
Alyx Documentation

Release 0.1

Max Hunter, Chris Burgess, Kenneth Harris

Sep 11, 2017

1	Motivation	2
2	Design considerations	3
2.1	Metadata vs bulk data	3
2.2	Unstructured and arbitrary metadata	3
2.3	Unique identifiers	4
2.4	API and webform access	4
2.5	Sharing	4
3	API specification	5
3.1	Versioning	5
3.2	Schema	5
3.3	Authentication and public endpoints	6
3.4	Root endpoint	6
3.5	Summary vs detailed representations	6
3.6	Parameters	6
3.7	Hypermedia and URL cross-links	7
4	Users API	8
4.1	Get all users	8
4.2	Get a specific user	9
4.3	List all actions for a user	9
5	Subjects API	11
5.1	List all subjects	11
5.2	Get a specific subject	12
5.3	List all actions for a subject	13
5.4	List all datasets for a subject	14
6	Actions API	15
6.1	List all actions	15
6.2	Get a specific action	16
7	Datasets API	18
7.1	List all datasets	18
7.2	Get a specific dataset	19

8	List of models	20
8.1	Coordinate transforms	20
8.2	Subjects	22
8.3	Actions	26
8.4	Equipment	30
8.5	Data	32
8.6	Behavior	35
8.7	Electrophysiology	39
8.8	Imaging	43
9	Indices and tables	47
	Python Module Index	48

Alyx is a fast, flexible database designed for easy storage and retrieval of all data in an experimental neuroscience laboratory - from subject management through data acquisition, raw data file tracking and storage of metadata resulting from manual analysis.

Alyx is not yet production-ready and is still pre-alpha. For information on a usable, testable beta-release, please email maximilian.hunter@ucl.ac.uk

Alyx is built using industry-standard tools ([PostgreSQL 9.5](#) and [Django 1.9](#)) and designed to be interacted with using online webforms to enter and retrieve data manually, or with a documented and easy-to-use REST API programmatically (using built-in functions in [MATLAB](#), [Python](#), and most other modern programming languages, or command-line tools such as [curl](#)). It is easy to integrate with existing applications and allows for powerful queries to be performed to filter and return a specific subset of records in milliseconds. It requires minimal setup and can be hosted on your own internal server or in the cloud, for example with Amazon EC2.

Table of contents:

Motivation

Previous standardisation efforts, as well as current data organisation methods used in laboratories, have several drawbacks. Most recently, the [Neurodata Without Borders](#) project has worked to standardise a format for all experimental neurophysiology data. Alyx builds on this work and adds a number of key advantages:

- **Searchability:** To use data, one has to first find the data. Alyx allows a user to quickly and simply search a database of neurophysiology experiments to find that needed for their scientific question. The search could run over all data collected in the user’s own lab, or all the shared data in the world.
- **Lightweight organization:** A barrier to use of the current NWB format is its monolithic nature: in order use a dataset, a user must download the entire file, even if they only need a small part of it. The large size of these files presents a serious barrier to many users.
- **Ease of use:** The HDF5 format at the base of the current NWB format is an obstacle to its adoption by the neurophysiology community; it will be replaced by simple binary files.
- **Cloud-ready:** As more scientists move to cloud-based computing platforms, it is essential that large data files be quickly readable on these systems. HDF5 presents problems in this regard, that are solved by simple binary files.
- **Encouraging uptake:** Working scientists will only switch from their current file formats if there is a strong incentive to do so. The proposed format comes with two “killer apps” that will encourage widespread adoption: a REST API to facilitate those building scientific tools in integrating support for Alyx into their applications, and set of online webforms to ensure all manually-entered metadata is provided by experimenters.

Design considerations

Metadata vs bulk data

The data accompanying a neurophysiology experiment has two components: **metadata**, which has complex relational organization but small size; and **bulk data**, which consists of large numerical arrays. While the boundary between metadata and bulk data is not always clear, here we use a simple definition: *metadata is any data that a user will want to search over*. Details of an experiment, and summary statistics of the recorded neurons would be searchable metadata: for example, a user could search for all electrophysiological recordings made in hippocampal area CA1 during linear-track running that contain at least 20 well-isolated narrow-spiking units. However the precise spike times and extracellular waveforms of each unit would be unsearchable bulk data.

Storage of complex relational data, and storage of large numerical arrays are both solved problems in computing. We apply tried-and-tested solutions:

- Metadata is stored in an SQL database
- Bulk data is stored with one binary file per numerical array, and the URI (i.e. location) of each file stored in the SQL database.

Unstructured and arbitrary metadata

It is impossible to predict the metadata users will want to store for their experiments, as experimental paradigms are constantly evolving. Traditional relational databases, however, require all columns to be defined ahead of time. We will solve this problem by giving each table an additional “json” column that contains a JSON-formatted string, which can easily be converted into a MATLAB struct. PostgreSQL allows such fields to be searched in a similar way to traditional columns.

Unique identifiers

Databases identify each entry in a table with a “primary key”. As primary keys, we give each entry in a table (e.g. a subject, a lab, an experiment, or a neuron) a “GUID” – a random 128-bit number – which makes the chance of key duplication vanishingly small, even if this schema were used to store all the world’s neuroscience data. This GUID is assigned when the item is first added to the database (e.g. when a web form is filled in for an experiment; after spike sorting; etc.). The use of GUIDs allows users to copy or transfer their data to centralized stores with negligible danger of ID clash.

API and webform access

All database tables described in *the Models page* are accessible by a series of online webforms; each webform broadly represents one table. A subset of data, described in *the API documentation*, are accessible programmatically via an online HTTP REST API; these are ‘serialized’ to avoid users having to write their own database joins and views as much as possible.

Initially, only the data which is likely to be accessed programmatically is made accessible via the API; subjects, experiments, data and analysis results are all exposed in the API, but information about the lab or editing detailed user permissions information is not.

Sharing

This system is designed to be used within a lab, as an “electronic lab notebook” that keeps track of the experiments performed and datasets produced. Eventually, multi-lab support is planned:

This system also makes data sharing straightforward: a lab can share selected datasets by granting read-only access to the REST API and bulk data store; and a user can analyze this data using exactly the same tools they would use to analyze data collected in their own lab. Alternatively, the bulk data files and database entries for a set of completed experiments can be uploaded to a centralized store, to ensure long-term archival without requiring continuing support from the originating lab.

This page describes the structure of the Alyx v1 API. While Alyx is still under development, this is liable to change.

Versioning

All requests receive the v1 version of the API. For forwards-compatibility, we encourage you to explicitly request this version via the Accept header:

```
Accept: application/vnd.alyx.v1+json
```

Schema

All API access should take place over HTTPS; all data is sent and received as JSON:

```
curl -i http://alyx.cortexlab.net/  
HTTP/1.1 200 OK  
Date: Tue, 12 Jul 2016 12:58:38 GMT  
Server: Apache/2.4.7 (Ubuntu)  
Vary: Accept, Cookie  
Allow: GET, OPTIONS  
X-Frame-Options: SAMEORIGIN  
Transfer-Encoding: chunked  
Content-Type: application/json
```

Blank fields are included as null instead of being omitted.

All timestamps are returned in ISO 8601 format:

```
YYYY-MM-DDTHH:MM:SSZ
```


Authentication and public endpoints

By default, all endpoints (except for a summary representation of users and their subjects) are only accessible after authentication.

To authenticate, you need to fetch a token. By default, these expire after 24 hours.

```
curl -X POST -F 'username=foo' -F 'password=bar' https://alyx.cortexlab.net/auth-
↪token/
{"token":"c719825a24d13ddc52969ba240f9ab6353783095"}
```

Once you have a token, you should pass it as an *Authorization* header to any request you make:

```
curl -X GET http://alyx.cortexlab.net/subjects/ -H 'Authorization: Token_
↪c719825a24d13ddc52969ba240f9ab6353783095'
```

If you pass the wrong credentials, the API will return an error:

```
curl -X POST -F 'username=foo' -F 'password=wrong' https://alyx.cortexlab.net/auth-
↪token/
{"non_field_errors":["Unable to log in with provided credentials."]}
```

Similarly, if the token has expired or is invalid:

```
curl -X GET http://alyx.cortexlab.net/subjects/ -H 'Authorization: Token 123456789abc'
{"detail":"Invalid token."}
```

Root endpoint

You can issue a GET request to the root endpoint to get all the endpoint categories that the API supports.

```
curl https://alyx.cortexlab.net
```

Summary vs detailed representations

When you fetch a list of items, the response may include a subset of the attributes for that resource for performance or bandwidth reasons. This is the “summary” representation of the resource. However, when you fetch an individual item, the full record is returned.

For example, here we fetch a summary representation of all subjects:

```
GET /subjects
```

And now, we request the detailed representation of one specific subject:

```
GET /subjects/EJ010
```

Parameters

Many API methods take optional parameters. For GET requests, any parameters not specified as a segment in the path can be passed as an HTTP query string parameter:

```
curl -i "https://alyx.cortexlab.net/users/thomas/actions?type=experiment"
```

In this example, the `:username` parameter is in the path (with value `'thomas'`) while `:type` is passed in the query string.

For `POST`, `PATCH`, `PUT`, and `DELETE` requests, parameters not included in the URL should be encoded as JSON with a `Content-Type` of `'application/json'`.

Hypermedia and URL cross-links

All resources may have one or more `*_url` properties linking to other resources. These provide explicit URLs so that proper API clients don't need to construct URLs on their own. It is highly recommended that API clients use these. Doing so will make future upgrades of the API easier for developers. All URLs are expected to be proper [RFC 6570](#) URI templates.

Many of the resources on the users API provide a shortcut for getting information about the currently authenticated user. If a request URL does not include a `:username` parameter then the response will be for the logged in user.

Get all users

Returns list of users in summary representation. By default, this API is publicly accessible, to help automated login forms.

```
GET /users
```

Response:

```
[
  {
    "username": "thomas",
    "url": "https://alyx.cortexlab.net/subjects/thomas",
    "subjects_responsible": []
  },
  {
    "username": "chris",
    "url": "https://alyx.cortexlab.net/subjects/chris",
    "subjects_responsible":
      [
        "CMSubj104",
        "CMSubj105"
      ]
  }
]
```

Get a specific user

Returns specific user in detailed representation.

```
GET /users/:username
```

Response:

```
{
  "id": 4,
  "username": "chris",
  "first_name": "Chris",
  "last_name": "Burgess",
  "subjects_responsible": [
    "CMSubj104",
    "CMSubj105"
  ],
  "actions_url": "https://alyx.cortexlab.net/users/chris/actions",
  "email": "chris@cortexlab.net",
  "created_at": "2010-02-16T01:33:35Z",
  "updated_at": "2010-02-16T01:33:35Z",
}
```

List all actions for a user

Returns a list of actions performed by a user, in summary representation.

```
GET /users/chris/actions
```

Response:

```
[
  {
    "type": "surgery",
    "id": "660fd619-561c-416a-b528-7b0291d25378",
    "subject": "CBGADCre01",
    "url": "https://alyx.cortexlab.net/actions/660fd619-561c-416a-b528-
↪7b0291d25378",
    "start_date_time": "2016-05-12T14:45:01Z",
    "end_date_time": "2016-07-12T14:45:36Z",
    "tags": [
      "surgeon training",
      "3D printed implant"
    ],
    "location": "Room 120",
    "users": [
      "chris",
      "thomas"
    ],
    "procedures": [
      "Headplate",
      "Craniotomy"
    ]
  }
]
```

```

    "type": "experiment",
    "id": "134fd619-561c-416a-b528-7b0291d2541b",
    "subject": "CBGADCre01",
    "url": "https://alyx.cortexlab.net/actions/134fd619-561c-416a-b528-
↪7b0291d2541b",
    "start_date_time": "2016-03-12T14:45:10Z",
    "end_date_time": "2016-05-12T14:46:30Z",
    "tags": [
      "headfixed",
      "training session"
    ],
    "location": "BigRig",
    "users": [
      "chris"
    ],
    "procedures": []
  }
]

```

Parameters

Name	Type	Description
type	string	Defaults to all. Can be experiment, surgery, virus_injection, note, or any other Action.

List all subjects

Returns list of subjects in summary representation. By default, this will only return currently alive subjects belonging to the current logged-in user. Set `?alive=all&user=all` to see all subjects.

```
GET /subjects
```

Response:

```
[
  {
    "nickname": "CBGADCrel",
    "id": "ac6cfdad-e771-4222-9132-659d972a79dd",
    "responsible_user": "chris",
    "sex": "F",
    "birth_date": "2016-06-12",
    "death_date": null,
    "notes": "",
    "species": "Mus musculus",
    "litter": null,
    "strain": "C57BL/6J",
    "source": "SupplierACME",
    "cage": "Cage4010",
  },
  {
    "nickname": "M141010_CBECB",
    "id": "6a21e543-9f00-4fd2-ac0c-f1dd60830c1f",
    "responsible_user": "thomas",
    "sex": "U",
    "birth_date": null,
    "death_date": null,
    "notes": null,
    "species": null,
    "litter": null,
  }
]
```

```

    "strain": null,
    "source": null,
    "cage": null,
  },
]

```

Parameters

Name	Type	Description
alive	string	Can be one of dead or alive or all. Default: alive
user-name	string	Defaults to showing subjects for logged-in user. Specify this parameter to show a specific user's subjects, or all to show all users.

Get a specific subject

Returns specific subject in detailed representation.

```
GET /subjects/CBGADCre1
```

Response:

```

[
  {
    "nickname": "CBGADCre1",
    "id": "ac6cfdad-e771-4222-9132-659d972a79dd",
    "responsible_user": "chris",
    "sex": "F",
    "birth_date": "2016-06-12",
    "death_date": null,
    "notes": "",
    "species": "Mus musculus",
    "litter": null,
    "strain": "C57BL/6J",
    "source": "SupplierACME",
    "cage": "Cage4010",
    "genotype": [
      {
        "name": "Pvalb-Cre",
        "zygosity": "Heterozygous"
      },
      {
        "name": "Piezo2-cKO",
        "zygosity": "Homozygous"
      }
    ],
    "actions_url": "https://alyx.cortexlab.net/CBGADCre1/actions",
    "datasets_url": "https://alyx.cortexlab.net/CBGADCre1/datasets"
  }
]

```

List all actions for a subject

Returns a list of actions performed on a subject, in summary representation.

```
GET /subjects/CBGADCre1/actions
```

Response:

```
[
  {
    "type": "surgery",
    "id": "660fd619-561c-416a-b528-7b0291d25378",
    "subject": "CBGADCre01",
    "url": "https://alyx.cortexlab.net/actions/660fd619-561c-416a-b528-
↪7b0291d25378",
    "start_date_time": "2016-05-12T14:45:01Z",
    "end_date_time": "2016-07-12T14:45:36Z",
    "tags": [
      "surgeon training",
      "3D printed implant"
    ],
    "location": "Room 120",
    "users": [
      "chris",
      "thomas"
    ],
    "procedures": [
      "Headplate",
      "Craniotomy"
    ]
  },
  {
    "type": "experiment",
    "id": "134fd619-561c-416a-b528-7b0291d2541b",
    "subject": "CBGADCre01",
    "url": "https://alyx.cortexlab.net/actions/134fd619-561c-416a-b528-
↪7b0291d2541b",
    "start_date_time": "2016-03-12T14:45:10Z",
    "end_date_time": "2016-05-12T14:46:30Z",
    "tags": [
      "headfixed",
      "training session"
    ],
    "location": "BigRig",
    "users": [
      "chris"
    ],
    "procedures": []
  }
]
```

Parameters

Name	Type	Description
type	string	Defaults to all. Can be experiment, surgery, virus_injection, note, or any other Action.

List all datasets for a subject

This will list all the URLs of datasets acquired for a particular subject

By default, the summary representation is displayed. However, the 'detailed' representation can be requested; this saves time compared with listing all experiments and manually requesting the API to get all files multiple times.

```
GET /subjects/CBGADCre1/datasets
```

Response:

```
[
  {
    "id": "660fd619-561c-416a-b528-7b0291d25378",
    "url": "https://alyx.cortexlab.net/datasets/660fd619-561c-416a-b528-
↪7b0291d25378",
    "created_time": "2016-05-12T14:45:01Z",
    "modified_time": "2016-05-12T14:45:01Z",
    "md5": "qwertyu9wef0n",
    "filename": "my_imaging_a1345.tiff",
    "tags": [
      "uncompressed"
    ],
  },
  {
    "id": "110fd619-561c-416a-b528-7b0291d25123",
    "url": "https://alyx.cortexlab.net/datasets/110fd619-561c-416a-b528-
↪7b0291d25123",
    "created_time": "2015-05-12T14:45:01Z",
    "modified_time": "2015-05-12T14:45:01Z",
    "md5": "a9wfnap9w4fn2pfnoi",
    "filename": "my_imaging_aabcc1.tiff",
    "tags": [],
  }
]
```

Name	Type	Description
representa- tion	string	Defaults to 'summary'. Setting this to detailed fetches all filepaths, but will be slower.

List all actions

Returns all actions in summary representation. By default, this will only return actions by currently alive subjects belonging to the current logged-in user. Set `?alive=all&user=all` to see all subjects.

```
GET /actions
```

Response:

```
[
  {
    "type": "surgery",
    "id": "660fd619-561c-416a-b528-7b0291d25378",
    "subject": "CBGADCre01",
    "url": "https://alyx.cortexlab.net/actions/660fd619-561c-416a-b528-
↪7b0291d25378",
    "start_date_time": "2016-05-12T14:45:01Z",
    "end_date_time": "2016-07-12T14:45:36Z",
    "tags": [
      "surgeon training",
      "3D printed implant"
    ],
    "location": "Room 120",
    "users": [
      "chris",
      "thomas"
    ],
    "procedures": [
      "Headplate",
      "Craniotomy"
    ]
  }
]
```

```

    "id": "134fd619-561c-416a-b528-7b0291d2541b",
    "subject": "CBGADCre01",
    "url": "https://alyx.cortexlab.net/actions/134fd619-561c-416a-b528-
↪7b0291d2541b",
    "start_date_time": "2016-03-12T14:45:10Z",
    "end_date_time": "2016-05-12T14:46:30Z",
    "tags": [
      "headfixed",
      "training session"
    ],
    "location": "BigRig",
    "users": [
      "chris"
    ],
    "procedures": []
  }
]

```

Name	Type	Description
type	string	Defaults to all. Can be experiment, surgery, virus_injection, note, or any other Action.
subject	string	Defaults to showing subjects belonging to the logged-in user. Specify this parameter to show a specific subject, or all to show all subjects.

Get a specific action

Returns the detailed view of a specific action.

```
GET /actions/660fd619-561c-416a-b528-7b0291d25378
```

Response:

```

[
  {
    "type": "surgery",
    "id": "660fd619-561c-416a-b528-7b0291d25378",
    "subject": "CBGADCre01",
    "url": "https://alyx.cortexlab.net/actions/660fd619-561c-416a-b528-
↪7b0291d25378",
    "start_date_time": "2016-05-12T14:45:01Z",
    "end_date_time": "2016-07-12T14:45:36Z",
    "tags": [
      "surgeon training",
      "3D printed implant"
    ],
    "location": "Room 120",
    "users": [
      "chris",
      "thomas"
    ],
    "procedures": [
      "Headplate",
      "Craniotomy"
    ]
  }
]

```

```
}  
]
```

List all datasets

Returns list of datasets in summary representation

```
GET /datasets
```

Response:

```
[
  {
    "id": "660fd619-561c-416a-b528-7b0291d25378",
    "url": "https://alyx.cortexlab.net/datasets/660fd619-561c-416a-b528-
↪7b0291d25378",
    "created_time": "2016-05-12T14:45:01Z",
    "modified_time": "2016-05-12T14:45:01Z",
    "md5": "qwertyu9wef0n",
    "filename": "my_imaging_a1345.tiff",
    "tags": [
      "uncompressed"
    ],
  },
  {
    "id": "110fd619-561c-416a-b528-7b0291d25123",
    "url": "https://alyx.cortexlab.net/datasets/110fd619-561c-416a-b528-
↪7b0291d25123",
    "created_time": "2015-05-12T14:45:01Z",
    "modified_time": "2015-05-12T14:45:01Z",
    "md5": "a9wfnap9w4fn2pfnoi",
    "filename": "my_imaging_aabcc1.tiff",
    "tags": [],
  }
]
```

Get a specific dataset

Returns specific subject in detailed representation.

```
GET /datasets/660fd619-561c-416a-b528-7b0291d25378
```

Response:

```
[
  {
    "id": "660fd619-561c-416a-b528-7b0291d25378",
    "url": "https://alyx.cortexlab.net/datasets/660fd619-561c-416a-b528-
↪7b0291d25378",
    "created_time": "2016-05-12T14:45:01Z",
    "modified_time": "2016-05-12T14:45:01Z",
    "md5": "2016-05-12T14:45:01Z",
    "filename": "my_imaging_a1345.tiff",
    "filepaths": [
      {
        "type": "local",
        "hostname": "chris_pc",
        "path": "C:/DATA/CBGADCre1/2016-05-12/2photon/my_imaging_a1345.tiff",
      },
      {
        "type": "smb",
        "path": "\\dataserver.cortexlab.net\\subjects\\CBGADCre1\\2016-05-
↪12\\2photon\\my_imaging_a1345.tiff",
      },
      {
        "type": "tape_archive",
        "id": "CL0005",
      }
    ],
    "tags": [
      "uncompressed"
    ],
  }
]
```

These models generally correspond one-to-one with tables in the database.

Coordinate transforms

```
class misc.models.OrderedUser(id, password, last_login, is_superuser, username, first_name,  
                             last_name, email, is_staff, is_active, date_joined)
```

Bases: `django.contrib.auth.models.User`

Parameters

- **id** (*AutoField*) – Id
- **password** (*CharField*) – Password
- **last_login** (*DateTimeField*) – Last login
- **is_superuser** (*BooleanField*) – Designates that this user has all permissions without explicitly assigning them.
- **username** (*CharField*) – Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.
- **first_name** (*CharField*) – First name
- **last_name** (*CharField*) – Last name
- **email** (*EmailField*) – Email address
- **is_staff** (*BooleanField*) – Designates whether the user can log into this admin site.
- **is_active** (*BooleanField*) – Designates whether this user should be treated as active. Unselect this instead of deleting accounts.
- **date_joined** (*DateTimeField*) – Date joined
- **groups** (*ManyToManyField to Group*) – The groups this user belongs to. A user will get all permissions granted to each of their groups.

- **user_permissions** (ManyToManyField to `Permission`) – Specific permissions for this user.

class `misc.models.Note` (*id, json, user, date_time, text, content_type, object_id*)

Bases: `alyx.base.BaseModel`

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **user_id** (ForeignKey to *OrderedUser*) – User
- **date_time** (*DateTimeField*) – Date time
- **text** (*TextField*) – Text
- **content_type_id** (ForeignKey to *ContentType*) – Content type
- **object_id** (*UUIDField*) – Object id

class `misc.models.BrainLocation` (**args, **kwargs*)

Bases: `alyx.base.BaseModel`

Gives a brain location in stereotaxic coordinates, plus other information about location.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **name** (*CharField*) – Name
- **stereotaxic_coordinates** (*ArrayField*) – Stereotaxic coordinates
- **description** (*TextField*) – Description
- **allen_location_ontology** (*CharField*) – Allen location ontology

class `misc.models.CoordinateTransformation` (**args, **kwargs*)

Bases: `alyx.base.BaseModel`

This defines how to convert from a local coordinate system (e.g. of a silicon probe) to stereotaxic coordinates. It is an affine transformation: `stereotaxic_coordinates = origin + transformation_matrix*local_coordinates`. The description and `allen_location_ontology` apply to the coordinate origin (e.g. electrode tip).

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **name** (*CharField*) – Name
- **description** (*TextField*) – Description
- **allen_location_ontology** (*CharField*) – Allen location ontology
- **origin** (*ArrayField*) – Origin
- **transformation_matrix** (*ArrayField*) – Transformation matrix

Subjects

class `subjects.models.Subject` (*args, **kwargs)

Bases: `alyx.base.BaseModel`

Metadata about an experimental subject (animal or human).

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **nickname** (*CharField*) – Easy-to-remember, unique name (e.g. ‘Hercules’).
- **species_id** (*ForeignKey* to *Species*) – Species
- **litter_id** (*ForeignKey* to *Litter*) – Litter
- **sex** (*CharField* with choices: (('M', 'Male'), ('F', 'Female'), ('U', 'Unknown'))) – Sex
- **strain_id** (*ForeignKey* to *Strain*) – Strain
- **source_id** (*ForeignKey* to *Source*) – Source
- **line_id** (*ForeignKey* to *Line*) – Line
- **birth_date** (*DateField*) – Birth date
- **death_date** (*DateField*) – Death date
- **wean_date** (*DateField*) – Wean date
- **genotype_date** (*DateField*) – Genotype date
- **responsible_user_id** (*ForeignKey* to *OrderedUser*) – Who has primary or legal responsibility for the subject.
- **lamis_cage** (*IntegerField*) – Lamis cage
- **request_id** (*ForeignKey* to *SubjectRequest*) – Request
- **implant_weight** (*FloatField*) – Implant weight in grams
- **ear_mark** (*CharField*) – Ear mark
- **protocol_number** (*CharField* with choices: (('1', '1'), ('2', '2'), ('3', '3'), ('4', '4'))) – Protocol number
- **description** (*TextField*) – Description
- **cull_method** (*TextField*) – Cull method
- **adverse_effects** (*TextField*) – Adverse effects
- **actual_severity** (*IntegerField* with choices: ((None, ''), (1, 'Sub-threshold'), (2, 'Mild'), (3, 'Moderate'), (4, 'Severe'), (5, 'Non-recovery'))) – Actual severity
- **to_be_genotyped** (*BooleanField*) – To be genotyped
- **to_be_culled** (*BooleanField*) – To be culled
- **reduced** (*BooleanField*) – Reduced
- **reduced_date** (*DateField*) – Reduced date
- **genotype** (*ManyToManyField* to *Allele*) – Genotype
- **genotype_test** (*ManyToManyField* to *Sequence*) – Genotype test

class `subjects.models.SubjectRequest` (*id, json, user, line, count, date_time, due_date, description*)

Bases: `alyx.base.BaseModel`

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **user_id** (*ForeignKey to OrderedUser*) – Who requested this subject.
- **line_id** (*ForeignKey to Line*) – Line
- **count** (*IntegerField*) – Count
- **date_time** (*DateField*) – Date time
- **due_date** (*DateField*) – Due date
- **description** (*TextField*) – Description

`subjects.models.send_subject_request_mail_new` (*sender, instance=None, **kwargs*)
Send an email when a subject request is created.

`subjects.models.send_subject_request_mail_change` (*sender, instance=None, **kwargs*)
Send an email when a subject's request changes.

`subjects.models.send_subject_responsible_user_mail_change` (*sender, instance=None, **kwargs*)
Send an email when a subject's responsible user changes.

class `subjects.models.Litter` (**args, **kwargs*)

Bases: `alyx.base.BaseModel`

A litter, containing a mother, father, and children with a shared date of birth.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **descriptive_name** (*CharField*) – Descriptive name
- **line_id** (*ForeignKey to Line*) – Line
- **breeding_pair_id** (*ForeignKey to BreedingPair*) – Breeding pair
- **description** (*TextField*) – Description
- **birth_date** (*DateField*) – Birth date

class `subjects.models.BreedingPair` (*id, json, name, line, start_date, end_date, father, mother1, mother2, description*)

Bases: `alyx.base.BaseModel`

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **name** (*CharField*) – Leave to “-” to autofill.
- **line_id** (*ForeignKey to Line*) – Line
- **start_date** (*DateField*) – Start date
- **end_date** (*DateField*) – End date

- **father_id** (ForeignKey to *Subject*) – Father
- **mother1_id** (ForeignKey to *Subject*) – Mother1
- **mother2_id** (ForeignKey to *Subject*) – Mother2
- **description** (*TextField*) – Description

class `subjects.models.Line`(*id*, *json*, *name*, *description*, *target_phenotype*, *auto_name*, *strain*, *species*, *subject_autoname_index*, *breeding_pair_autoname_index*, *litter_autoname_index*, *is_active*)

Bases: `alyx.base.BaseModel`

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **name** (*CharField*) – Name
- **description** (*TextField*) – Description
- **target_phenotype** (*CharField*) – Target phenotype
- **auto_name** (*CharField*) – Auto name
- **strain_id** (ForeignKey to *Strain*) – Strain
- **species_id** (ForeignKey to *Species*) – Species
- **subject_autoname_index** (*IntegerField*) – Subject autoname index
- **breeding_pair_autoname_index** (*IntegerField*) – Breeding pair autoname index
- **litter_autoname_index** (*IntegerField*) – Litter autoname index
- **is_active** (*BooleanField*) – Is active
- **sequences** (ManyToManyField to *Sequence*) – Sequences

class `subjects.models.Species` (*args, **kwargs)

Bases: `alyx.base.BaseModel`

A single species, identified uniquely by its binomial name.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **binomial** (*CharField*) – Binomial name, e.g. “*mus musculus*”
- **display_name** (*CharField*) – common name, e.g. “mouse”

class `subjects.models.Strain` (*args, **kwargs)

Bases: `alyx.base.BaseModel`

A strain with a standardised name.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **descriptive_name** (*CharField*) – Standard descriptive name E.g. “C57BL/6J”, <http://www.informatics.jax.org/mgihome/nomen/>

- **description** (*TextField*) – Description

class `subjects.models.Source` (*args, **kwargs)

Bases: `alyx.base.BaseModel`

A supplier / source of subjects.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **name** (*CharField*) – Name
- **description** (*TextField*) – Description

class `subjects.models.StockManager` (*id, json, user*)

Bases: `alyx.base.BaseModel`

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **user_id** (*OneToOneField to User*) – User

class `subjects.models.Allele` (*args, **kwargs)

Bases: `alyx.base.BaseModel`

A single allele.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **standard_name** (*CharField*) – MGNC-standard genotype name e.g. Pvalb, <http://www.informatics.jax.org/mgihome/nomen/>
- **informal_name** (*CharField*) – informal name in lab, e.g. Pvalb-Cre

class `subjects.models.Zygotity` (*args, **kwargs)

Bases: `alyx.base.BaseModel`

A junction table between Subject and Allele.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **subject_id** (*ForeignKey to Subject*) – Subject
- **allele_id** (*ForeignKey to Allele*) – Allele
- **zygotity** (*IntegerField with choices: ((0, 'Absent'), (1, 'Heterozygous'), (2, 'Homozygous'), (3, 'Present'))*) – Zygotity

class `subjects.models.Sequence` (*args, **kwargs)

Bases: `alyx.base.BaseModel`

A genetic sequence that you run a genotyping test for.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **base_pairs** (*TextField*) – the actual sequence of base pairs in the test
- **description** (*CharField*) – any other relevant information about this test
- **informal_name** (*CharField*) – informal name in lab, e.g. ROSA-WT

class `subjects.models.GenotypeTest` (*id, json, subject, sequence, test_result*)

Bases: `alyx.base.BaseModel`

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **subject_id** (*ForeignKey* to *Subject*) – Subject
- **sequence_id** (*ForeignKey* to *Sequence*) – Sequence
- **test_result** (*IntegerField* with choices: ((0, 'Absent'), (1, 'Present'))) – Test result

TEST_RESULTS = ((0, 'Absent'), (1, 'Present'))

A junction table between Subject and Sequence.

Actions

class `actions.models.ProcedureType` (**args, **kwargs*)

Bases: `alyx.base.BaseModel`

A procedure to be performed on a subject.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **name** (*CharField*) – Short procedure name
- **description** (*TextField*) – Detailed description of the procedure

class `actions.models.Weighing` (**args, **kwargs*)

Bases: `alyx.base.BaseModel`

A weighing of a subject.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **user_id** (*ForeignKey* to *OrderedUser*) – The user who weighed the subject
- **subject_id** (*ForeignKey* to *Subject*) – The subject which was weighed
- **date_time** (*DateTimeField*) – Date time
- **weight** (*FloatField*) – Weight in grams
- **weighing_scale_id** (*ForeignKey* to *WeighingScale*) – The scale record that was used to weigh the subject

expected()

Expected weighing.

class `actions.models.WaterAdministration(*args, **kwargs)`

Bases: `alyx.base.BaseModel`

For keeping track of water for subjects not on free water.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **user_id** (ForeignKey to *OrderedUser*) – The user who administered water
- **subject_id** (ForeignKey to *Subject*) – The subject to which water was administered
- **date_time** (*DateTimeField*) – Date time
- **water_administered** (*FloatField*) – Water administered, in millilitres
- **hydrogel** (*NullBooleanField*) – Hydrogel

class `actions.models.BaseAction(*args, **kwargs)`

Bases: `alyx.base.BaseModel`

Base class for an action performed on a subject, such as a recording; surgery; etc. This should always be accessed through one of its subclasses.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **subject_id** (ForeignKey to *Subject*) – The subject on which this action was performed
- **location_id** (ForeignKey to *LabLocation*) – The physical location at which the action was performed
- **narrative** (*TextField*) – Narrative
- **start_time** (*DateTimeField*) – Start time
- **end_time** (*DateTimeField*) – End time
- **users** (ManyToManyField to *OrderedUser*) – The user(s) involved in this action
- **procedures** (ManyToManyField to *ProcedureType*) – The procedure(s) performed

class `actions.models.VirusInjection(*args, **kwargs)`

Bases: `actions.models.BaseAction`

A virus injection.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **subject_id** (ForeignKey to *Subject*) – The subject on which this action was performed
- **location_id** (ForeignKey to *LabLocation*) – The physical location at which the action was performed

- **narrative** (*TextField*) – Narrative
- **start_time** (*DateTimeField*) – Start time
- **end_time** (*DateTimeField*) – End time
- **virus_batch_id** (ForeignKey to *VirusBatch*) – Virus batch
- **injection_volume** (*FloatField*) – Volume in nanoliters
- **rate_of_injection** (*FloatField*) – TODO: Nanoliters per second / per minute?
- **injection_type** (CharField with choices: (('I', 'Iontophoresis'), ('P', 'Pressure'))) – Whether the injection was through iontophoresis or pressure
- **users** (ManyToManyField to *OrderedUser*) – The user(s) involved in this action
- **procedures** (ManyToManyField to *ProcedureType*) – The procedure(s) performed

class `actions.models.Surgery` (*args, **kwargs)

Bases: `actions.models.BaseAction`

Surgery performed on a subject.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **subject_id** (ForeignKey to *Subject*) – The subject on which this action was performed
- **narrative** (*TextField*) – Narrative
- **start_time** (*DateTimeField*) – Start time
- **end_time** (*DateTimeField*) – End time
- **brain_location_id** (ForeignKey to *BrainLocation*) – Brain location
- **outcome_type** (CharField with choices: (('a', 'Acute'), ('r', 'Recovery'))) – Outcome type
- **location_id** (ForeignKey to *LabLocation*) – The physical location at which the surgery was performed
- **users** (ManyToManyField to *OrderedUser*) – The user(s) involved in this action
- **procedures** (ManyToManyField to *ProcedureType*) – The procedure(s) performed

class `actions.models.Session` (*args, **kwargs)

Bases: `actions.models.BaseAction`

A recording or training session performed on a subject. There is normally only one of these per day, for example corresponding to a period of uninterrupted head fixation.

Note that you can organize sessions hierarchically by assigning a `parent_session`. Sub-sessions could for example correspond to periods of time in which the same neurons were recorded, or a particular set of stimuli were presented. Top-level sessions should have `parent_session` set to null.

If the fields (e.g. `users`) of a sub-session are null, they should be inherited from the parent.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way

- **subject_id** (ForeignKey to *Subject*) – The subject on which this action was performed
- **location_id** (ForeignKey to *LabLocation*) – The physical location at which the action was performed
- **narrative** (*TextField*) – Narrative
- **start_time** (*DateTimeField*) – Start time
- **end_time** (*DateTimeField*) – End time
- **parent_session_id** (ForeignKey to *Session*) – Hierarchical parent to this session
- **type** (*CharField*) – User-defined session type (e.g. Base, Experiment)
- **number** (*IntegerField*) – Optional session number for this level
- **users** (ManyToManyField to *OrderedUser*) – The user(s) involved in this action
- **procedures** (ManyToManyField to *ProcedureType*) – The procedure(s) performed

class `actions.models.WaterRestriction` (*args, **kwargs)

Bases: `actions.models.BaseAction`

Water restriction.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **subject_id** (ForeignKey to *Subject*) – The subject on which this action was performed
- **location_id** (ForeignKey to *LabLocation*) – The physical location at which the action was performed
- **narrative** (*TextField*) – Narrative
- **start_time** (*DateTimeField*) – Start time
- **end_time** (*DateTimeField*) – End time
- **users** (ManyToManyField to *OrderedUser*) – The user(s) involved in this action
- **procedures** (ManyToManyField to *ProcedureType*) – The procedure(s) performed

class `actions.models.OtherAction` (*args, **kwargs)

Bases: `actions.models.BaseAction`

Another type of action.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **subject_id** (ForeignKey to *Subject*) – The subject on which this action was performed
- **location_id** (ForeignKey to *LabLocation*) – The physical location at which the action was performed
- **narrative** (*TextField*) – Narrative
- **start_time** (*DateTimeField*) – Start time

- **end_time** (*DateTimeField*) – End time
- **users** (*ManyToManyField* to *OrderedUser*) – The user(s) involved in this action
- **procedures** (*ManyToManyField* to *ProcedureType*) – The procedure(s) performed

Equipment

class `equipment.models.LabLocation` (*args, **kwargs)

Bases: `alyx.base.BaseModel`

The physical location at which an session is performed or appliances are located. This could be a room, a bench, a rig, etc.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **name** (*CharField*) – Name

class `equipment.models.Supplier` (*args, **kwargs)

Bases: `alyx.base.BasePolymorphicModel`

A company or individual that provides lab equipment or supplies. This is a base class, to be accessed by subclasses

Parameters

- **polymorphic_ctype_id** (*ForeignKey* to *ContentType*) – Polymorphic ctype
- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **name** (*CharField*) – i.e. ‘NeuroNexus’
- **description** (*TextField*) – Description

class `equipment.models.EquipmentModel` (*args, **kwargs)

Bases: `alyx.base.BaseModel`

An equipment model. i.e. “BrainScanner 4X”

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **manufacturer_id** (*ForeignKey* to *Supplier*) – Manufacturer
- **model_name** (*CharField*) – e.g. ‘BrainScanner 4X’
- **description** (*CharField*) – Description

class `equipment.models.VirusBatch` (*args, **kwargs)

Bases: `alyx.base.BaseModel`

A virus batch

Parameters

- **id** (*UUIDField*) – Id

- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **virus_type** (*CharField*) – UPenn ID or equivalent
- **description** (*CharField*) – Description
- **virus_source_id** (ForeignKey to *Supplier*) – Who supplied the virus
- **date_time_made** (*DateTimeField*) – Date time made
- **nominal_titer** (*FloatField*) – TODO: What unit?

class `equipment.models.Appliance` (*args, **kwargs)

Bases: `alyx.base.BasePolymorphicModel`

An appliance, provided by a specific manufacturer. This class is only accessed through its subclasses.

Parameters

- **polymorphic_ctype_id** (ForeignKey to *ContentType*) – Polymorphic ctype
- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **location_id** (ForeignKey to *LabLocation*) – The physical location of the appliance.
- **equipment_model_id** (ForeignKey to *EquipmentModel*) – Equipment model
- **serial** (*CharField*) – The serial number of the appliance.
- **description** (*TextField*) – Description
- **descriptive_name** (*CharField*) – Descriptive name

class `equipment.models.WeighingScale` (*args, **kwargs)

Bases: `equipment.models.Appliance`

A weighing scale.

Parameters **appliance_ptr_id** (OneToOneField to *Appliance*) – Appliance ptr

class `equipment.models.LightSource` (*args, **kwargs)

Bases: `equipment.models.Appliance`

A light source (e.g. for use in optogenetics).

Parameters **appliance_ptr_id** (OneToOneField to *Appliance*) – Appliance ptr

class `equipment.models.Amplifier` (*args, **kwargs)

Bases: `equipment.models.Appliance`

An amplifier used in electrophysiology sessions.

Parameters **appliance_ptr_id** (OneToOneField to *Appliance*) – Appliance ptr

class `equipment.models.PipettePuller` (*args, **kwargs)

Bases: `equipment.models.Appliance`

A pipette puller for intracellular electrophysiology.

Parameters **appliance_ptr_id** (OneToOneField to *Appliance*) – Appliance ptr

class `equipment.models.DAQ` (*args, **kwargs)

Bases: `equipment.models.Appliance`

A DAQ for extracellular electrophysiology.

Parameters **appliance_ptr_id** (OneToOneField to *Appliance*) – Appliance ptr

Data

class `data.models.DataRepositoryType` (*args, **kwargs)

Bases: `alyx.base.BaseModel`

A type of data repository, e.g. local SAMBA file server; web archive; LTO tape

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **name** (*CharField*) – Name

class `data.models.DataRepository` (*args, **kwargs)

Bases: `alyx.base.BaseModel`

A data repository e.g. a particular local drive, specific cloud storage location, or a specific tape.

Stores an absolute path to the repository root as a URI (e.g. for SMB file://myserver.mylab.net/Data/ALF/; for web <https://www.neurocloud.edu/Data/>). Additional information about the repository can be stored in JSON in a type-specific manner (e.g. which cardboard box to find a tape in)

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **name** (*CharField*) – Name
- **repository_type_id** (*ForeignKey* to *DataRepositoryType*) – Repository type
- **path** (*CharField*) – absolute URI path to the repository

class `data.models.DatasetType` (*args, **kwargs)

Bases: `alyx.base.BaseModel`

A descriptor to accompany a dataset, saying what sort of information is contained in it E.g. “Neuropixels raw data” “eye camera movie”, etc.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **name** (*CharField*) – description of data type

class `data.models.BaseExperimentalData` (*args, **kwargs)

Bases: `alyx.base.BaseModel`

Abstract base class for all data acquisition models. Never used directly.

Contains an `Session` link, to provide information about who did the experiment etc. Note that sessions can be organized hierarchically, and this can point to any level of the hierarchy

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **session_id** (*ForeignKey* to *Session*) – The `Session` to which this data belongs
- **created_by_id** (*ForeignKey* to *User*) – The creator of the data.

- **created_datetime** (*DateTimeField*) – The creation datetime.
- **generating_software** (*CharField*) – e.g. ‘ChoiceWorld 0.8.3’
- **provenance_directory_id** (*ForeignKey* to *Dataset*) – link to directory containing intermediate results

class `data.models.Dataset` (*args, **kwargs)

Bases: `data.models.BaseExperimentalData`

A chunk of data that is stored outside the database, most often a rectangular binary array. There can be multiple *FileRecords* for one *Dataset*, which will be different physical files, all containing identical data, with the same MD5.

Note that by convention, binary arrays are stored as `.npy` and text arrays as `.tsv`

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **session_id** (*ForeignKey* to *Session*) – The Session to which this data belongs
- **created_by_id** (*ForeignKey* to *User*) – The creator of the data.
- **created_datetime** (*DateTimeField*) – The creation datetime.
- **generating_software** (*CharField*) – e.g. ‘ChoiceWorld 0.8.3’
- **provenance_directory_id** (*ForeignKey* to *Dataset*) – link to directory containing intermediate results
- **name** (*CharField*) – Name
- **md5** (*UUIDField*) – MD5 hash of the data buffer
- **dataset_type_id** (*ForeignKey* to *DatasetType*) – Dataset type

class `data.models.FileRecord` (*args, **kwargs)

Bases: `alyx.base.BaseModel`

A single file on disk or tape. Normally specified by a path within an archive. If required, more details can be in the JSON

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **dataset_id** (*ForeignKey* to *Dataset*) – Dataset
- **data_repository_id** (*ForeignKey* to *DataRepository*) – Data repository
- **relative_path** (*CharField*) – path name within repository

class `data.models.DataCollection` (*args, **kwargs)

Bases: `data.models.BaseExperimentalData`

A collection of datasets that all describe different aspects of the same objects. For example, the filtered and unfiltered waveforms of a set of spike clusters. Each file in the collection must have the same number of rows (or the same leading dimension for higher-dim arrays). The timeseries classes will inherit from this.

Parameters

- **id** (*UUIDField*) – Id

- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **session_id** (ForeignKey to *Session*) – The Session to which this data belongs
- **created_by_id** (ForeignKey to *User*) – The creator of the data.
- **created_datetime** (*DateTimeField*) – The creation datetime.
- **generating_software** (*CharField*) – e.g. ‘ChoiceWorld 0.8.3’
- **provenance_directory_id** (ForeignKey to *Dataset*) – link to directory containing intermediate results
- **name** (*CharField*) – description of the data in this collection (e.g. cluster information)
- **data** (ManyToManyField to *Dataset*) – Datasets, each of which should have their own descriptions and DatasetTypes

class `data.models.Timescale` (*args, **kwargs)

Bases: `alyx.base.BaseModel`

A timescale that is used to align recordings on multiple devices. There could be multiple timescales for a single experiment, which could be used for example if some information could only be aligned with poor temporal resolution.

However there can only be one timescale with the flag “final” set to True at any moment. This should reflect a final, accurate time alignment, that can be used by data analysts who do not need to understand how time alignment was performed. It should have a sample rate of 1.

When users search for data, they will normally search for a timescale that is linked to timeseries of the appropriate kind.

A timescale is always associated with a session, and can also optionally be associated with a series or experiment.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **name** (*CharField*) – informal name describing this field
- **nominal_start** (*DateTimeField*) – Approximate date and time corresponding to 0 samples
- **nominal_time_unit** (*FloatField*) – Nominal time unit for this timescale (in seconds)
- **final** (*BooleanField*) – set to true for the final results of time alignment, in seconds
- **info** (*CharField*) – any information, e.g. length of break around 300s inferred approximately from computer clock

class `data.models.TimeSeries` (*args, **kwargs)

Bases: `data.models.DataCollection`

A collection of Datasets that were all sampled together, associated with a single set of timestamps, relative to specified timescale. This is a DataCollection together with a timescale ID and a timestamps file.

In principle, you could store multiple TimeSeries with the same data but different Timescales. To avoid confusing data users, however, this is not recommended - only register the final timescale into the database, and don’t register intermediate timescales unless there is a good reason to.

Parameters

- **datacollection_ptr_id** (OneToOneField to *DataCollection*) – Datacollection ptr
- **timestamps_id** (ForeignKey to *Dataset*) – N*2 array containing sample numbers and their timestamps on associated timescale
- **timescale_id** (ForeignKey to *Timescale*) – which timescale this is on

class `data.models.EventSeries` (*args, **kwargs)

Bases: `data.models.DataCollection`

Links to a file containing a set of event times, and other files with further information, such as behavioral events or sensory stimuli. This is a DataCollection together with a times file and a timestamp ID.

Parameters

- **datacollection_ptr_id** (OneToOneField to *DataCollection*) – Datacollection ptr
- **timescale_id** (ForeignKey to *Timescale*) – which timescale this is on
- **times_id** (ForeignKey to *Dataset*) – n*1 array of times on specified timescale

class `data.models.IntervalSeries` (*args, **kwargs)

Bases: `data.models.BaseExperimentalData`

Links to a file containing a set of event times, and other files with further information, such as behavioral events or sensory stimuli. This is a DataCollection together with a times file and a timestamp ID.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **session_id** (ForeignKey to *Session*) – The Session to which this data belongs
- **created_by_id** (ForeignKey to *User*) – The creator of the data.
- **created_datetime** (*DateTimeField*) – The creation datetime.
- **generating_software** (*CharField*) – e.g. ‘ChoiceWorld 0.8.3’
- **provenance_directory_id** (ForeignKey to *Dataset*) – link to directory containing intermediate results
- **timescale_id** (ForeignKey to *Timescale*) – which timescale this is on
- **intervals_id** (ForeignKey to *Dataset*) – n*2 array of start and end times

Behavior

class `behavior.models.PupilTracking` (*args, **kwargs)

Bases: `data.models.BaseExperimentalData`

Describes the results of a pupil tracking algorithm.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **session_id** (ForeignKey to *Session*) – The Session to which this data belongs

- **created_by_id** (ForeignKey to `User`) – The creator of the data.
- **created_datetime** (`DateTimeField`) – The creation datetime.
- **x_y_d_id** (ForeignKey to `TimeSeries`) – n*3 timeseries giving x and y coordinates of center plus diameter
- **movie_id** (ForeignKey to `TimeSeries`) – Link to raw data
- **eye** (CharField with choices: (('L', 'Left'), ('R', 'Right'))) – Which eye was tracked; left or right
- **description** (`TextField`) – misc. narrative e.g. ('unit: mm' or 'unknown scale factor')
- **generating_software** (`CharField`) – e.g. 'PupilTracka 0.8.3'
- **provenance_directory_id** (ForeignKey to `Dataset`) – link to directory containing intermediate results

class `behavior.models.HeadTracking` (*args, **kwargs)

Bases: `data.models.BaseExperimentalData`

Describes the results of a head tracking algorithm.

Parameters

- **id** (`UUIDField`) – Id
- **json** (`JSONField`) – Structured data, formatted in a user-defined way
- **session_id** (ForeignKey to `Session`) – The Session to which this data belongs
- **created_by_id** (ForeignKey to `User`) – The creator of the data.
- **created_datetime** (`DateTimeField`) – The creation datetime.
- **x_y_theta_id** (ForeignKey to `TimeSeries`) – 3*n timeseries giving x and y coordinates of head plus angle
- **movie_id** (ForeignKey to `TimeSeries`) – Link to raw data
- **description** (`TextField`) – misc. narrative e.g. ('unit: cm' or 'unknown scale factor')
- **generating_software** (`CharField`) – e.g. 'HeadTracka 0.8.3'
- **provenance_directory_id** (ForeignKey to `Dataset`) – link to directory containing intermediate results

class `behavior.models.EventSeries` (*args, **kwargs)

Bases: `data.models.BaseExperimentalData`

Links to a file containing a set of event times and descriptions, such as behavioral events or sensory stimuli.

Parameters

- **id** (`UUIDField`) – Id
- **json** (`JSONField`) – Structured data, formatted in a user-defined way
- **session_id** (ForeignKey to `Session`) – The Session to which this data belongs
- **created_by_id** (ForeignKey to `User`) – The creator of the data.
- **created_datetime** (`DateTimeField`) – The creation datetime.

- **event_times_id** (ForeignKey to *Dataset*) – n*1 array of times in seconds (universal timescale)
- **type_descriptions_id_id** (ForeignKey to *Dataset*) – Type descriptions id
- **event_types_id_id** (ForeignKey to *Dataset*) – n*1 array listing the type of each event
- **description** (*TextField*) – misc. narrative e.g. ‘drifting gratings of different orientations’, ‘ChoiceWorld behavior events’
- **generating_software** (*CharField*) – e.g. ‘ChoiceWorld 0.8.3’
- **provenance_directory_id** (ForeignKey to *Dataset*) – link to directory containing intermediate results

class `behavior.models.IntervalSeries` (*args, **kwargs)

Bases: `data.models.BaseExperimentalData`

Links to a file containing a set of start/end pairs and descriptions, such as behavioral intervals or extended sensory stimuli.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **session_id** (ForeignKey to *Session*) – The Session to which this data belongs
- **created_by_id** (ForeignKey to *User*) – The creator of the data.
- **created_datetime** (*DateTimeField*) – The creation datetime.
- **interval_times_id** (ForeignKey to *Dataset*) – n*2 array, with associated array of row labels.
- **interval_types_id** (ForeignKey to *Dataset*) – n*1 array listing the type of each interval
- **type_descriptions_id** (ForeignKey to *Dataset*) – Type descriptions
- **description** (*TextField*) – misc. narrative e.g. ‘drifting gratings of different orientations’, ‘ChoiceWorld behavior intervals’
- **generating_software** (*CharField*) – e.g. ‘ChoiceWorld 0.8.3’
- **provenance_directory_id** (ForeignKey to *Dataset*) – link to directory containing intermediate results

class `behavior.models.OptogeneticStimulus` (*args, **kwargs)

Bases: `data.models.BaseExperimentalData`

This is a special type of interval series, to deal with optogenetic stimuli.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **session_id** (ForeignKey to *Session*) – The Session to which this data belongs
- **created_by_id** (ForeignKey to *User*) – The creator of the data.
- **created_datetime** (*DateTimeField*) – The creation datetime.
- **generating_software** (*CharField*) – e.g. ‘ChoiceWorld 0.8.3’

- **provenance_directory_id** (ForeignKey to *Dataset*) – link to directory containing intermediate results
- **apparatus_id** (ForeignKey to *Appliance*) – e.g. Laser that was used for stimulation.
- **light_delivery** (*CharField*) – e.g. ‘fiber pointed at craniotomy’
- **description** (*CharField*) – e.g. ‘square pulses’, ‘ramps’
- **wavelength** (*FloatField*) – in nm
- **brain_location_id** (ForeignKey to *BrainLocation*) – of fiber tip, craniotomy, etc.
- **stimulus_times_id** (ForeignKey to *Dataset*) – link to an n*2 array of start and stop of each pulse (sec)
- **stimulus_positions_id** (ForeignKey to *Dataset*) – link to an n*3 array of stimulus positions
- **power_id** (ForeignKey to *Dataset*) – link to an n*1 array giving each pulse power
- **power_calculation_method** (*CharField*) – TODO: normalize? measured, nominal
- **waveform_id** (ForeignKey to *Dataset*) – link to a file giving the waveform of each stimulus.?

class `behavior.models.Pharmacology` (*args, **kwargs)

Bases: `data.models.BaseExperimentalData`

Describes a drug application during the session.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **session_id** (ForeignKey to *Session*) – The Session to which this data belongs
- **created_by_id** (ForeignKey to *User*) – The creator of the data.
- **created_datetime** (*DateTimeField*) – The creation datetime.
- **generating_software** (*CharField*) – e.g. ‘ChoiceWorld 0.8.3’
- **provenance_directory_id** (ForeignKey to *Dataset*) – link to directory containing intermediate results
- **drug** (*CharField*) – TODO: normalize? Also say what it is dissolved in (DMSO etc)
- **administration_route** (*CharField*) – TODO: normalize? IP, IV, IM, surface etc...
- **start_time** (*FloatField*) – in seconds relative to session start. TODO: not DateTimeField? / TimeDifference
- **end_time** (*FloatField*) – equals start time if single application. TODO: should this be an offset? Or DateTimeField? Or TimeDifference?
- **concentration** (*CharField*) – TODO: not FloatField? include unit (e.g. g/kg; mM; %)
- **volume** (*CharField*) – TODO: not FloatField? include unit (e.g. µL)

Electrophysiology

class `electrophysiology.models.ProbeInsertion` (*args, **kwargs)

Bases: `alyx.base.BaseModel`

Contains info about the geometry and probe model of a single probe insertion.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **extracellular_recording_id** (*ForeignKey* to *ExtracellularRecording*) – id of extracellular recording
- **entry_point_rl** (*FloatField*) – mediolateral position of probe entry point relative to midline (microns). Positive means right
- **entry_point_ap** (*FloatField*) – anteroposterior position of probe entry point relative to bregma (microns). Positive means anterior
- **vertical_angle** (*FloatField*) – vertical angle of probe (degrees). Zero means horizontal. Positive means pointing down.
- **horizontal_angle** (*FloatField*) – horizontal angle of probe (degrees), after vertical rotation. Zero means anterior. Positive means counterclockwise (i.e. left).
- **axial_angle** (*FloatField*) – axial angle of probe (degrees). Zero means that without vertical and horizontal rotations, the probe contacts would be pointing up. Positive means counterclockwise.
- **distance_advanced** (*FloatField*) – How far the probe was moved forward from its entry point. (microns).
- **probe_model_id** (*ForeignKey* to *ProbeModel*) – model of probe used
- **channel_mapping_id** (*ForeignKey* to *Dataset*) – numerical array of size nSites x 1 giving the row of the raw data file for each contact site. You will have one of these files per probe, including if you record multiple probes through the same amplifier. Sites that were not recorded should have NaN or -1.

class `electrophysiology.models.ProbeModel` (*args, **kwargs)

Bases: `alyx.base.BaseModel`

Metadata describing each probe model

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **probe_manufacturer_id** (*ForeignKey* to *Supplier*) – Probe manufacturer
- **probe_model** (*CharField*) – manufacturer’s part number e.g. A4x8-5mm-100-200-177
- **description** (*CharField*) – optional informal description e.g. ‘Michigan 4x4 tetrode’; ‘Neuropixels phase 2 option 1’
- **site_positions_id** (*ForeignKey* to *Dataset*) – numerical array of size nSites x 2 giving locations of each contact site in local coordinates. Probe tip is at the origin.

class `electrophysiology.models.BaseBrainLocation` (*args, **kwargs)
 Bases: `alyx.base.BaseModel`

Abstract base class for brain location. Never used directly.

Contains curated anatomical location in Allen CCG with an acronym. This is usually figured out using histology, so should override what you might compute from `ProbeInsertion`.

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **ccf_ap** (*FloatField*) – Allen CCF antero-posterior coordinate (microns)
- **ccf_dv** (*FloatField*) – Allen CCF dorso-ventral coordinate (microns)
- **ccf_lr** (*FloatField*) – Allen CCF left-right coordinate (microns)
- **allen_ontology** (*CharField*) – Manually curated site location. Use Allen’s acronyms to represent the appropriate hierarchical level, e.g. SS, SSp, or SSp6a

class `electrophysiology.models.RecordingSite` (*args, **kwargs)
 Bases: `electrophysiology.models.BaseBrainLocation`

Contains estimated anatomical location of each recording site in each probe insertion. This is usually figured out using histology, so should override what you might compute from `ProbeInsertion`. Location a

Parameters

- **basebrainlocation_ptr_id** (*OneToOneField* to `BaseBrainLocation`) – Base-brainlocation ptr
- **probe_insertion_id** (*ForeignKey* to `ProbeInsertion`) – id of probe insertion
- **site_no** (*IntegerField*) – which site on the probe

class `electrophysiology.models.ExtracellularRecording` (*args, **kwargs)
 Bases: `data.models.TimeSeries`

Superclass of `TimeSeries` to describe raw data when you make an electrophys recording. There should a `Dataset` of `DatasetType` “`ephys.raw`” corresponding to this.

You can also link to a `Lfp` timeseries, that contains low-pass data at a lower sample rate with the same channel mapping

Parameters

- **timeseries_ptr_id** (*OneToOneField* to `TimeSeries`) – Timeseries ptr
- **lfp_id** (*ForeignKey* to `TimeSeries`) – lfp: low-pass filtered and downsampled
- **impedances_id** (*ForeignKey* to `Dataset`) – dataset containing measured impedance of each channel (ohms).
- **gains_id** (*ForeignKey* to `Dataset`) – dataset containing gain of each channel microvolts/bit
- **filter_info** (*CharField*) – Details of hardware corner frequencies, filter type, order.
- **recording_type** (*CharField* with choices: (('C', 'Chronic'), ('A', 'Acute'))) – Whether the recording is chronic or acute
- **ground_electrode** (*CharField*) – e.g. ‘screw above cerebellum’
- **reference_electrode** (*CharField*) – e.g. ‘shorted to ground’

- **amplifier_id** (ForeignKey to *Amplifier*) – The amplifier used in this recording.
- **daq_description_id** (ForeignKey to *DAQ*) – The DAQ used.

class `electrophysiology.models.SpikeSorting` (*args, **kwargs)

Bases: `data.models.EventSeries`

An entry in the *spike_sorting* table contains the output of a single spike sorting run of an extracellular recording. There will usually be only one of these per recording, but there could be several if you want to store multiple alternative clusterings.

This inherits from *EventSeries*, so is stored in the same format. As well as the spike times there should be an associated Dataset containing cluster IDs of each spike. Optionally, you can also have other datasets with implementation-specific information such as feature vectors but these are not standardized. Like all models derived from *BaseExperimentalData*, it also contain a *provenance_directory* that can contain these intermediate steps, in a non-standardized format.

Sometimes people do sortings per probe, sometimes per recording. The *probe_insertion* field should be null if it is a sorting for the whole recording. NOTE: to be strictly relational, if the *probe_insertion* is not null, the *extracellular_recording* ought to be.

Parameters

- **eventseries_ptr_id** (OneToOneField to *EventSeries*) – Eventseries ptr
- **extracellular_recording_id** (ForeignKey to *ExtracellularRecording*) – Extracellular recording
- **probe_insertion_id** (ForeignKey to *ExtracellularRecording*) – Probe insertion

class `electrophysiology.models.SpikeSortedUnit` (*args, **kwargs)

Bases: `electrophysiology.models.BaseBrainLocation`

This is going to be the biggest table, containing anatomical and other information on every unit resulting from spike sorting. (There is a separate table for units resulting from 2-photon).

Parameters

- **basebrainlocation_ptr_id** (OneToOneField to *BaseBrainLocation*) – Base-brainlocation ptr
- **cluster_number** (*IntegerField*) – Cluster number
- **spike_sorting_id** (ForeignKey to *SpikeSorting*) – The spike sorting this unit came from
- **channel_group** (*IntegerField*) – which shank this unit came from (an integer not a link)
- **trough_to_peak_width** (*FloatField*) – ms, computed from unfiltered mean spike waveform.
- **half_width** (*FloatField*) – ms, half width of negative peak in unfiltered spike waveform.
- **trough_to_peak_amplitude** (*FloatField*) – μV , from filtered spike waveform.
- **refractory_violation_rate** (*FloatField*) – fraction of spikes occurring $< 2\text{ms}$.
- **isolation_distance** (*FloatField*) – A measure of isolation quality
- **l_ratio** (*FloatField*) – A measure of isolation quality
- **mean_firing_rate** (*FloatField*) – spikes/s

- **cluster_group** (CharField with choices: (('0', 'Noise'), ('1', 'Multi-unit activity'), ('2', 'Single-unit activity'), ('3', 'Unsorted'))) – Human decision on cluster group
- **spike_width_class** (CharField with choices: (('N', 'Narrow'), ('W', 'Wide'))) – Human decision on spike width
- **optogenetic_response** (CharField) – e.g. 'Short latency' (only if applicable)
- **putative_cell_type** (CharField) – e.g. 'Sst interneuron', 'PT cell'.

class `electrophysiology.models.IntracellularRecording` (*args, **kwargs)
 Bases: `data.models.BaseExperimentalData`

This describes a single intracellular electrode used in one recording.

Parameters

- **id** (UUIDField) – Id
- **json** (JSONField) – Structured data, formatted in a user-defined way
- **session_id** (ForeignKey to `Session`) – The Session to which this data belongs
- **created_by_id** (ForeignKey to `User`) – The creator of the data.
- **created_datetime** (DateTimeField) – The creation datetime.
- **generating_software** (CharField) – e.g. 'ChoiceWorld 0.8.3'
- **provenance_directory_id** (ForeignKey to `Dataset`) – link to directory containing intermediate results
- **tip_location_id** (ForeignKey to `BrainLocation`) – Estimated location of probe tip
- **electrode_type** (CharField with choices: (('W', 'Whole-cell'), ('S', 'Sharp'))) – Electrode type
- **pipette_puller_id** (ForeignKey to `PipettePuller`) – Pipette puller
- **inner_diameter** (FloatField) – mm – before pulling
- **outer_diameter** (FloatField) – mm – before pulling
- **electrode_solution** (TextField) – Solution details.
- **cp_fast** (FloatField) – (pF)
- **cp_slow** (FloatField) – (pF)
- **whole_cell_cap_comp** (FloatField) – (pF)
- **whole_cell_series_resistance** (FloatField) – (Mohm)
- **series_resistance_compensation_bandwidth** (FloatField) – (kHz)
- **series_resistance_compensation_correction** (FloatField) – (%)
- **series_resistance_compensation_prediction** (FloatField) – (%)
- **recorded_current_id** (ForeignKey to `Dataset`) – nA. TODO: time series? flat file? sample rate?
- **voltage_command_id** (ForeignKey to `Dataset`) – mV
- **pipette_cap_comp** (FloatField) – (pF)
- **bridge_balance** (FloatField) – (M Ohm)

- **recorded_voltage_id** (ForeignKey to *Dataset*) – mV
- **current_command_id** (ForeignKey to *Dataset*) – nA
- **gain** (*FloatField*) – (V/V) – for info only; not required to convert raw data to volts

Imaging

class `imaging.models.SVDCompressedMovie` (*id, json, session, created_by, created_datetime, generating_software, provenance_directory, compressed_data_U, compressed_data_V*)

Bases: `data.models.BaseExperimentalData`

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **session_id** (ForeignKey to *Session*) – The Session to which this data belongs
- **created_by_id** (ForeignKey to *User*) – The creator of the data.
- **created_datetime** (*DateTimeField*) – The creation datetime.
- **generating_software** (*CharField*) – e.g. ‘ChoiceWorld 0.8.3’
- **provenance_directory_id** (ForeignKey to *Dataset*) – link to directory containing intermediate results
- **compressed_data_U_id** (ForeignKey to *Dataset*) – nSVs*nY*nX binary array giving normalized eigenframesSVD-compression eigenframes
- **compressed_data_V_id** (ForeignKey to *TimeSeries*) – nSamples*nSVs binary array SVD-compression timecourses

class `imaging.models.WidefieldImaging` (*id, json, session, created_by, created_datetime, generating_software, provenance_directory, raw_data, compressed_data, nominal_start_time, nominal_end_time, imaging_indicator, preprocessing, description, image_position, excitation_nominal_wavelength, recording_nominal_wavelength, excitation_device, recording_device*)

Bases: `data.models.BaseExperimentalData`

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **session_id** (ForeignKey to *Session*) – The Session to which this data belongs
- **created_by_id** (ForeignKey to *User*) – The creator of the data.
- **created_datetime** (*DateTimeField*) – The creation datetime.
- **generating_software** (*CharField*) – e.g. ‘ChoiceWorld 0.8.3’
- **provenance_directory_id** (ForeignKey to *Dataset*) – link to directory containing intermediate results
- **raw_data_id** (ForeignKey to *TimeSeries*) – pointer to nT by nX by nY by nC (colors) binary file

- **compressed_data_id** (ForeignKey to *SVDCompressedMovie*) – Link to SVD compressed movie, if compression was run
- **nominal_start_time** (*DateTimeField*) – in seconds relative to session start. TODO: not DateTimeField? / TimeDifference
- **nominal_end_time** (*DateTimeField*) – Equals start time if single application. TODO: should this be an offset? Or DateTimeField? Or TimeDifference?
- **imaging_indicator** (*CharField*) – . TODO: normalize!
- **preprocessing** (*CharField*) – e.g. ‘computed (F-F0) / F0, estimating F0 as running min’
- **description** (*CharField*) – e.g. ‘field of view includes V1, S1, retrosplenial’
- **image_position_id** (ForeignKey to *CoordinateTransformation*) – Image position
- **excitation_nominal_wavelength** (*FloatField*) – in nm. Can be array for multispectral
- **recording_nominal_wavelength** (*FloatField*) – in nm. Can be array for multispectral
- **excitation_device_id** (ForeignKey to *LightSource*) – Excitation device
- **recording_device** (*CharField*) – e.g. camera manufacturer, plus filter description etc. TODO: Appliance subclass - what name?

```
class imaging.models.TwoPhotonImaging(id, json, session, created_by, created_datetime,
                                       generating_software, provenance_directory, raw_data,
                                       compressed_data, description, image_position,
                                       excitation_wavelength, recording_wavelength,
                                       reference_stack)
```

Bases: *data.models.BaseExperimentalData*

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **session_id** (ForeignKey to *Session*) – The Session to which this data belongs
- **created_by_id** (ForeignKey to *User*) – The creator of the data.
- **created_datetime** (*DateTimeField*) – The creation datetime.
- **generating_software** (*CharField*) – e.g. ‘ChoiceWorld 0.8.3’
- **provenance_directory_id** (ForeignKey to *Dataset*) – link to directory containing intermediate results
- **raw_data_id** (ForeignKey to *TimeSeries*) – array of size nT by nX by nY by nZ by nC
- **compressed_data_id** (ForeignKey to *SVDCompressedMovie*) – to Compressed_movie, if compression was run
- **description** (*CharField*) – e.g. ‘V1 layers 2-4’
- **image_position_id** (ForeignKey to *CoordinateTransformation*) – Note if different planes have different alignment (e.g. flyback plane), this can’t be done in a single 3x3 transformation matrix, instead you would have an array of 3x2 matrices. TODO: how do we deal with this?

- **excitation_wavelength** (*FloatField*) – in nm
- **recording_wavelength** (*FloatField*) – in nm. Can be array for multispectral imaging. TODO: deal with arrays?
- **reference_stack_id** (*ForeignKey* to *Dataset*) – TODO: reference stack / BrainImage

class `imaging.models.ROIDetection` (*id, json, session, created_by, created_datetime, generating_software, provenance_directory, masks, plane, preprocessing, f, f0, two_photon_imaging_id*)

Bases: `data.models.BaseExperimentalData`

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **session_id** (*ForeignKey* to *Session*) – The Session to which this data belongs
- **created_by_id** (*ForeignKey* to *User*) – The creator of the data.
- **created_datetime** (*DateTimeField*) – The creation datetime.
- **generating_software** (*CharField*) – e.g. ‘ChoiceWorld 0.8.3’
- **provenance_directory_id** (*ForeignKey* to *Dataset*) – link to directory containing intermediate results
- **masks_id** (*ForeignKey* to *Dataset*) – array of size nROIs by nY by nX
- **plane_id** (*ForeignKey* to *Dataset*) – array saying which plane each roi is found in. TODO: is this an *ArrayField*? JSON?
- **preprocessing** (*CharField*) – computed $(F-F_0) / F_0$, estimating F_0 as running min’
- **f_id** (*ForeignKey* to *TimeSeries*) – array of size nT by nROIs giving raw fluorescence
- **f0_id** (*ForeignKey* to *TimeSeries*) – array of size nT by nROIs giving resting fluorescence
- **two_photon_imaging_id_id** (*ForeignKey* to *TwoPhotonImaging*) – 2P imaging stack.

class `imaging.models.ROI` (*id, json, session, created_by, created_datetime, generating_software, provenance_directory, roi_type, optogenetic_response, putative_cell_type, estimated_layer, roi_detection_id*)

Bases: `data.models.BaseExperimentalData`

Parameters

- **id** (*UUIDField*) – Id
- **json** (*JSONField*) – Structured data, formatted in a user-defined way
- **session_id** (*ForeignKey* to *Session*) – The Session to which this data belongs
- **created_by_id** (*ForeignKey* to *User*) – The creator of the data.
- **created_datetime** (*DateTimeField*) – The creation datetime.
- **generating_software** (*CharField*) – e.g. ‘ChoiceWorld 0.8.3’
- **provenance_directory_id** (*ForeignKey* to *Dataset*) – link to directory containing intermediate results
- **roi_type** (*CharField*) – soma, dendrite, neuropil, ...> TODO: normalize?

- **optogenetic_response** (*CharField*) – e.g. ‘Short latency’ (only if applicable)
- **putative_cell_type** (*CharField*) – e.g. ‘Sst interneuron’, ‘PT cell’
- **estimated_layer** (*CharField*) – e.g. ‘Layer 5b’
- **roi_detection_id_id** (ForeignKey to *ROIDetection*) – link to detection entry

CHAPTER 9

Indices and tables

- `genindex`
- `modindex`
- `search`

a

`actions.models`, 26

b

`behavior.models`, 35

d

`data.models`, 32

e

`electrophysiology.models`, 39

`equipment.models`, 30

i

`imaging.models`, 43

m

`misc.models`, 20

s

`subjects.models`, 22

A

actions.models (module), 26
Allele (class in subjects.models), 25
Amplifier (class in equipment.models), 31
Appliance (class in equipment.models), 31

B

BaseAction (class in actions.models), 27
BaseBrainLocation (class in electrophysiology.models),
39
BaseExperimentalData (class in data.models), 32
behavior.models (module), 35
BrainLocation (class in misc.models), 21
BreedingPair (class in subjects.models), 23

C

CoordinateTransformation (class in misc.models), 21

D

DAQ (class in equipment.models), 31
data.models (module), 32
DataCollection (class in data.models), 33
DataRepository (class in data.models), 32
DataRepositoryType (class in data.models), 32
Dataset (class in data.models), 33
DatasetType (class in data.models), 32

E

electrophysiology.models (module), 39
equipment.models (module), 30
EquipmentModel (class in equipment.models), 30
EventSeries (class in behavior.models), 36
EventSeries (class in data.models), 35
expected() (actions.models.Weighing method), 26
ExtracellularRecording (class in electrophysiol-
ogy.models), 40

F

FileRecord (class in data.models), 33

G

GenotypeTest (class in subjects.models), 26

H

HeadTracking (class in behavior.models), 36

I

imaging.models (module), 43
IntervalSeries (class in behavior.models), 37
IntervalSeries (class in data.models), 35
IntracellularRecording (class in electrophysiol-
ogy.models), 42

L

LabLocation (class in equipment.models), 30
LightSource (class in equipment.models), 31
Line (class in subjects.models), 24
Litter (class in subjects.models), 23

M

misc.models (module), 20

N

Note (class in misc.models), 21

O

OptogeneticStimulus (class in behavior.models), 37
OrderedUser (class in misc.models), 20
OtherAction (class in actions.models), 29

P

Pharmacology (class in behavior.models), 38
PipettePuller (class in equipment.models), 31
ProbeInsertion (class in electrophysiology.models), 39
ProbeModel (class in electrophysiology.models), 39
ProcedureType (class in actions.models), 26
PupilTracking (class in behavior.models), 35

R

RecordingSite (class in electrophysiology.models), 40

ROI (class in imaging.models), 45

ROIDetection (class in imaging.models), 45

S

send_subject_request_mail_change() (in module subjects.models), 23

send_subject_request_mail_new() (in module subjects.models), 23

send_subject_responsible_user_mail_change() (in module subjects.models), 23

Sequence (class in subjects.models), 25

Session (class in actions.models), 28

Source (class in subjects.models), 25

Species (class in subjects.models), 24

SpikeSortedUnit (class in electrophysiology.models), 41

SpikeSorting (class in electrophysiology.models), 41

StockManager (class in subjects.models), 25

Strain (class in subjects.models), 24

Subject (class in subjects.models), 22

SubjectRequest (class in subjects.models), 22

subjects.models (module), 22

Supplier (class in equipment.models), 30

Surgery (class in actions.models), 28

SVDCompressedMovie (class in imaging.models), 43

T

TEST_RESULTS (subjects.models.GenotypeTest attribute), 26

Timescale (class in data.models), 34

TimeSeries (class in data.models), 34

TwoPhotonImaging (class in imaging.models), 44

V

VirusBatch (class in equipment.models), 30

VirusInjection (class in actions.models), 27

W

WaterAdministration (class in actions.models), 27

WaterRestriction (class in actions.models), 29

Weighing (class in actions.models), 26

WeighingScale (class in equipment.models), 31

WidefieldImaging (class in imaging.models), 43

Z

Zygoty (class in subjects.models), 25