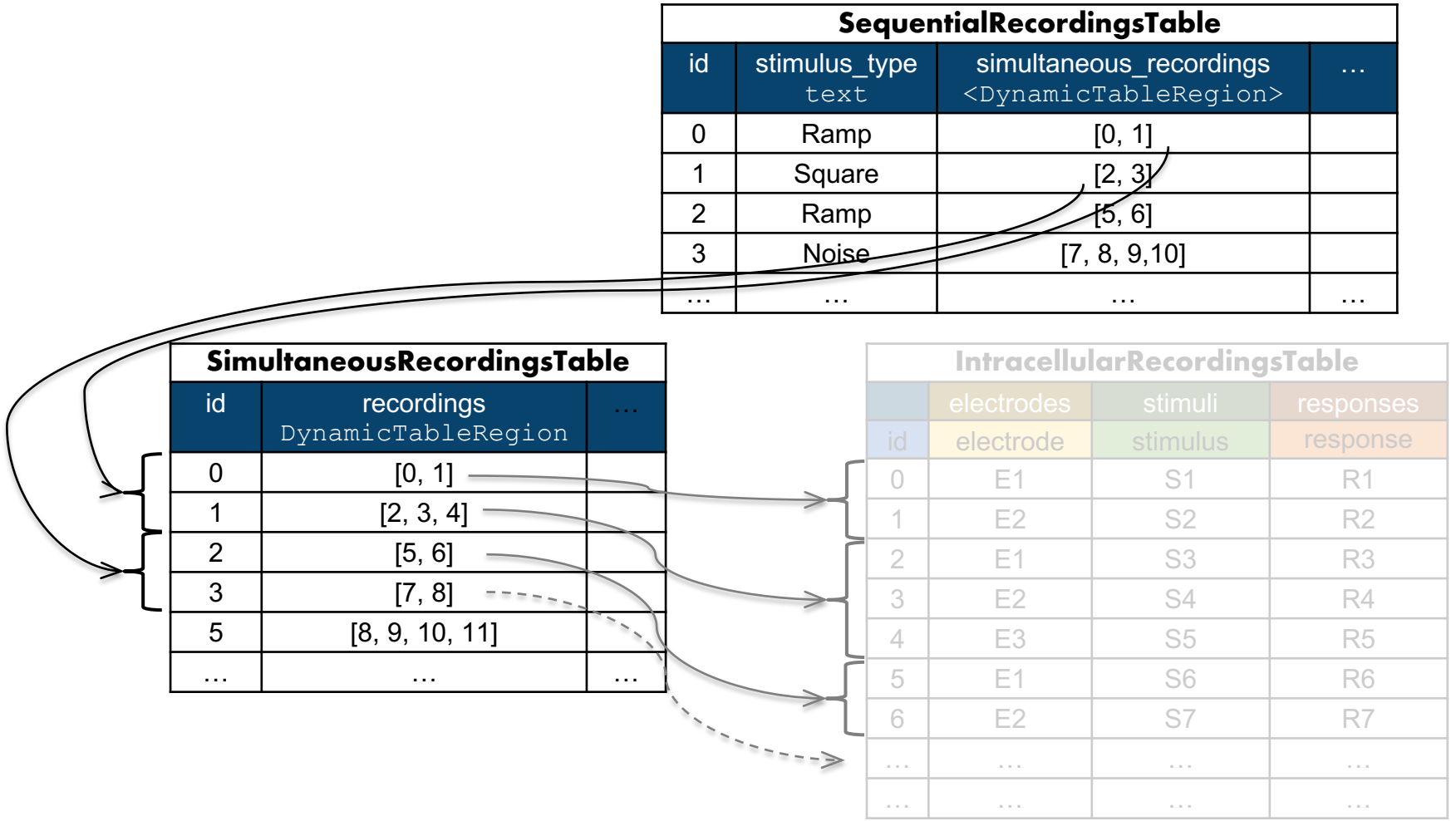
ICEphys Metadata Hierarchy

Simultaneous Recordings



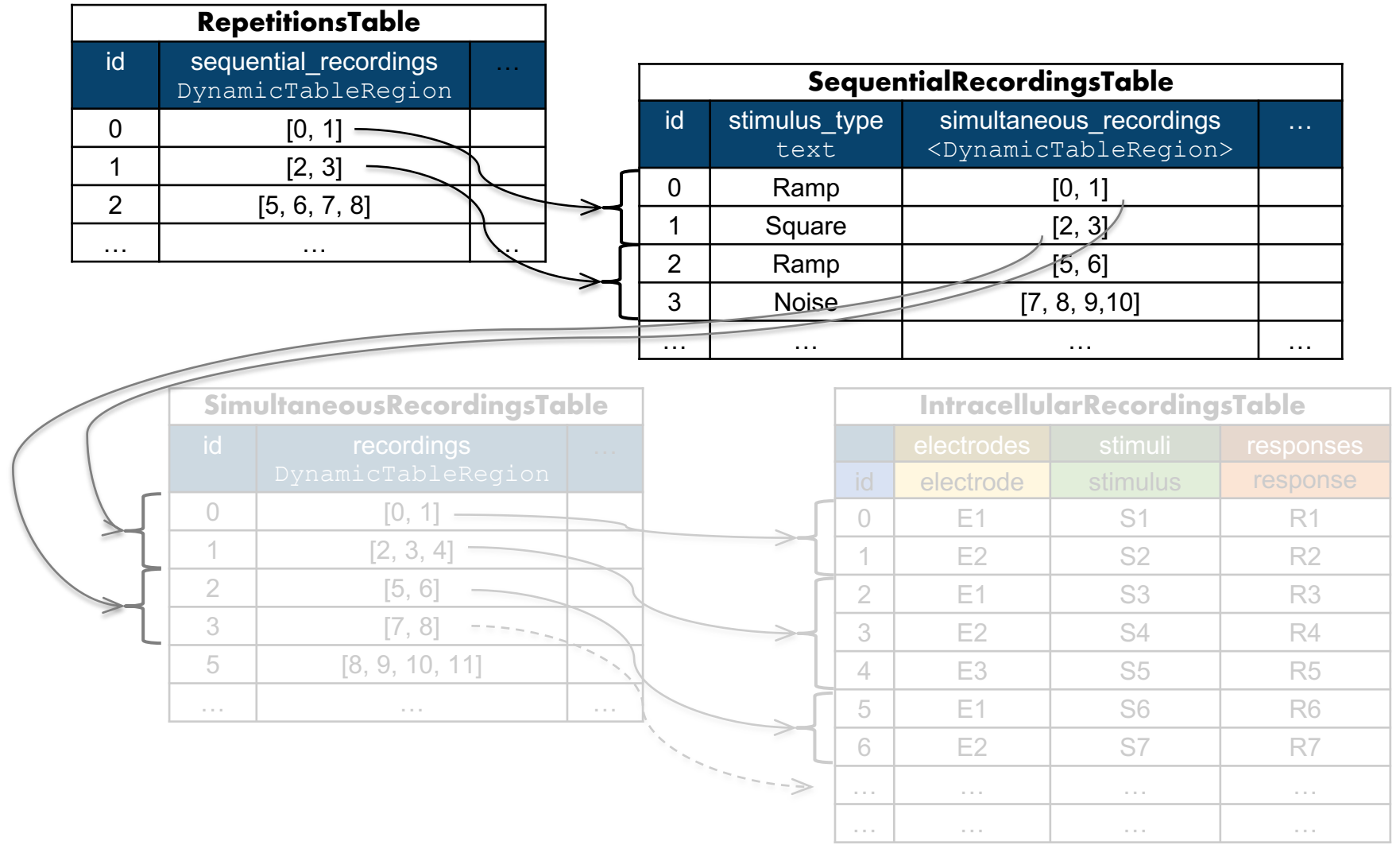
ICEphys Metadata Hierarchy

Sequential Recordings



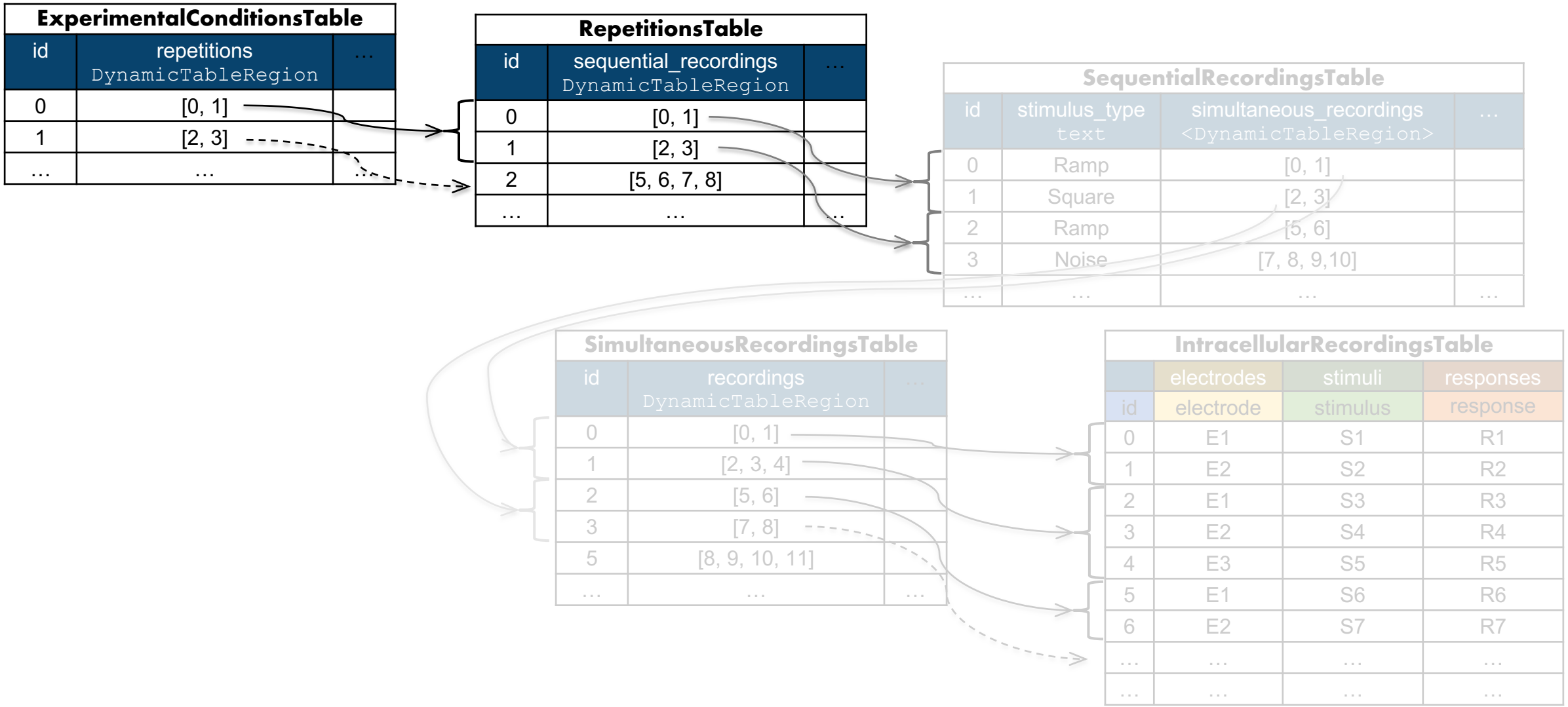
ICEphys Metadata Hierarchy

Repetitions



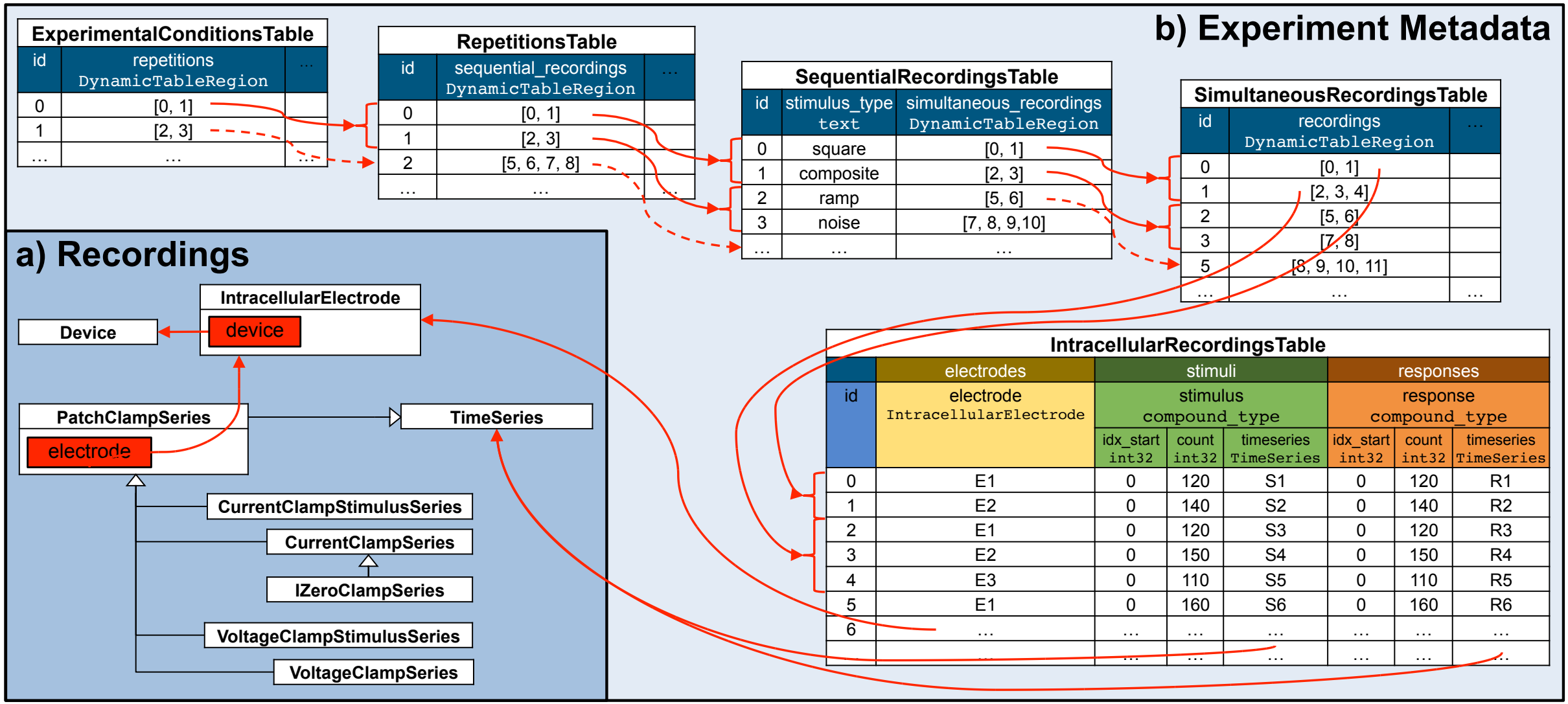
ICEphys Metadata Hierarchy

Experimental Conditions



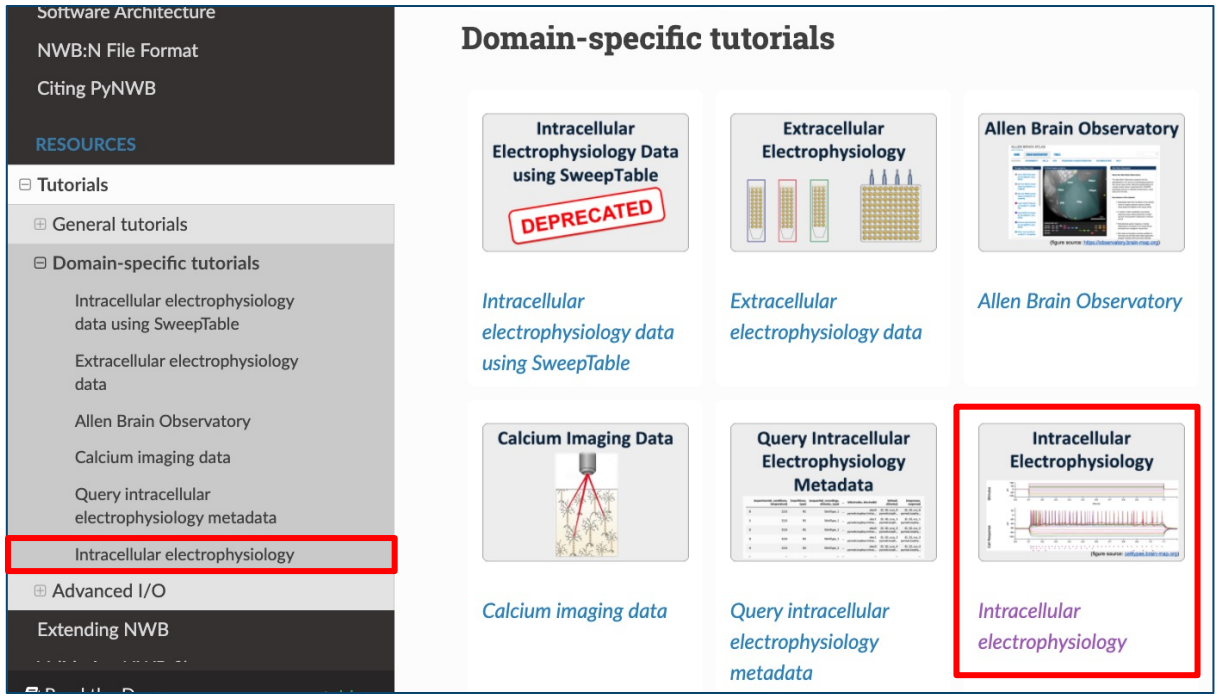
ICEphys Metadata Hierarchy

Overview

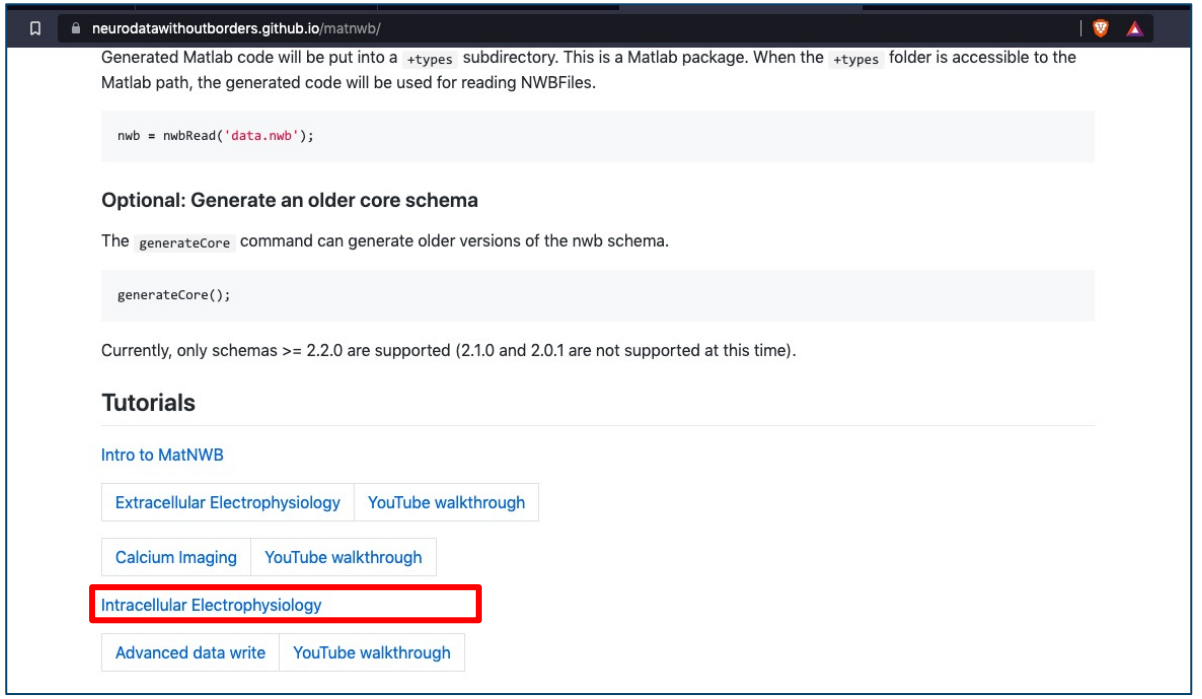


Creating Intracellular Electrophysiology NWB Files Using PyNWB and MatNWB

PyNWB tutorial at :
https://pynwb.readthedocs.io/en/stable/tutorials/domain/plot_icephys.html



MatNWB tutorial at:
https://neurodatawithoutborders.github.io/matnwb/tutorials/html/icephys.html



Step 1: Setup the NWBFile

1.1 Create the **NWBFile**

```
# Create the file
nwbfile = NWBFile(
    session_description='my first synthetic recording',
    identifier='EXAMPLE_ID',
    session_start_time=datetime.now(tzlocal()),
    experimenter='Dr. Bilbo Baggins',
    lab='Bag End Laboratory',
    institution='University of Middle Earth at the Shire',
    experiment_description='I went on an adventure with thirt
    session_id='LONELYMTN'
)
```

1.2 Create the **Device**

```
device = nwbfile.create_device(name='Heka ITC-1600')
```

1.3 Create the **IntracellularElectrode**

```
electrode = nwbfile.create_icephys_electrode(
    name="elec0",
    description='a mock intracellular electrode',
    device=device
)
```

1.4 Create the **PatchClampSeries**

```
# Create an example icephys stimulus.
stimulus = VoltageClampStimulusSeries(
    name="ccss",
    data=[1, 2, 3, 4, 5],
    starting_time=123.6,
    rate=10e3,
    electrode=electrode,
    gain=0.02,
    sweep_number=np.uint64(15)
)

# Create and icephys response
response = VoltageClampSeries(
    name='vcs',
    data=[0.1, 0.2, 0.3, 0.4, 0.5],
    conversion=1e-12,
    resolution=np.nan,
    starting_time=123.6,
    rate=20e3,
    electrode=electrode,
    gain=0.02,
    capacitance_slow=100e-12,
    resistance_comp_correction=70.0,
    sweep_number=np.uint64(15)
)
```

Step 1: Setup the NWBFile

MATLAB®

1.1 Create the **NWBFile**

```
nwbfile = NwbFile( ...
    'session_description', 'my first synthetic recording', ...
    'identifier', 'EXAMPLE_ID', ...
    'session_start_time', session_start_time, ...
    'general_experimenter', 'Dr. Bilbo Baggins', ...
    'general_lab', 'Bag End Laboratory', ...
    'general_institution', 'University of Middle Earth at the Shire', ...
    'general_experiment_description', 'I went on an adventure with thirtee
    'general_session_id', 'LONELYMTN' ...
);
```

1.2 Create the **Device**

```
device = types.core.Device();
nwbfile.general_devices.set('Heka ITC-1600', device);
```

1.3 Create the **IntracellularElectrode**

```
electrode = types.core.IntracellularElectrode( ...
    'description', 'a mock intracellular electrode', ...
    'device', types.untyped.SoftLink(device) ...
);
nwbfile.general_intracellular_ephys.set('elec0', electrode);
```

1.4 Create the **PatchClampSeries**

```
ccss = types.core.VoltageClampStimulusSeries( ...
    'data', [1, 2, 3, 4, 5], ...
    'starting_time', 123.6, ...
    'starting_time_rate', 10e3, ...
    'electrode', types.untyped.SoftLink(electrode), ...
    'gain', 0.02, ...
    'sweep_number', uint64(15), ...
    'stimulus_description', 'N/A' ...
);

nwbfile.stimulus_presentation.set('ccss',  ccss);

vcs = types.core.VoltageClampSeries( ...
    'data', [0.1, 0.2, 0.3, 0.4, 0.5], ...
    'data_conversion', 1e-12, ...
    'data_resolution', NaN, ...
    'starting_time', 123.6, ...
    'starting_time_rate', 20e3, ...
    'electrode', types.untyped.SoftLink(electrode), ...
    'gain', 0.02, ...
    'capacitance_slow', 100e-12, ...
    'resistance_comp_correction', 70.0, ...
    'stimulus_description', 'N/A', ...
    'sweep_number', uint64(15) ...
);
nwbfile.acquisition.set('vcs',  vcs);
```

Step 2: Populate the **IntracellularRecordingsTable**

```
rowindex = nwbfile.add_intracellular_recording(
    electrode=electrode,
    stimulus=stimulus,
    response=response,
    id=10
)
```

automatically adds the **stimulus**, **response** and **electrode** to the **NWBFile** if necessary

id is optional but must be unique

```
rowindex2 = nwbfile.add_intracellular_recording(
    electrode=electrode,
    stimulus=stimulus,
    stimulus_start_index=1,
    stimulus_index_count=3,
    response=response,
    response_start_index=2,
    response_index_count=3,
    id=11
)
```

optionally select ranges in time, e.g., in case that the recorded stimulus and response do not align

A recording may also consist of just an electrode and stimulus or electrode and response

```
rowindex3 = nwbfile.add_intracellular_recording(
    electrode=electrode,
    response=response,
    id=12
)
```

Step 2.1: Add custom intracellular recording metadata

Add a custom column to the main table

```
nwbfile.intracellular_recordings.add_column(
    name='recording_tag',
    data=['A1', 'A2', 'A3'],
    description='String with a recording tag'
)
```

Add a custom column to a category table

```
nwbfile.intracellular_recordings.add_column(
    name='voltage_threshold',
    data=[0.1, 0.12, 0.13],
    description='Just an example column on the
    category='electrodes'
)
```

Add a custom category table

```
# Create a new DynamicTable for our category that contains a location
location_column = VectorData(
    name='location',
    data=['Mordor', 'Gondor', 'Rohan'],
    description='Recording location in Middle Earth'
)

lab_category = DynamicTable(
    name='recording_lab_data',
    description='category table for lab-specific recording metadata',
    colnames=['location', ],
    columns=[location_column, ]
)
# Add the table as a new category to our intracellular_recordings
nwbfile.intracellular_recordings.add_category(category=lab_category)
```

Step 2: Populate the IntracellularRecordingsTable

Create the IntracellularRecrodingsTable

```
ic_rec_table = types.core.IntracellularRecordingsTable( ...
    'categories', {'electrodes', 'stimiuli', 'responses'}, ...
    'colnames', {'recordings_tag'}, ...
    'description', [ ...
        '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.'], ...
    'id', types.hdmf_common.ElementIdentifiers( ...
        'data', int64([10, 11, 12]) ...
    ) ...
);
```

Create the electrodes, stimuli, and responses category tables

```
ic_rec_table.stimuli = types.core.IntracellularStimuliTable( ...
    'description', 'Table for storing intracellular stimulus related metadata.', ...
    'colnames', {'stimulus'}, ...
    'id', types.hdmf_common.ElementIdentifiers( ...
        'data', int64([0, 1, 2]) ...
    ), ...
    'stimulus', types.core.TimeSeriesReferenceVectorData( ...
        'description', 'Column storing the reference to the recorded stimulus for the recording (rows)', ...
        'data', struct( ...
            'idx_start', [0, 1, -1], ...
            'count', [5, 3, -1], ...
            'timeseries', [ ...
                types.untyped.ObjectView(ccss), ...
                types.untyped.ObjectView(ccss), ...
                types.untyped.ObjectView(vcs) ...
            ] ...
        )...
    )...
);
```

```
ic_rec_table.electrodes = types.core.IntracellularElectrodesTable( ...
    'description', 'Table for storing intracellular electrode related metadata.', ...
    'colnames', {'electrode'}, ...
    'id', types.hdmf_common.ElementIdentifiers( ...
        'data', int64([0, 1, 2]) ...
    ), ...
    'electrode', types.hdmf_common.VectorData( ...
        'data', repmat(types.untyped.ObjectView(electrode), 3, 1), ...
        'description', 'Column for storing the reference to the intracellular electrode' ...
    ) ...
);
```

```
ic_rec_table.responses = types.core.IntracellularResponsesTable( ...
    'description', 'Table for storing intracellular response related metadata.', ...
    'colnames', {'response'}, ...
    'id', types.hdmf_common.ElementIdentifiers( ...
        'data', int64([0, 1, 2]) ...
    ), ...
    'response', types.core.TimeSeriesReferenceVectorData( ...
        'description', 'Column storing the reference to the recorded response for the recording (rows)', ...
        'data', struct( ...
            'idx_start', [0, 2, 0], ...
            'count', [5, 3, 5], ...
            'timeseries', [ ...
                types.untyped.ObjectView(vcs), ...
                types.untyped.ObjectView(vcs), ...
                types.untyped.ObjectView(vcs) ...
            ] ...
        )...
    )...
);
```

Step 2.1: Add custom intracellular recording metadata

Add a custom column to a category table

```
% Add voltage threshold as column of electrodes table
ic_rec_table.electrodes.colnames = [ic_rec_table.electrodes.colnames {'voltage_threshold'}];
ic_rec_table.electrodes.vectordata.set('voltage_threshold', types.hdmf_common.VectorData( ...
    'data', [0.1, 0.12, 0.13], ...
    'description', 'Just an example column on the electrodes category table' ...
    ) ...
);
```

Add a custom category table

```
% add category
ic_rec_table.categories = [ic_rec_table.categories, {'recording_lab_data'}];
ic_rec_table.dynamictable.set( ...
    'recording_lab_data', types.hdmf_common.DynamicTable( ...
        'description', 'category table for lab-specific recording metadata', ...
        'colnames', {'location'}, ...
        'id', types.hdmf_common.ElementIdentifiers( ...
            'data', int64([0, 1, 2]) ...
        ), ...
        'location', types.hdmf_common.VectorData( ...
            'data', {'Mordor', 'Gondor', 'Rohan'}, ...
            'description', 'Recording location in Middle Earth' ...
        ) ...
    ) ...
);
```

Don't forget to add the IntracellularRecordingsTable to the NWB file!

```
nwbfile.general_intracellular_ephys_intracellular_recordings = ic_rec_table;
```

What we have so far:

```
nwbfile.intracellular_recordings.to_dataframe()
```

	intracellular_recordings	electrodes			stimuli		responses		recording_lab_data	
	recording_tag	id	electrode	voltage_threshold	id	stimulus	id	response	id	location
(intracellular_recordings, id)										
10	A1	0	elec0 pynwb.icephys.Intrac...	0.10	0	(0, 5, ccss pynwb.icephys....	0	(0, 5, vcs pynwb.icephys.V...	0	Mordor
11	A2	1	elec0 pynwb.icephys.Intrac...	0.12	1	(1, 3, ccss pynwb.icephys....	1	(2, 3, vcs pynwb.icephys.V...	1	Gondor
12	A3	2	elec0 pynwb.icephys.Intrac...	0.13	2	(None, None, None)	2	(0, 5, vcs pynwb.icephys.V...	2	Rohan

Step 3: Populate the SimultaneousRecordingsTable

Example: Adding a custom column to the SimultaneousRecordingsTable before we populate the table

```
icephys_simultaneous_recordings = nwbfile.get_icephys_simultaneous_recordings()
icephys_simultaneous_recordings.add_column(
    name='simultaneous_recording_tag',
    description='A custom tag for simultaneous_recordings'
)
```

Note: NWBFile.get_icephys_simultaneous_recordings() creates a new **SimultaneousRecordingsTable** if it doesn't exist, while the **NWBFile.icephys_simultaneous_recordings** property does not.

Adding a simultaneous recording

```
rowindex = nwbfile.add_icephys_simultaneous_recording(
    recordings=[rowindex, rowindex2, rowindex3],
    id=12,
    simultaneous_recording_tag='LabTag1'
)
```

Note: These are always **row indices** NOT id's.

Note: Since we added a custom column we now also need to specify the corresponding value when we add simultaneous recordings.

Query the IntracellularRecordingsTable to find row indices if we only know the id's

```
temp_row_indices = (nwbfile.intracellular_recordings.id == [10, 11])
```

Note: NWBFile.add_intracellular_recording returns the row index for the added.

Step 3: Populate the SimultaneousRecordingsTable

MATLAB®

Creating a SimultaneousRecordingsTable (with a custom column)

```
% create simultaneous recordings table with custom column
% 'simultaneous_recording_tag'

[recordings_vector_data, recordings_vector_index] = util.create_indexed_column( ...
    {[0, 1, 2],}, ...
    'Column with references to one or more rows in the IntracellularRecordingsTable table', ...
    ic_rec_table);

ic_sim_recs_table = types.core.SimultaneousRecordingsTable( ...
    'description', [ ...
        'A table for grouping different intracellular recordings from ', ...
        'the IntracellularRecordingsTable table together that were recorded ', ...
        'simultaneously from different electrodes.'...
    ], ...
    'colnames', {'recordings', 'simultaneous_recording_tag'}, ...
    'id', types.hdmf_common.ElementIdentifiers( ...
        'data', int64(12) ...
    ), ...
    'recordings', recordings_vector_data, ...
    'recordings_index', recordings_vector_index, ...
    'simultaneous_recording_tag', types.hdmf_common.VectorData( ...
        'description', 'A custom tag for simultaneous_recordings', ...
        'data', {'LabTag1'} ...
    ) ...
);
```

Adding the SimultaneousRecordingsTable to the NWB file

```
nwbfile.general_intracellular_ephys_simultaneous_recordings = ic_sim_recs_table;
```

Step 4 – 7: Populate the remaining tables and write

- Step 4: Populating the SequentialRecordingsTable

```
rowindex = nwbfile.add_icephys_sequential_recording(
    simultaneous_recordings=[0],
    stimulus_type='square',
    id=15
)
```

- Step 5: Populating the RepetitionsTable

```
rowindex = nwbfile.add_icephys_repetition(sequential_recordings=[0], id=17)
```

- Step 6: Populating the ExperimentalConditionsTable

```
rowindex = nwbfile.add_icephys_experimental_condition(repetitions=[0], id=19)
```

- Step 7: Write the NWBFile as usual:

```
testpath = "test_icephys_file.nwb"
with NWBHDF5IO(testpath, 'w') as io:
    io.write(nwbfile)
```

Step 4 – 7: Populate the remaining tables and write

- Step 4: Populating the SequentialRecordingsTable

```
[simultaneous_recordings_vector_data, simultaneous_recordings_vector_index] = util.create_indexed_column( ...
    {0,}, ...
    'Column with references to one or more rows in the SimultaneousRecordingsTable table', ...
    ic_sim_recs_table);

sequential_recordings = types.core.SequentialRecordingsTable( ...
    'description', [ ...
        'A table for grouping different intracellular recording ', ...
        'simultaneous_recordings from the SimultaneousRecordingsTable ', ...
        'table together. This is typically used to group together ', ...
        'simultaneous_recordings where the a sequence of stimuli of ', ...
        'the same type with varying parameters have been presented in ', ...
        'a sequence.' ...
    ], ...
    'colnames', {'simultaneous_recordings', 'stimulus_type'}, ...
    'id', types.hdmf_common.ElementIdentifiers( ...
        'data', int64(15) ...
    ), ...
    'simultaneous_recordings', simultaneous_recordings_vector_data, ...
    'simultaneous_recordings_index', simultaneous_recordings_vector_index, ...
    'stimulus_type', types.hdmf_common.VectorData( ...
        'description', 'Column storing the type of stimulus used for the sequential recording', ...
        'data', {'square'} ...
    ) ...
);

nwbfile.general_intracellular_ephys_sequential_recordings = sequential_recordings;
```

Step 4 – 7: Populate the remaining tables and write

- Step 5: Populating the RepetitionsTable

```
[sequential_recordings_vector_data, sequential_recordings_vector_index] = util.create_indexed_column( ...
    {0,}, ...
    'Column with references to one or more rows in the SequentialRecordingsTable table', ...
    sequential_recordings);


nwbfile.general_intracellular_ephys_repetitions = types.core.RepetitionsTable( ...
    'description', [ ...
        'A table for grouping different intracellular recording sequential ', ...
        'recordings together. With each SimultaneousRecording typically ', ...
        'representing a particular type of stimulus, the RepetitionsTable ', ...
        'table is typically used to group sets of stimuli applied in sequence.' ...
    ], ...
    'colnames', {'sequential_recordings'}, ...
    'id', types.hdmf_common.ElementIdentifiers( ...
        'data', int64(17) ...
    ), ...
    'sequential_recordings', sequential_recordings_vector_data, ...
    'sequential_recordings_index', sequential_recordings_vector_index ...
);
```

Step 4 – 7: Populate the remaining tables and write

- Step 6: Populating the ExperimentalConditionsTable

```
[repetitions_vector_data, repetitions_vector_index] = util.create_indexed_column( ...
    {0, 0}, ...
    'Column with references to one or more rows in the RepetitionsTable table', ...
    nwbfile.general_intracellular_ephys_repetitions);

nwbfile.general_intracellular_ephys_experimental_conditions = types.core.ExperimentalConditionsTable( ...
    'description', [ ...
        'A table for grouping different intracellular recording ', ...
        'repetitions together that belong to the same experimental ', ...
        'conditions.' ...
    ], ...
    'colnames', {'repetitions', 'tag'}, ...
    'id', types.hdmf_common.ElementIdentifiers( ...
        'data', int64([19, 21]) ...
    ), ...
    'repetitions', repetitions_vector_data, ...
    'repetitions_index', repetitions_vector_index, ...
    'tag', types.hdmf_common.VectorData( ...
        'description', 'integer tag for a experimental condition', ...
        'data', [1,3] ...
    ) ...
);
```

- Step 7: Write the NWBFile as usual:

```
nwbExport(nwbfile, 'test_new_icephys.nwb');
```

Step 8: Reading the file

- Read the NWB file:

```
# Read the data back in
with NWBHDF5IO(testpath, 'r') as io:
    infile = io.read()
```

- Access intracellular stimuli: **nwbfile.stimulus** and **nwbfile.get_stimulus(...)**
- Access intracellular responses: : **nwbfile.acquisition** and **nwbfile.get_acquisition(...)**
- Access intracellular electrodes : **nwbfile.icephys_electrodes** and **nwbfile.get_icephys_electrode(...)**
- Access intracellular electrodes : **nwbfile.icephys_electrodes** and **nwbfile.get_icephys_electrode(...)**

Step 8: Reading the file

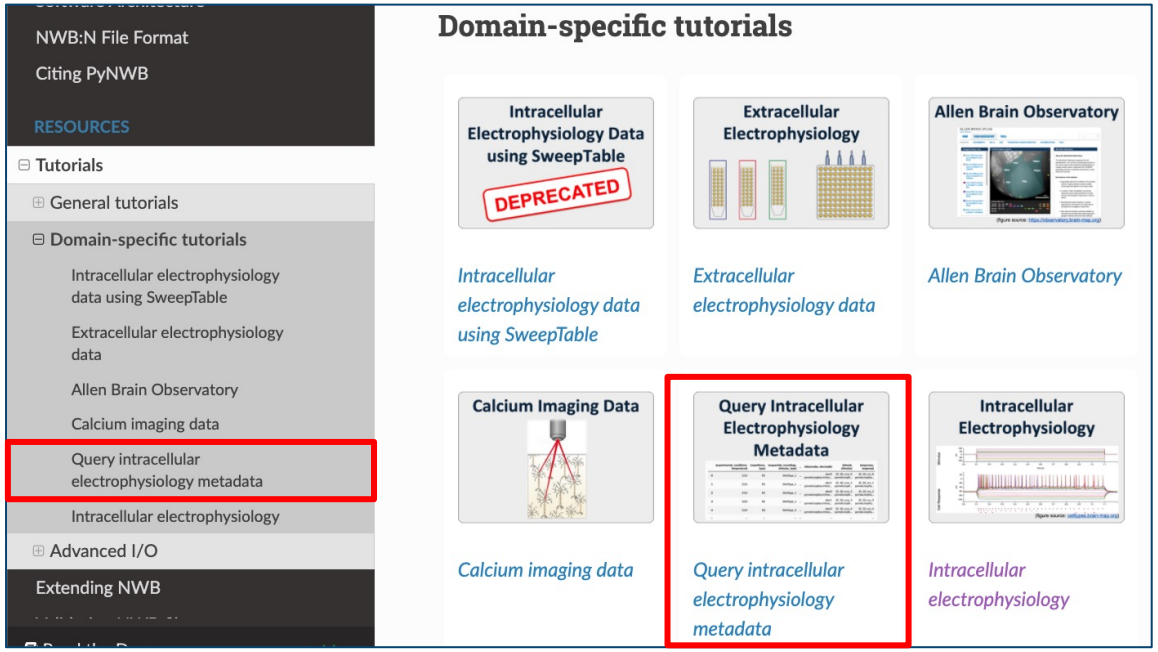
- Read the NWB file:

```
nwbfile2 = nwbRead('test_new_icephys.nwb')

nwbfile2 =
  NwbFile with properties:

                                        nwb_version: '2.4.0'
                                        acquisition: [1×1 types.untyped.Set]
                                           analysis: [0×1 types.untyped.Set]
                                   file_create_date: [1×1 types.untyped.DataStub]
                                            general: [0×1 types.untyped.Set]
                                 general_data_collection: []
                                    general_devices: [1×1 types.untyped.Set]
                       general_experiment_description: 'I went on an adventure with thirteen
                              general_experimenter: [1×1 types.untyped.DataStub]
                         general_extracellular_ephys: [0×1 types.untyped.Set]
                  general_extracellular_ephys_electrodes: []
                                 general_institution: 'University of Middle Earth at the Shi
```

- Access intracellular stimuli: **nwbfile.stimulus_presentation** and **nwbfile.stimulus_presentation.get(…)**
- Access intracellular responses: : **nwbfile.acquisition** and **nwbfile.acquisition.get(…)**

Querying the Intracellular Electrophysiology Experiments Metadata Tables Using PyNWB and pandas

PyNWB tutorial at :
https://pynwb.readthedocs.io/en/stable/tutorials/domain/plot_icephys_pandas.html



Accessing the ICEphys tables

- Getting the root table:

```
root_table = nwbfile.get_icephys_meta_parent_table()
```

- Use the NWBFile icephys table properties to access a specific table and check if the table exists.
  Warning: Do not use the **NWBFile.get_icephys_…()** methods just to access a table, as this will create a table if it doesn't exist!

```
nwbfile.icephys_sequential_recordings is not None
```

- To inspect the table hierarchy, we can also use the get_foreign_columns and get_linked_tables functions of the tables

```
linked_tables = root_table.get_linked_tables()

# Print the links
for i, link in enumerate(linked_tables):
    print("%s (%s, %s) ----> %s" % ("    " * i,
                                    link.source_table.name,
                                    link.source_column.name,
                                    link.target_table.name))
```

Out:

```
(experimental_conditions, repetitions) ----> repetitions
    (repetitions, sequential_recordings) ----> sequential_recordings
        (sequential_recordings, simultaneous_recordings) ----> simultaneous_recordings
            (simultaneous_recordings, recordings) ----> intracellular_recordings
```

Convert individual tables to nested DataFrames

```
exp_cond_df = root_table.to_dataframe()
exp_cond_df
```

id	repetitions	temperature
100000	sequential_...	32.0
100001	sequential_...	24.0

```
exp_cond_df.iloc[0]['repetitions']
```

id	sequential_recordings	type
10000	simultaneous_r...	R1
10001	simultaneous_r...	R2

Convert individual tables to indexed DataFrames

```
root_table.to_dataframe(index=True)
```

id	repetitions	temperature
100000	[0, 1]	32.0
100001	[2, 3]	24.0

Perform lookups in linked tables via the DynamicTable

```
root_table['repetitions'][0]
```

Or by resolving the links manually

```
target_table = root_table['repetitions'].target.table
target_table[[0, 1]]
```

id	sequential_recordings	type
10000	simultaneous_r...	R1
10001	simultaneous_r...	R2

Convert the tables to a single hierarchical DataFrame

```
from hdmf.common.hierarchicaltable import to_hierarchical_dataframe
icephys_meta_df = to_hierarchical_dataframe(root_table)
```

								source_table		intracellular_recordings								
								label	id	(intracellular_recordings, recording_tags)	(electrodes, id)	...	(stimuli, stimulus)	(responses, id)	(responses, response)
(experimental_conditions, id)	(experimental_conditions, temperature)	(repetitions, id)	(repetitions, type)	(sequential_recordings, id)	(sequential_recordings, stimulus_type)	(sequential_recordings, type)	(simultaneous_recordings, id)	(simultaneous_recordings, tag)										
100000	32.0	10000	R1	1000	StimType_1	T1	100	0	0	A1	0	...	(None, None, None)	0	(0, 10, vcs_0 pynwb.icephy...
								0	1	A2	1	...	(0, 10, ccss_1 pynwb.iceph...	1	(0, 10, vcs_1 pynwb.icephy...
							101	1	2	B1	2	...	(0, 10, ccss_2 pynwb.iceph...	2	(0, 10, vcs_2 pynwb.icephy...
								1	3	B2	3	...	(0, 10, ccss_3 pynwb.iceph...	3	(0, 10, vcs_3 pynwb.icephy...
		10001	R2	1001	StimType_2	T2	102	2	4	C1	4	...	(0, 10, ccss_4 pynwb.iceph...	4	(0, 10, vcs_4 pynwb.icephy...
								2	5	C2	5	...	(0, 10, ccss_5 pynwb.iceph...	5	(0, 10, vcs_5 pynwb.icephy...
								2	6	C3	6	...	(0, 10, ccss_6 pynwb.iceph...	6	(0, 10, vcs_6 pynwb.icephy...
				1002	StimType_3	T3	103	3	7	D1	7	...	(0, 10, ccss_7 pynwb.iceph...	7	(0, 10, vcs_7 pynwb.icephy...
								3	8	D2	8	...	(0, 10, ccss_8 pynwb.iceph...	8	(0, 10, vcs_8 pynwb.icephy...
								3	9	D3	9	...	(0, 10, ccss_9 pynwb.iceph...	9	(0, 10, vcs_9 pynwb.icephy...
100001	24.0	10002	R1	1003	StimType_1	T1	104	4	10	A1	10	...	(None, None, None)	10	(0, 10, vcs_10 pynwb.iceph...
								4	11	A2	11	...	(0, 10, ccss_11 pynwb.icep...	11	(0, 10, vcs_11 pynwb.iceph...
							105	5	12	B1	12	...	(0, 10, ccss_12 pynwb.icep...	12	(0, 10, vcs_12 pynwb.iceph...
								5	13	B2	13	...	(0, 10, ccss_13 pynwb.icep...	13	(0, 10, vcs_13 pynwb.iceph...
		10003	R2	1004	StimType_2	T2	106	6	14	C1	14	...	(0, 10, ccss_14 pynwb.icep...	14	(0, 10, vcs_14 pynwb.iceph...
								6	15	C2	15	...	(0, 10, ccss_15 pynwb.icep...	15	(0, 10, vcs_15 pynwb.iceph...
								6	16	C3	16	...	(0, 10, ccss_16 pynwb.icep...	16	(0, 10, vcs_16 pynwb.iceph...
				1005	StimType_3	T3	107	7	17	D1	17	...	(0, 10, ccss_17 pynwb.icep...	17	(0, 10, vcs_17 pynwb.iceph...
								7	18	D2	18	...	(0, 10, ccss_18 pynwb.icep...	18	(0, 10, vcs_18 pynwb.iceph...
								7	19	D3	19	...	(0, 10, ccss_19 pynwb.icep...	19	(0, 10, vcs_19 pynwb.iceph...

Convert the tables to a single, denormalized DataFrame

```
from hdmf.common.hierarchicaltable import drop_id_columns, flatten_column_index
# Reset the index of the dataframe and turn the values into columns instead
icephys_meta_df.reset_index(inplace=True)
# Flatten the column-index, turning the pandas.MultiIndex into a pandas.Index of tuples
flatten_column_index(dataframe=icephys_meta_df, max_levels=2, inplace=True)
# Remove the id columns. By setting inplace=False allows us to visualize the result of this
# action while keeping the id columns in our main icephys_meta_df table
drop_id_columns(dataframe=icephys_meta_df, inplace=False)
```

	(experimental_conditions, temperature)	(repetitions, type)	(sequential_recordings, stimulus_type)	(sequential_recordings, type)	(simultaneous_recordings, tag)	(intracellular_recordings, recording_tags)	(electrodes, electrode)	(stimuli, stimulus)	(responses, response)
0	32.0	R1	StimType_1	T1	0	A1	elec0 pynwb.icephys.Intrac...	(None, None, None)	(0, 10, vcs_0 pynwb.icephy...
1	32.0	R1	StimType_1	T1	0	A2	elec1 pynwb.icephys.Intrac...	(0, 10, ccss_1 pynwb.iceph...	(0, 10, vcs_1 pynwb.icephy...
2	32.0	R1	StimType_1	T1	1	B1	elec0 pynwb.icephys.Intrac...	(0, 10, ccss_2 pynwb.iceph...	(0, 10, vcs_2 pynwb.icephy...
3	32.0	R1	StimType_1	T1	1	B2	elec1 pynwb.icephys.Intrac...	(0, 10, ccss_3 pynwb.iceph...	(0, 10, vcs_3 pynwb.icephy...
4	32.0	R2	StimType_2	T2	2	C1	elec0 pynwb.icephys.Intrac...	(0, 10, ccss_4 pynwb.iceph...	(0, 10, vcs_4 pynwb.icephy...
5	32.0	R2	StimType_2	T2	2	C2	elec1 pynwb.icephys.Intrac...	(0, 10, ccss_5 pynwb.iceph...	(0, 10, vcs_5 pynwb.icephy...
6	32.0	R2	StimType_2	T2	2	C3	elec0 pynwb.icephys.Intrac...	(0, 10, ccss_6 pynwb.iceph...	(0, 10, vcs_6 pynwb.icephy...
7	32.0	R2	StimType_3	T3	3	D1	elec1 pynwb.icephys.Intrac...	(0, 10, ccss_7 pynwb.iceph...	(0, 10, vcs_7 pynwb.icephy...
8	32.0	R2	StimType_3	T3	3	D2	elec0 pynwb.icephys.Intrac...	(0, 10, ccss_8 pynwb.iceph...	(0, 10, vcs_8 pynwb.icephy...
9	32.0	R2	StimType_3	T3	3	D3	elec1 pynwb.icephys.Intrac...	(0, 10, ccss_9 pynwb.iceph...	(0, 10, vcs_9 pynwb.icephy...
10	24.0	R1	StimType_1	T1	4	A1	elec0 pynwb.icephys.Intrac...	(None, None, None)	(0, 10, vcs_10 pynwb.iceph...
11	24.0	R1	StimType_1	T1	4	A2	elec1 pynwb.icephys.Intrac...	(0, 10, ccss_11 pynwb.icep...	(0, 10, vcs_11 pynwb.iceph...
12	24.0	R1	StimType_1	T1	5	B1	elec0 pynwb.icephys.Intrac...	(0, 10, ccss_12 pynwb.icep...	(0, 10, vcs_12 pynwb.iceph...
13	24.0	R1	StimType_1	T1	5	B2	elec1 pynwb.icephys.Intrac...	(0, 10, ccss_13 pynwb.icep...	(0, 10, vcs_13 pynwb.iceph...
14	24.0	R2	StimType_2	T2	6	C1	elec0 pynwb.icephys.Intrac...	(0, 10, ccss_14 pynwb.icep...	(0, 10, vcs_14 pynwb.iceph...
15	24.0	R2	StimType_2	T2	6	C2	elec1 pynwb.icephys.Intrac...	(0, 10, ccss_15 pynwb.icep...	(0, 10, vcs_15 pynwb.iceph...
16	24.0	R2	StimType_2	T2	6	C3	elec0 pynwb.icephys.Intrac...	(0, 10, ccss_16 pynwb.icep...	(0, 10, vcs_16 pynwb.iceph...
17	24.0	R2	StimType_3	T3	7	D1	elec1 pynwb.icephys.Intrac...	(0, 10, ccss_17 pynwb.icep...	(0, 10, vcs_17 pynwb.iceph...
18	24.0	R2	StimType_3	T3	7	D2	elec0 pynwb.icephys.Intrac...	(0, 10, ccss_18 pynwb.icep...	(0, 10, vcs_18 pynwb.iceph...
19	24.0	R2	StimType_3	T3	7	D3	elec1 pynwb.icephys.Intrac...	(0, 10, ccss_19 pynwb.icep...	(0, 10, vcs_19 pynwb.iceph...

Augmenting the denormalized DataFrame

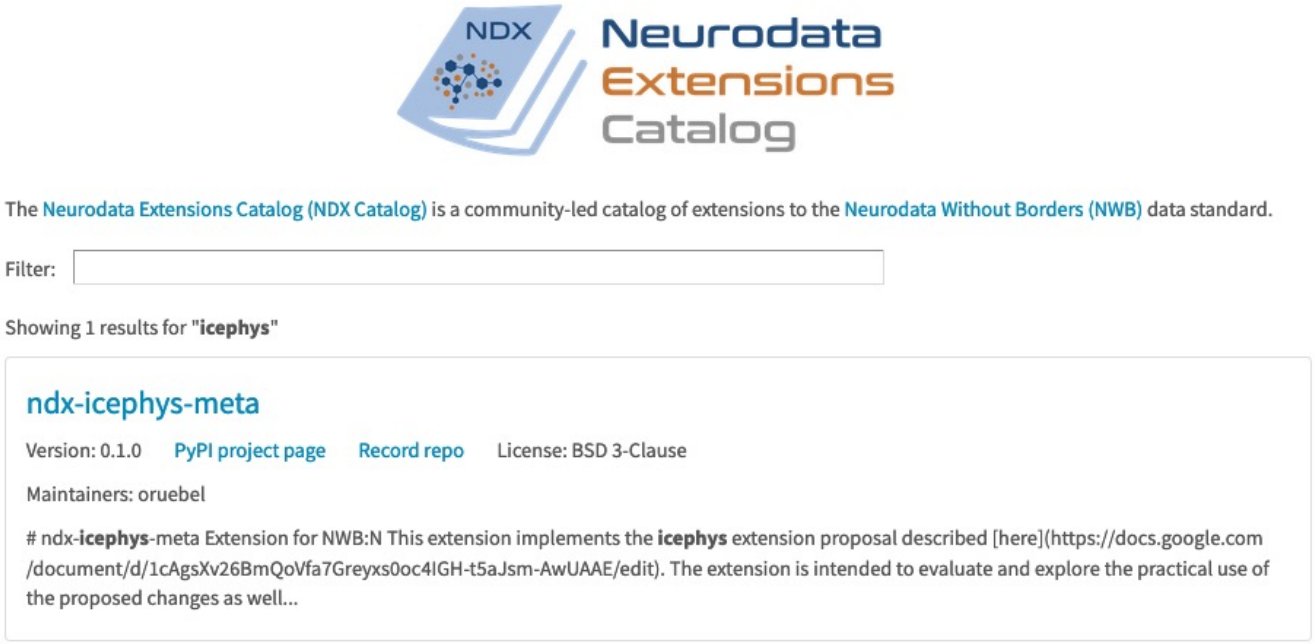
```python
# Add a column with the name of the stimulus TimeSeries object.
# Note: We use getattr here to easily deal with missing values, i.e., cases where no stimulus is present
col = ('stimuli', 'name')
icephys_meta_df[col] = [getattr(s.timeseries, 'name', None)
                        for s in icephys_meta_df[('stimuli', 'stimulus')]]

# Often we can easily do this in bulk-fashion by specifing the collection of fields of interest
for field in ['neurodata_type', 'gain', 'rate', 'starting_time', 'object_id']:
    col = ('stimuli', field)
    icephys_meta_df[col] = [getattr(s.timeseries, field, None)
                            for s in icephys_meta_df[('stimuli', 'stimulus')]]
icephys_meta_df
```

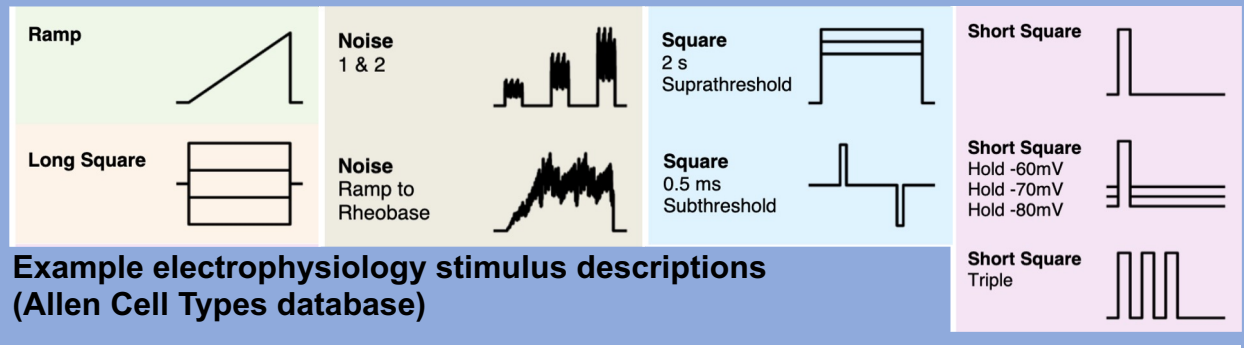
	(experimental_conditions, id)	(experimental_conditions, temperature)	(repetitions, id)	(repetitions, type)	(sequential_recordings, id)	(sequential_recordings, stimulus_type)	(sequential_recordings, type)	(simultaneous_recordings, id)	(simultaneous_recordings, tag)	(intracellular_recordings, id)	...	(stimuli, id)	(stimuli, stimulus)	(responses, id)	(responses, response)	(stimuli, name)	(stimuli, neurodata_type)	(stimuli, gain)	(stimuli, rate)	(stimuli, starting_time)	(stimuli, object_id)
0	100000	32.0	10000	R1	1000	StimType_1	T1	100	0	0	...	0	(None, None, None)	0	(0, 10, vcs_0 pynwb.icephy...	None	None	NaN	NaN	NaN	None
1	100000	32.0	10000	R1	1000	StimType_1	T1	100	0	1	...	1	(0, 10, ccss_1 pynwb.iceph...	1	(0, 10, vcs_1 pynwb.icephy...	ccss_1	VoltageClampStimulusSeries	0.607860	8000.0	29.715988	43b6fa59-188e-4f30-8b37-28...
2	100000	32.0	10000	R1	1000	StimType_1	T1	101	1	2	...	2	(0, 10, ccss_2 pynwb.iceph...	2	(0, 10, vcs_2 pynwb.icephy...	ccss_2	VoltageClampStimulusSeries	0.631178	10000.0	43.637043	05841e68-97e4-47a9-beb1-45...
3	100000	32.0	10000	R1	1000	StimType_1	T1	101	1	3	...	3	(0, 10, ccss_3 pynwb.iceph...	3	(0, 10, vcs_3 pynwb.icephy...	ccss_3	VoltageClampStimulusSeries	0.200074	6000.0	7.863054	f9c99c53-40ab-49fe-8aba-fb...
4	100000	32.0	10001	R2	1001	StimType_2	T2	102	2	4	...	4	(0, 10, ccss_4 pynwb.iceph...	4	(0, 10, vcs_4 pynwb.icephy...	ccss_4	VoltageClampStimulusSeries	0.160150	2000.0	28.665112	5879972f-b8f6-4183-bfcc-6c...
5	100000	32.0	10001	R2	1001	StimType_2	T2	102	2	5	...	5	(0, 10, ccss_5 pynwb.iceph...	5	(0, 10, vcs_5 pynwb.icephy...	ccss_5	VoltageClampStimulusSeries	0.313899	3000.0	61.525763	95cfd4f4-a824-4704-ae93-f4...
6	100000	32.0	10001	R2	1001	StimType_2	T2	102	2	6	...	6	(0, 10, ccss_6 pynwb.iceph...	6	(0, 10, vcs_6 pynwb.icephy...	ccss_6	VoltageClampStimulusSeries	0.309658	1000.0	64.311643	9820e6d9-9d8e-43ad-877c-82...
7	100000	32.0	10001	R2	1002	StimType_3	T3	103	3	7	...	7	(0, 10, ccss_7 pynwb.iceph...	7	(0, 10, vcs_7 pynwb.icephy...	ccss_7	VoltageClampStimulusSeries	0.573035	6000.0	67.561140	8bfa5f9d-78bb-4dde-9856-f1...
8	100000	32.0	10001	R2	1002	StimType_3	T3	103	3	8	...	8	(0, 10, ccss_8 pynwb.iceph...	8	(0, 10, vcs_8 pynwb.icephy...	ccss_8	VoltageClampStimulusSeries	0.768311	4000.0	22.019846	65ec0230-49f2-4714-8906-80...
9	100000	32.0	10001	R2	1002	StimType_3	T3	103	3	9	...	9	(0, 10, ccss_9 pynwb.iceph...	9	(0, 10, vcs_9 pynwb.icephy...	ccss_9	VoltageClampStimulusSeries	0.035710	7000.0	13.459802	d75e1106-d783-4f4e-9927-f1...
10	100001	24.0	10002	R1	1003	StimType_1	T1	104	4	10	...	10	(None, None, None)	10	(0, 10, vcs_10 pynwb.iceph...	None	None	NaN	NaN	NaN	None
11	100001	24.0	10002	R1	1003	StimType_1	T1	104	4	11	...	11	(0, 10, ccss_11 pynwb.icep...	11	(0, 10, vcs_11 pynwb.iceph...	ccss_11	VoltageClampStimulusSeries	0.357555	9000.0	96.474344	ab742195-cbb0-4288-afe0-b2...
12	100001	24.0	10002	R1	1003	StimType_1	T1	105	5	12	...	12	(0, 10, ccss_12 pynwb.icep...	12	(0, 10, vcs_12 pynwb.iceph...	ccss_12	VoltageClampStimulusSeries	0.172989	2000.0	49.171500	4a0e2f2c-6027-4dad-a977-39...
13	100001	24.0	10002	R1	1003	StimType_1	T1	105	5	13	...	13	(0, 10, ccss_13 pynwb.icep...	13	(0, 10, vcs_13 pynwb.iceph...	ccss_13	VoltageClampStimulusSeries	0.149877	8000.0	69.699305	f0e6f3a5-fb52-4337-8a16-94...
14	100001	24.0	10003	R2	1004	StimType_2	T2	106	6	14	...	14	(0, 10, ccss_14 pynwb.icep...	14	(0, 10, vcs_14 pynwb.iceph...	ccss_14	VoltageClampStimulusSeries	0.063291	10000.0	64.532674	1716dcc6-6a10-40ea-80d6-05...
15	100001	24.0	10003	R2	1004	StimType_2	T2	106	6	15	...	15	(0, 10, ccss_15 pynwb.icep...	15	(0, 10, vcs_15 pynwb.iceph...	ccss_15	VoltageClampStimulusSeries	0.162957	4000.0	49.291549	4d4ad96d-bdb1-48fe-ae73-21...
16	100001	24.0	10003	R2	1004	StimType_2	T2	106	6	16	...	16	(0, 10, ccss_16 pynwb.icep...	16	(0, 10, vcs_16 pynwb.iceph...	ccss_16	VoltageClampStimulusSeries	0.987200	7000.0	69.485013	1a87d525-3909-4cc0-8104-82...
17	100001	24.0	10003	R2	1005	StimType_3	T3	107	7	17	...	17	(0, 10, ccss_17 pynwb.icep...	17	(0, 10, vcs_17 pynwb.iceph...	ccss_17	VoltageClampStimulusSeries	0.545845	4000.0	11.603639	605c60ae-b143-4716-bda2-4f...
18	100001	24.0	10003	R2	1005	StimType_3	T3	107	7	18	...	18	(0, 10, ccss_18 pynwb.icep...	18	(0, 10, vcs_18 pynwb.iceph...	ccss_18	VoltageClampStimulusSeries	0.029371	4000.0	45.814067	5e08b570-da25-4569-89a8-d8...
19	100001	24.0	10003	R2	1005	StimType_3	T3	107	7	19	...	19	(0, 10, ccss_19 pynwb.icep...	19	(0, 10, vcs_19 pynwb.iceph...	ccss_19	VoltageClampStimulusSeries	0.904281	2000.0	33.859644	bf48e0d9-7501-49f7-81ca-3d...

The Role of Neurodata Extensions (NDX) in Advancing Neurophysiology Data Standardization

ICEphys Metadata Extensions

- The ICEPhys experiment metadata model was developed as part of an NDX extension proposal developed in collaboration with the member of the BlueBrainProject (channelpedia.epfl.ch) and the Allen Institute for Brain Science (celltypes.brain-map.org)
- The NDX extension and proposal were made available for public review as part of the NDX Catalog and integrated with NWB core in v.2.4 (PyNWB 2.0, MatNWB 2.4.0.0)





Example electrophysiology stimulus descriptions (Allen Cell Types database)

Electrophysiology Stimulation Extension

Goal: Enable the FAIR description of electrophysiology stimulation

Approach: Founded INCF Electrophysiology Stimulation Ontology Working Group

- **Ontology:** Created ontology for 1D stimulus waveforms (built by Tom Gillespie (UCSD) and Patrick Ray (AIBS))
- **NDX:** Developed NWB extension to enable standardized description and storage of formal, parameterized descriptions of stimulus functions
- **External Resources:** Link *StimulusFunctionTable* to the stimulus ontology

StimulusFunctionTable

id	function_name	float_parameter compound_type			functional_parameter compound_type		
		name text	value float	unit text	name text	function_index float	unit text
0	square	duration	1.0	s			
		start_time	0.1	s			
1	triangle	amplitude	5	V			
2	square	duration	1.5	s			
		start_time	3	s			
3	sin	amplitude	2	V	frequency	1	Hz
4	composition				func1	0	V
					func2	2	V
					func3	3	V
…	…	…	…	…	…	…	…

ExternalResources

objects		keys	entities	resources
object_id	field	key	…	…
c8b35201-0fba-4574-be1d-bd516f2b5678	float_parameter/name	duration	…	…
		…	…	
c8b35201-0fba-4574-be1d-bd516f2b5678	function_name	square	…	…
		triangle	…	
		…	…	
…	…	…	…	…

NEURODATA
WITHOUT BORDERS

Visit us at NWB.org and nwb-overview.readthedocs.io

2019 R&D 100 WINNER

incf ENDORSED

BERKELEY LAB
Bringing Science Solutions to the World

Legal

BERKELEY LAB
Bringing Science Solutions to the World

- **Disclaimer:** This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor the Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or the Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or the Regents of the University of California.
-