
owmeta Documentation

Release 0.12.3

owmeta

Dec 23, 2021

Contents

| | | |
|----------|--|-----------|
| 1 | owmeta | 3 |
| 1.1 | owmeta package | 3 |
| 2 | For Users | 17 |
| 2.1 | owmeta Data Sources | 17 |
| 2.2 | Requirements for data storage in OpenWorm | 19 |
| 2.3 | Adding Data to <i>YOUR</i> OpenWorm Database | 21 |
| 2.4 | Software Versioning | 24 |
| 2.5 | Python Release Compatibility | 24 |
| 3 | For Developers | 27 |
| 3.1 | Testing in owmeta | 27 |
| 3.2 | Adding documentation | 28 |
| 3.3 | owmeta coding standards | 29 |
| 4 | Issues | 31 |
| 5 | Indices and tables | 33 |
| | Python Module Index | 35 |
| | Index | 37 |

Our main README is available online on Github.¹ This documentation contains additional materials beyond what is covered there.

Contents:

¹ <http://github.com/openworm/owmeta>

1.1 owmeta package

1.1.1 owmeta

OpenWorm Unified Data Abstract Layer.

An introduction to owmeta can be found in the README on our [Github page](#).

1.1.2 Subpackages

owmeta.commands package

Various commands of the same kind as OWM, mostly intended as sub-commands of OWM.

Submodules

owmeta.commands.biology module

owmeta.data_trans package

Data translators

Some `DataSource` and `DataTranslator` types. Some deal with generic file types (e.g., comma-separated values) while others are specific to the format of a kind of file housed in owmeta.

Submodules

owmeta.data_trans.bibtex module

```
class owmeta.data_trans.bibtex.BibTexDataSource (bibtex_file_name, **kwargs)
    Bases:          owmeta.data_trans.common_data.DSMixin,          owmeta_core.data_trans.
                    local_file_ds.LocalFileDataSource

    File name [DatatypeProperty] Attribute: file_name
    Torrent file name [DatatypeProperty] Attribute: torrent_file_name
    MD5 hash [DatatypeProperty] Attribute: md5
    SHA-256 hash [DatatypeProperty] Attribute: sha256
    SHA-512 hash [DatatypeProperty] Attribute: sha512
    Input source [ObjectProperty] Attribute: source
        The data source that was translated into this one
    Transformation [ObjectProperty] Attribute: transformation
        Information about the transformation process that created this object
    Translation [ObjectProperty] Attribute: translation
        Information about the translation process that created this object
    Description [DatatypeProperty] Attribute: description
        Free-text describing the data source

class owmeta.data_trans.bibtex.BibTexDataTranslator (*args, **kwargs)
    Bases:          owmeta.data_trans.common_data.DTMixin,          owmeta_core.datasources.
                    DataTranslator

    Input type(s): BibTexDataSource
    Output type(s): EvidenceDataSource

    input_type
        alias of BibTexDataSource

    output_type
        alias of EvidenceDataSource

    translate()
        Notionally, this method takes one or more data sources, and translates them into some other data source
        that captures essentially the same information, but, possibly, in a different format. Additional sources can
        be passed in as well for auxiliary information which are not “translated” in their entirety into the output
        data source. Such auxiliary data sources should be distinguished from the primary ones in the translation

        Parameters

            *args Input data sources

            **kwargs Named input data sources

        Returns

            the output data source

class owmeta.data_trans.bibtex.EvidenceDataSource (*args, **kwargs)
    Bases:          owmeta.data_trans.common_data.DSMixin,          owmeta_core.datasources.
                    DataSource
```


Context [ObjectProperty] Attribute: evidence_context

The context

Input source [ObjectProperty] Attribute: source

The data source that was translated into this one

Transformation [ObjectProperty] Attribute: transformation

Information about the transformation process that created this object

Translation [ObjectProperty] Attribute: translation

Information about the translation process that created this object

Description [DatatypeProperty] Attribute: description

Free-text describing the data source

context_property

“Context”, a ObjectProperty: The context

owmeta.data_trans.common_data module

owmeta.data_trans.connections module

owmeta.data_trans.context_merge module

```
class owmeta.data_trans.context_merge.ContextMergeDataTranslator (*args,
                                                                    **kwargs)
    Bases: owmeta.data_trans.common_data.DTMixin, owmeta_core.datasource.
            DataTranslator
    Input type(s): owmeta_core.datasource.OneOrMore (DataWithEvidenceDataSource)
    Output type(s): DataWithEvidenceDataSource
    output_type
        alias of owmeta.data_trans.data_with_evidence_ds.DataWithEvidenceDataSource
    translate (*sources)
        Notionally, this method takes one or more data sources, and translates them into some other data source
        that captures essentially the same information, but, possibly, in a different format. Additional sources can
        be passed in as well for auxiliary information which are not “translated” in their entirety into the output
        data source. Such auxiliary data sources should be distinguished from the primary ones in the translation

    Parameters
        *args Input data sources
        **kwargs Named input data sources

    Returns
        the output data source
```

owmeta.data_trans.data_with_evidence_ds module

```
class owmeta.data_trans.data_with_evidence_ds.DataWithEvidenceDataSource (*args,  
                                                                    **kwargs)  
    Bases:      owmeta.data_trans.common_data.DSMixin,      owmeta_core.datasource.  
    DataSource
```

A data source that has an “evidence context” containing statements which support those in its “data context”. The data source also has a combined context which imports both the data and evidence contexts.

combined_context_property

“Combined context”, a `ObjectProperty`: Context importing both the data and evidence contexts

data_context_property

“Data context”, a `ObjectProperty`: The context in which primary data for this data source is defined

evidence_context_property

“Evidence context”, a `ObjectProperty`: The context in which evidence for the “Data context” is defined

owmeta.data_trans.neuron_data module

owmeta.data_trans.wormatlas module

owmeta.data_trans.wormbase module

1.1.3 Submodules

owmeta.bibtex module

```
owmeta.bibtex.bibtex_to_document (bibtex_entry, context=None)
```

Takes a single BibTeX entry and translates it into a *Document* object

```
owmeta.bibtex.load (bibtex_file)
```

Load BibTeX records from a file

Parameters

bibtex_file [*file object*] File containing one or more BibTeX records

Returns

bibtexparser.bibdatabase.BibDatabase Records represented in the string

```
owmeta.bibtex.load_from_file_named (file_name)
```

Loads from a file with the given name

Parameters

file_name [*str*] Name of the bibtex file to open

Returns

bibtexparser.bibdatabase.BibDatabase Records from the named file

```
owmeta.bibtex.loads (bibtex_string)
```

Load BibTeX records from a string

Parameters

bibtex_string [*str*] Text of one or more BibTeX records

Returns

bibtexparser.bibdatabase.BibDatabase Records represented in the string

owmeta.bibtex.parse_bibtex_into_documents (*file_name*, *context=None*)

Parses BibTeX records into a dictionary of *Document* instances

Parameters

bibtex_file [*file object*] File containing one or more BibTeX records

Returns

dict *Document* instances from the records in the file

owmeta.bibtex_customizations module

bibtexparser customizations

owmeta.bibtex_customizations.author (*record*)

Split author field by the string ‘and’ into a list of names.

Parameters

record [*dict*] the record

Returns

dict the given *record* with any updates applied

owmeta.bibtex_customizations.customizations (*record*)

Standard *owmeta bibtexparser* customizations

Includes: *url*, *note_url*, *doi*, *listify*, and *author*

Parameters

record [*dict*] the record

Returns

dict the given *record* with any updates applied

owmeta.bibtex_customizations.doi (*record*)

Adds a doi URI to the record if there’s a *doi* entry in the record

Parameters

record [*dict*] the record to update

Returns

dict the given *record* with any updates applied

owmeta.bibtex_customizations.listify (*record*)

Turns every value in the record into a list except for ENTRYTYPE and ID

owmeta.bibtex_customizations.listify_one (*record*, *name*)

If the given field *name* does not have a *list* value, then updates the record by turning that value into a list.

Parameters

record [*dict*] The record to update

name [*str*] The name of the field to turn into a list

Returns

dict the given record with any updates applied

`owmeta.bibtex_customizations.note_url(record)`

Extracts URLs from note entries in the given record

Parameters

record [**dict**] the record

Returns

dict the given record with any updates applied

`owmeta.bibtex_customizations.url(record)`

Merges any URL from `\url{...}` in `howpublished`, and any existing `link` or `url` values in the record and normalizes them into a **list** in the `url` field of the record

Parameters

record [**dict**] the record

Returns

dict the given record with any updates applied

owmeta.biology module

```
class owmeta.biology.BiologyType(**kwargs)
    Bases: owmeta_core.dataobject.DataObject
```

owmeta.cell module**owmeta.cell_common module****owmeta.channel module**

```
class owmeta.channel.Channel(name=None, **kwargs)
    Bases: owmeta.biology.BiologyType
```

A biological ion channel.

defined_augment()

This function must return **False** if `identifier_augment()` would raise an `IdentifierMissingException`. Override it when defining a non-standard identifier for subclasses of `DataObjects`.

identifier_augment()

Override this method to define an identifier in lieu of one explicitly set.

One must also override `defined_augment()` to return **True** whenever this method could return a valid identifier. `IdentifierMissingException` should be raised if an identifier cannot be generated by this method.

Raises

IdentifierMissingException

appearsIn

Cell types in which the ion channel has been expressed

description

A description of the ion channel

expression_pattern

A pattern of expression of this cell within an organism

gene_WB_ID

Wormbase ID of the encoding gene

gene_class

Classification of the encoding gene

gene_name

Name of the gene that codes for this ion channel

model

Get experimental models of this ion channel

models

Alias to *model*

name

Ion channel's name

proteins

Proteins associated with this channel

subfamily

Ion channel's subfamily

class `owmeta.channel.ExpressionPattern` (*wormbaseid=None, **kwargs*)

Bases: *owmeta.biology.BiologyType*

defined_augment()

This function must return False if *identifier_augment()* would raise an `IdentifierMissingException`. Override it when defining a non-standard identifier for subclasses of `DataObjects`.

identifier_augment()

Override this method to define an identifier in lieu of one explicitly set.

One must also override *defined_augment()* to return True whenever this method could return a valid identifier. `IdentifierMissingException` should be raised if an identifier cannot be generated by this method.

Raises

IdentifierMissingException

description

Natural language description of the expression pattern

wormbaseID

Alias to *wormbaseid*

wormbaseURL

The URL for the expression pattern in Wormbase

wormbaseid

The ID for the expression pattern in Wormbase

owmeta.channel_common module

`owmeta.channel_common.CHANNEL_RDF_TYPE = rdflib.term.URIRef('http://schema.openworm.org/2012/01/channel')`
Shared RDF type for channels

owmeta.channelworm module

class `owmeta.channelworm.ChannelModel` (*modelType=None, *args, **kwargs*)
Bases: `owmeta_core.dataobject.DataObject`

A model for an ion channel.

There may be multiple models for a single channel.

Example usage:

```
>>> from owmeta_core.quantity import Quantity

# Create a ChannelModel
>>> cm = PatchClampChannelModel(key='ca_boyle',
...     gating='voltage',
...     ion='Ca',
...     conductance=Quantity.parse('10pS'))
```

conductance

The conductance of this ion channel. This is the initial value, and should be entered as a Quantity object.

gating

The gating mechanism for this channel (“voltage” or name of ligand(s))

ion

The type of ion this channel selects for

modelType

The type of model employed to describe a channel

neuroML

Property for attaching NeuroML documents to resources

class `owmeta.channelworm.HomologyChannelModel` (***kwargs*)
Bases: `owmeta.channelworm.ChannelModel`

class `owmeta.channelworm.PatchClampChannelModel` (***kwargs*)
Bases: `owmeta.channelworm.ChannelModel`

class `owmeta.channelworm.PatchClampExperiment` (***kwargs*)
Bases: `owmeta.experiment.Experiment`

Store experimental conditions for a patch clamp experiment.

Ca_concentration

Calcium concentration

Cl_concentration

Chlorine concentration

blockers

Channel blockers used for this experiment

cell

The cell this experiment was performed on

cell_age
Age of the cell

initial_voltage
Starting voltage of the patch clamp

ion_channel
The ion channel being clamped

membrane_capacitance
Initial membrane capacitance

mutants
Type(s) of mutants being used in this experiment

patch_type
Type of patch clamp being used ('voltage' or 'current')

pipette_solution
Type of solution in the pipette

owmeta.cli_hints module**owmeta.command module****owmeta.connection module****owmeta.document module**

exception `owmeta.document.PubmedRetrievalException`
Bases: `Exception`

exception `owmeta.document.WormbaseRetrievalException`
Bases: `Exception`

class `owmeta.document.BaseDocument` (***kwargs*)
Bases: `owmeta_core.dataobject.DataObject`

class `owmeta.document.Document` (*bibtex=None, doi=None, pubmed=None, wormbase=None, **kwargs*)
Bases: `owmeta.document.BaseDocument`

A representation of some document.

Possible keys include:

```
pmid, pubmed: a pubmed id or url (e.g., 24098140)
wbid, wormbase: a wormbase id or url (e.g., WBPaper00044287)
doi: a Digital Object id or url (e.g., s00454-010-9273-0)
uri: a URI specific to the document, preferably usable for accessing
the document
```

Parameters

bibtex [`str`] A string containing a single BibTeX entry. Parsed during initialization, but not saved thereafter. optional

doi [`str`] A Digital Object Identifier (DOI). optional

pubmed [*str*] A PubMed ID (PMID) or URL that points to a paper. Ignored if 'pmid' is provided. optional

wormbase [*str*] An ID or URL from WormBase that points to a record. Ignored if *wbid* or *wormbaseid* are provided. optional

defined_augment()

This function must return False if *identifier_augment()* would raise an `IdentifierMissingException`. Override it when defining a non-standard identifier for subclasses of `DataObjects`.

identifier_augment()

Override this method to define an identifier in lieu of one explicitly set.

One must also override *defined_augment()* to return True whenever this method could return a valid identifier. `IdentifierMissingException` should be raised if an identifier cannot be generated by this method.

Raises

`IdentifierMissingException`

update_from_wormbase (*replace_existing=False*)

Queries wormbase for additional data to fill in the Document.

If *replace_existing* is set to `True`, then existing values will be cleared.

author

An author of the document

date

Alias to year

doi

A Digital Object Identifier (DOI), optional

pmid

A PubMed ID (PMID) that points to a paper

title

The title of the document

uri

A non-standard URI for the document

wbid

An ID from WormBase.org that points to a record, optional

wormbaseid

An alias to *wbid*

year

The year (e.g., publication year) of the document

owmeta.documentContext module

class `owmeta.documentContext.DocumentContext` (*document*)

Bases: `owmeta_core.context.Context`

A Context that corresponds to a document.


```
class owmeta.documentContext.DocumentContextMeta (name, bases, dct)
    Bases: owmeta_core.context.ContextMeta
```

owmeta.evidence module

```
exception owmeta.evidence.EvidenceError
    Bases: Exception
```

```
class owmeta.evidence.Evidence (**kwargs)
    Bases: owmeta_core.dataobject.DataObject
```

A representation which provides evidence, for a group of statements.

Attaching evidence to an set of statements is done like this:

```
>>> from owmeta.connection import Connection
>>> from owmeta.evidence import Evidence
>>> from owmeta_core.context import Context
```

Declare contexts:

```
>>> ACTX = Context(ident="http://example.org/data/some_statements")
>>> BCTX = Context(ident="http://example.org/data/some_other_statements")
>>> EVCTX = Context(ident="http://example.org/data/some_statements#evidence")
```

Make statements in ACTX and BCTX contexts:

```
>>> ACTX(Connection)(pre_cell="VA11", post_cell="VD12", number=3)
>>> BCTX(Connection)(pre_cell="VA11", post_cell="VD12", number=2)
```

In EVCTX, state that a that a certain document supports the set of statements in ACTX, but refutes the set of statements in BCTX:

```
>>> doc = EVCTX(Document)(author='White et al.', date='1986')
>>> EVCTX(Evidence)(reference=doc, supports=ACTX.rdf_object)
>>> EVCTX(Evidence)(reference=doc, refutes=BCTX.rdf_object)
```

Finally, save the contexts:

```
>>> ACTX.save_context()
>>> BCTX.save_context()
>>> EVCTX.save_context()
```

One note about the *reference* predicate: the reference should, ideally, be an unambiguous link to a peer-reviewed piece of scientific literature detailing methods and data analysis that supports the set of statements. However, in gather data from pre-existing sources, going to that level of specificity may be difficult due to deficient query capability at the data source. In such cases, a broader reference, such as a *Website* with information which guides readers to a peer-reviewed article supporting the statement is sufficient.

defined_augment()

This fuction must return `False` if `identifier_augment()` would raise an `IdentifierMissingException`. Override it when defining a non-standard identifier for sub-classes of `DataObjects`.

identifier_augment()

Override this method to define an identifier in lieu of one explicitly set.

One must also override `defined_augment()` to return True whenever this method could return a valid identifier. `IdentifierMissingException` should be raised if an identifier cannot be generated by this method.

Raises

`IdentifierMissingException`

reference

The resource providing evidence supporting/refuting the attached context

refutes

A context naming a set of statements which are refuted by the attached reference

supports

A context naming a set of statements which are supported by the attached reference

`owmeta.evidence.evidence_for(qctx, ctx, evctx=None)`

Returns an iterable of Evidence

Parameters

qctx [`object`] an object supported by evidence. If the object is a `Context` with no identifier, then the query considers statements ‘staged’ (rather than stored) in the context

ctx [`Context`] Context that bounds where we look for statements about `qctx`. The contexts for statements found in this context are the actual targets of `Evidence.supports` statements.

evctx [`Context`] if the `Evidence.supports` statements should be looked for somewhere other than `ctx`, that can be specified in `evctx`. optional

`owmeta.evidence.query_context(graph, qctx)`

graph [`rdflib.graph.Graph`] Graph where we can find the contexts for statements in `qctx`

qctx [`owmeta.context.Context`] Container for statements

owmeta.experiment module

class `owmeta.experiment.Experiment(**kwargs)`

Bases: `owmeta_core.dataobject.DataObject`

Generic class for storing information about experiments

Should be overridden by specific types of experiments (example: see `PatchClampExperiment` in `channel-worm.py`).

Overriding classes should have a list called “conditions” that contains the names of experimental conditions for that particular type of experiment. Each of the items in “conditions” should also be either a `DatatypeProperty` or `ObjectProperty` for the experiment as well.

get_conditions()

Return conditions and their associated values in a dict.

reference

Supporting article for this experiment.

owmeta.muscle module

owmeta.my_neuroml module

owmeta.network module

owmeta.neuroml module

class owmeta.neuroml.**NeuroMLDocument** (***kwargs*)

Bases: owmeta_core.dataobject.DataObject

Represents a NeuroML document

The document may be represented literally in the RDF graph using `xml_content` or stored elsewhere and included by reference with `document_url`.

Example:

```

>>> embedded_nml = NeuroMLDocument(key='embedded_ex', content="""\
... <?xml version="1.0" encoding="UTF-8"?>
... <neuroml xmlns="http://www.neuroml.org/schema/neuroml2"
...     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
...     xsi:schemaLocation="http://www.neuroml.org/schema/neuroml2
...     https://raw.githubusercontent.com/NeuroML/NeuroML2/master/Schemas/NeuroML2/NeuroML_
...     v2beta.xsd"
...     id="k_slow">
...     <ionChannel id="k_slow" conductance="10pS" type="ionChannelHH" species="k
...     ">
...         <notes>K slow channel from Boyle and Cohen 2008</notes>
...         <gateHHtauInf id="n" instances="1">
...             <timeCourse type="fixedTimeCourse" tau="25.0007 ms"/>
...             <steadyState type="HHSigmoidVariable" rate="1" scale="15.8512 mV"
...             midpoint="19.8741 mV"/>
...         </gateHHtauInf>
...     </ionChannel>
... </neuroml>""")

>>> external_nml = NeuroMLDocument(ident='external_ex',
...     document_url='')

```

content

XML content for the document. Should be a complete NeuroML document rather than a fragment.

document_url

URL where the XML content of the document can be retrieved

class owmeta.neuroml.**NeuroMLProperty** (*resolver, *args, **kwargs*)

Bases: owmeta_core.dataobject_property.ObjectProperty

Property for attaching NeuroML documents to resources

value_type

alias of *NeuroMLDocument*

owmeta.neuron module

owmeta.plot module

class owmeta.plot.**Plot** (*data=None, *args, **kwargs*)

Bases: owmeta_core.dataobject.DataObject

Object for storing plot data in owmeta.

Parameters

data [2D list (list of lists)] List of XY coordinates for this Plot.

Example usage ::

```
>>> p1 = Plot([[1, 2], [3, 4]])
>>> p1.get_data()
# [[1, 2], [3, 4]]
```

get_data()

Get the data stored for this plot.

set_data(data)

Set the data attribute, which is user-facing, as well as the serialized `_data_string` attribute, which is used for db storage.

owmeta.sources module

owmeta.translators module

owmeta.utils module

Common utilities for translation, massaging data, etc., that don't fit elsewhere in owmeta

owmeta.website module

class owmeta.website.**Website** (*title=None, **kwargs*)

Bases: *owmeta.document.BaseDocument*

A representation of a website

defined_augment()

This fuction must return False if *identifier_augment()* would raise an IdentifierMissingException. Override it when defining a non-standard identifier for subclasses of DataObjects.

identifier_augment()

Override this method to define an identifier in lieu of one explicitly set.

One must also override *defined_augment()* to return True whenever this method could return a valid identifier. IdentifierMissingException should be raised if an identifier cannot be generated by this method.

Raises

IdentifierMissingException

title

The official name for the website

url

A URL for the website

owmeta.worm module

owmeta.worm_common module

2.1 owmeta Data Sources

The sources of data for owmeta are stored in the [OpenWormData repository](#). A few `DataTranslators` translate these data into common owmeta data sources. You can list these by running:

```
owm source list
```

and you can show some of the properties of a data source by running:

```
owm source show $SOURCE_IDENTIFIER
```

For instance, you can run the following to see the top-level data source, try:

```
owm source show http://openworm.org/data
```

This will print out summary descriptions of the sources that contribute to the main data source.

2.1.1 A Note on owmeta Data

Below, each major element of the worm's anatomy that owmeta stores data on is considered individually. The data being used is tagged by source in a superscript, and the decisions made during the curation process (if any) are described.

2.1.2 Neurons

- Neuron names²: Extracted from WormBase. Dynamic version on [this google spreadsheet](#). Staged in [this csv file](#). Parsed by [this method](#).

²

– Harris, T. W., Antoshechkin, I., Bieri, T., Blasiar, D., Chan, J., Chen, W. J., ... Sternberg, P. W. (2010). WormBase: a comprehensive resource for nematode research. *Nucleic Acids Research*, 38(Database issue), D463–7. <http://doi.org/10.1093/nar/gkp952>

– Lee, R. Y. N., & Sternberg, P. W. (2003). Building a cell and anatomy ontology of *Caenorhabditis elegans*. *Comparative and Functional*

- Neuron types¹: Extracted from WormAtlas.org. Staged in [this csv file](#). Parsed by [this method](#).
- Cell descriptions¹: Extracted from WormAtlas.org. Staged in [this tsv file](#). Parsed by [this method](#).
- Lineage names¹: Extracted from WormAtlas.org. Dynamic version on [this google spreadsheet](#). Staged in [this tsv file](#). Parsed by [this method](#).
- Neurotransmitters¹: Extracted from WormAtlas.org. Dynamic version on [this google spreadsheet](#). Staged in [this csv file](#). Parsed by [this method](#).
- Neuropeptides¹: Extracted from WormAtlas.org. Dynamic version on [this google spreadsheet](#). Staged in [this csv file](#). Parsed by [this method](#).
- Receptors¹: Extracted from WormAtlas.org. Dynamic version on [this google spreadsheet](#). Staged in [this csv file](#). Parsed by [this method](#).
- Innexins¹: Extracted from WormAtlas.org. Dynamic version on [this google spreadsheet](#). Staged in [this csv file](#). Parsed by [this method](#).

Gene expression data below, additional to that extracted from WormAtlas concerning receptors, neuropeptides, neurotransmitters and innexins are parsed by [this method](#):

- Monoamine secretors and receptors, neuropeptide secretors and receptors⁴: Dynamic version on [this google spreadsheet](#). Staged in [this csv file](#).

2.1.3 Muscle cells

- Muscle names²: Extracted from WormBase. Dynamic version on [this google spreadsheet](#). Staged in [this csv file](#). Parsed by [this method](#).
- Cell descriptions¹: Extracted from WormAtlas.org. Dynamic version on [this google spreadsheet](#). Staged in [this tsv file](#). Parsed by [this method](#).
- Lineage names¹: Extracted from WormAtlas.org. Dynamic version on [this google spreadsheet](#). Staged in [this tsv file](#). Parsed by [this method](#).
- Neurons that innervate each muscle³: Extracted from data personally communicated by S. Cook. Staged in [this csv file](#). Parsed by [this method](#).

2.1.4 Connectome

- Gap junctions between neurons³: Extracted from data personally communicated by S. Cook. Staged in [this csv file](#). Parsed by [this method](#).
- Synapses between neurons³: Extracted from data personally communicated by S. Cook. Staged in [this csv file](#). Parsed by [this method](#).

Genomics, 4(1), 121–6. <http://doi.org/10.1002/cfg.248>

¹ Altun, Z.F., Herndon, L.A., Wolkow, C.A., Crocker, C., Lints, R. and Hall, D. H. (2015). WormAtlas. Retrieved from <http://www.wormatlas.org> - WormAtlas Complete Cell List

⁴ Bentley B., Branicky R., Barnes C. L., Chew Y. L., Yemini E., Bullmore E. T., Vertes P. E., Schafer W. R. (2016) The Multilayer Connectome of *Caenorhabditis elegans*. PLoS Comput Biol 12(12): e1005283. <http://doi.org/10.1371/journal.pcbi.1005283>

³ Emmons, S., Cook, S., Jarrell, T., Wang, Y., Yakolev, M., Nguyen, K., Hall, D. Whole-animal *C. elegans* connectomes. C. Elegans Meeting 2015 <http://abstracts.genetics-gsa.org/cgi-bin/celegans15s/wsrch15.pl?author=emmons&sort=ptimes&sbutton=Detail&absno=155110844&sid=668862>

Curation note

There was another source of *C. elegans* connectome data that was created by members of the OpenWorm project that has since been retired. The history of this spreadsheet is mostly contained in [this forum post](#). We decided to use the Emmons data set³ as the authoritative source for connectome data, as it is the very latest version and updated version of the *C. elegans* connectome that we are familiar with.

2.1.5 Data Source References

2.2 Requirements for data storage in OpenWorm

Our OpenWorm database captures facts about *C. elegans*. The database stores data for generating model files and together with annotations describing the origins of the data. Below are a set of recommendations for implementation of the database organized around an RDF (Resource Description Framework) model.

2.2.1 Interface

Access is through a Python library which communicates with the database. This library serves the function of providing an object oriented view on the database that can be accessed through the Python scripts commonly used in the project. The *api* is described separately.

2.2.2 Data modeling

Biophysical and anatomical data are included in the database. A sketch of some features of the data model is below. Also included in our model are the relationships between these types. Given our choice of data types, we do not model the individual interactions between cells as entities in the database. Rather these are described by generic predicates in an [RDF triple](#). For instance, neuron A synapsing with muscle cell B would give a statement (A, synapsesWith, B), but A synapsing with neuron C would also have (A, synapsesWith, C).

Nervous system

For the worm's nervous system, we capture a few important data types (listed *below*). These correspond primarily to the anatomical structures and chemicals which are necessary for the worm to record external and internal stimuli and activate its body in response to those stimuli.

Data types

A non-exhaustive list of neurological data types in our *C. elegans* database:

- receptor types identified in the nerve cell
- neurons
- ion channels
- neurotransmitters
- muscle receptors

Development

C. elegans has very stable cell division patterns in the absence of mutations. This means that we can capture divisions in our database as static ‘daughterOf’ relationships. The theory of differentiation codes additionally gives an algorithmic description to the growth patterns of the worm which describes signals transmitted between developing cells. In order to test this theory we would like to leverage existing photographic data indicating the volume of cells at the time of their division as this relates to the differentiation code stored by the cell. Progress on this issue is documented on [GitHub](#)

Aging

Concurrently with development, we would like to begin modeling the effects of aging on the worm. Aging typically manifests in physiological changes due to transcription errors or cell death. These physiological changes can be represented abstractly as parameters to the function of biological entities. See [GitHub](#) for further discussion.

2.2.3 Information assurance

Provenance

Tracking the origins of facts stated in the database demands a method of annotating statements in our database. Providing citations for facts must be as simple as providing a global identifier (e.g., URI, DOI) or a local identifier (e.g., Bibtext identifier, Pubmed ID). With owmeta, supporting information can be attached to [named graphs](#), which are groupings of statements with a URI attached to them. A named graph can have as many or as few statements as desired. Furthermore, a given triple can occur in multiple named graphs. Further details for the attachment of evidence using this technique are given in the [api](#).

In line with current practices for communication through the source code management platform, GitHub, we track responsibility for new uploads to the database through the [OpenWormData](#) Git repository. Each named graph is canonicalized – essentially, triples are sorted and written to a text file – and committed to a Git repository which gives us, at least, an email address and a timestamp for all modifications.

Access control

Data in owmeta are distributed as a bundle, a packaging structure which contains a set of canonicalized named graphs and, optionally, some files. Responsibility for restricting who can modify a bundle is, in the first instance, up to the bundle creator. When the bundle is actually distributed, the responsibility then falls on the distributor to ensure authentication of the bundle’s provider and integrity of the bundle.

In OpenWorm, we create bundles from the OpenWormData GitHub repository. Access to the repository is managed by senior OpenWorm contributors. Bundles are deployed to Google Drive with write access controlled by Mark Watts. You can fetch OpenWorm bundles by adding a remote like this:

```
owm bundle remote add google-drive 'https://drive.google.com/uc?
↳id=1NYAcKdcvoFu5c7Nz3l4hK5UacG_eD56V&authuser=0&export=download'
```

google-drive can be substituted with any string.

2.2.4 Miscellaneous

Versioning

Experimental methods are constantly improving in biological research. These improvements may require updating the data we reference or store internally. However, in making updates we must not immediately expunge older content, breaking links created by internal and external agents. Instead, we utilize bundle versioning to track revisions to the data. Each successive release of the bundle increments the bundle version number.

2.2.5 Why RDF?

RDF offers advantages in resilience to schema additions and increased flexibility in integrating data from disparate sources.¹ These qualities can be valued by comparison to relational database systems. Typically, schema changes in a relational database require extensive work for applications using it.² In the author's experience, RDF databases offer more freedom in restructuring. Also, for data integration, SPARQL, the standard language for querying over RDF has [Federated queries](#) which allow for nearly painless integration of external SPARQL endpoints with existing queries.

The advantage of local storage of the database that goes along with each copy of the library is that the data will have the version number of the library. This means that data can be 'deprecated' along with a deprecated version of the library. This also will prevent changes made to a volatile database that break downstream code that uses the library.

2.3 Adding Data to *YOUR* OpenWorm Database

So, you've got some biological data about the worm and you'd like to save it in owmeta, but you don't know how it's done?

You've come to the right place!

A few biological entities (e.g., Cell, Neuron, Muscle, Worm) are pre-coded into owmeta. The full list is available in the [API](#). If these entities already cover your use-case, then all you need to do is add values for the appropriate fields and save them. If you have data already loaded into your database, then you can load objects from it:

```
>>> from owmeta.neuron import Neuron
>>> n = Neuron.query()
>>> n.receptor('UNC-13')
owmeta_core.statement.Statement(...obj=owmeta_core.dataobject_property.
↳ ContextualizedPropertyValue(rdflib.term.Literal(u'UNC-13')), context=None)
>>> for x in n.load():
...     do_something_with_unc13_neuron(n) # doctest.SKIP
```

If you need additional entities it's easy to create them. Documentation for this is provided [here](#).

Typically, you'll want to attach the data that you insert to entities already in the database. This allows you to recover objects in a hierarchical fashion from the database later. Worm, for instance, has a property, `neuron_network`, which points to the `Network` which should contain all neural cells and synaptic connections. To initialize the hierarchy you would do something like:

```
>>> from owmeta_core.context import Context
>>> from owmeta.worm import Worm
>>> from owmeta.network import Network
>>> ctx = Context('http://example.org/c-briggsae')
>>> w = ctx(Worm)('C. briggsae') # The name is optional and currently defaults to 'C.
↳ elegans'
>>> nn = ctx(Network)() # make a neuron network
```

(continues on next page)

¹ <http://answers.semanticweb.com/questions/19183/advantages-of-rdf-over-relational-databases>

² <http://research.microsoft.com/pubs/118211/andy%20maule%20-%20thesis.pdf>

(continued from previous page)

```

>>> w.neuron_network(nn)          # attach to the worm the neuron network
owmeta_core.statement.Statement(...)
>>> n = ctx(Neuron)('NeuronX')    # make a neuron
>>> n.receptor('UNC-13')          # state that the neuron has a UNC-13 type receptor
owmeta_core.statement.Statement(...)
>>> nn.neuron(n)                  # attach to the neuron network
owmeta_core.statement.Statement(...)
>>> ctx.save()                    # save all of the data attached to the worm

```

It is possible to create objects without attaching them to anything and they can still be referenced by calling `load()` on an instance of the object's class as in `n.load()` above. This also points out another fact: you don't have to set up the hierarchy for each insert in order for the objects to be linked to existing entities. If you have previously set up connections to an entity (e.g., `Worm('C. briggsae')`), assuming you *only* have one such entity, you can refer to things attached to it without respecifying the hierarchy for each script. The database packaged with owmeta should have only one Worm and one Network.

Remember that once you've made all of the statements, you must save the context in which the statements are made.

Future capabilities:

- Adding propositional logic to support making statements about all entities matching some conditions without needing to `load()` and `save()` them from the database.
- Statements like:

```

ctx = Context('http://example.org/c-briggsae')
w = ctx.stored(Worm)()
w.neuron_network.neuron.receptor('UNC-13')
l = list(w.load()) # Get a list of worms with neurons expressing 'UNC-13'

```

currently, to do the equivalent, you must work backwards, finding all neurons with UNC-13 receptors, then getting all networks with those neurons, then getting all worms with those networks:

```

worms = set()
n = ctx.stored(Neuron)()
n.receptor('UNC-13')
for ns in n.load():
    nn = ctx.stored(Network)()
    nn.neuron(ns)
    for z in nn.load():
        w = ctx.stored(Worm)()
        w.neuron_network(z)
        worms.add(w)
l = list(worms)

```

It's not difficult logic, but it's 8 extra lines of code for a, conceptually, very simple query.

- Also, queries like:

```

l = list(ctx.stored(Worm)('C. briggsae').neuron_network.neuron.receptor()) # get
↳ a list
# of all receptors expressed in neurons of C. briggsae

```

Again, not difficult to write out, but in this case it actually gives a much longer query time because additional values are queried in a `load()` call that are never returned.

We'd also like operators for composing many such strings so:

```
ctx.stored(Worm)('C. briggsae').neuron_network.neuron.get('receptor', 'innexin')
↪ # list
#of (receptor, innexin) values for each neuron
```

would be possible with one query and thus not requiring parsing and iterating over neurons twice—it's all done in a single, simple query.

2.3.1 Contexts

Above, we used contexts without explaining them. In natural languages, our statements are made in a context that influences how they should be interpreted. In owmeta, that kind of context-sensitivity is modeled by using `owmeta.Context` objects. To see what this looks like, let's start with an example.

Basics

I have data about widgets from BigDataWarehouse (BDW) that I want to translate into RDF using owmeta, but I don't want put them with my other widget data since BDW data may conflict with mine. Also, if get more BDW data, I want to be able to relate these data to that. A good way to keep data which are made at distinct times or which come from different, possibly conflicting, sources is using contexts. The code below shows how to do that:

```
>>> from rdflib import ConjunctiveGraph
>>> from owmeta_core.context import Context
>>> # from mymod import Widget # my own OWM widget model
>>> # from bdw import Load # BigDataWarehouse API

>>> # Create a Context with an identifier appropriate to this BDW data import
>>> ctx = Context('http://example.org/data/imports/BDW_Widgets_2017-2018')
>>> ctx.mapper.process_class(Widget)

>>> # Create a context manager using the default behavior of reading the
>>> # dictionary of current local variables
>>> with ctx(W=Widget) as c:
...     for record in Load(data_set='Widgets2017-2018'):
...         # declares Widgets in this context
...         c.W(part_number=record.pnum,
...              fullness=record.flns,
...              hardness=record.hrds)
Widget(ident=rdflib.term.URIRef(...))

>>> # Create an RDFLib graph as the target for the data
>>> g = ConjunctiveGraph()

>>> # Save the data
>>> ctx.save(g)

>>> # Serialize the data in the nquads format so we can see that all of our
>>> # statements are in the proper context
>>> print(g.serialize(format='nquads').decode('UTF-8'))
<http://example.org/BDW/entities/Widget#12> <http...> <http://example.org/data/
↪ imports/BDW_Widgets_2017-2018> .
<http://example.org/BDW/entities/Widget#12> <...>
```

If you've worked with lots of data before, this kind of pattern should be familiar. You can see how, with later imports, you would follow the naming scheme to create new contexts (e.g., `http://example.org/data/imports/`

BDW_Widgets_2018-2019). These additional contexts could then have separate metadata attached to them or they could be compared:

```
>>> len(list(ctx(Widget)().load()))
1
>>> len(list(ctx18(Widget)().load())) # 2018-2019 context
3
```

Context Metadata

Contexts, because they have identifiers just like any other objects, so we can make statements about them as well. An essential statement is imports: Contexts import other contexts, which means, if you follow owmeta semantics, that when you query objects from the importing context, that the imported contexts will also be available to query.

2.4 Software Versioning

The owmeta library follows the [semanitc versioning scheme](#). For the sake of versioning, the software interface consists of:

1. Extensions to the **owm** command line defined
2. All “public” definitions (i.e., those whose names do not begin with ‘_’) in the *owmeta* package, sub-packages, and sub-modules
3. The format of RDF data generated by subclasses of `owmeta_core.dataobject.DataObject` and defined in the *owmeta* package, sub-packages, and sub-modules
4. The API documentation for the *owmeta* package, sub-packages, and sub-modules

In addition, any changes to the packages released on PyPI mandates at least a patch version increment.

For Git, our software version control system, software releases will be represented as tags in the form `v$semantic_version` with all components of the semantic version represented.

2.4.1 Documentation versioning

The documentation will have a distinct version number from the software. The version numbers for the documentation must change at least as often as the software versioning since the relationship of the documentation to the software necessarily changes. However, changes `_only_` to the non-API documentation will not be a cause for a change to any of the components of the software version number. For documentation releases which coincide with software releases, the documentation version number will simply be the software version number. Any subsequent change to documentation between software releases will compel an increase in the documentation version number by one. The documentation version number for such documentation releases will be represented as `${software_version}+docs${documentation_increment}`.

2.5 Python Release Compatibility

All Python releases will be supported until they reach their official end-of-life, typically reported as “Release Schedule” PEPs (search “release schedule” on the [PEP index](#)) Thereafter, any regressions due to dependencies of owmeta dropping support for an EOL Python version, or due to a change in owmeta making use of a feature in a still-supported Python release will only be fixed for the sake of OpenWorm projects when requested by an issue on [our tracker](#) or for other projects when a compelling case can be made.

This policy is intended to provide support to most well-maintained projects which depend on owmeta while not over-burdening developers.

3.1 Testing in owmeta

3.1.1 Preparing for tests

Within the owmeta project directory, owmeta can be installed for development and testing like this:

```
pip install --editable .
```

The project database should be populated like:

```
owm clone https://github.com/openworm/OpenWormData.git
```

3.1.2 Running tests

Tests should be run via setup.py like:

```
pytest
```

you can pass options to pytest like so:

```
pytest -k ChannelTest
```

3.1.3 Writing tests

Tests are written using Python's unittest. In general, a collection of closely related tests should be in one file. For selecting different classes of tests, tests can also be tagged using pytest marks like:

```
@pytest.mark.tag
class TestClass(unittest.TestCase):
    ...
```

Currently, marks are used to distinguish between unit-level tests and others which have the `inttest` mark. All marks are listed in `pytest.ini` under ‘markers’.

3.1.4 Data Bundle Tests

The tests in `DataIntegrityTest.py` require that the `openworm/owmeta-data` bundle is installed. Normally, these will run in the CI environment. If you are doing work that affects what goes in the bundle, you can install new versions of the bundle and run the tests with `pytest -m data_bundle`.

3.2 Adding documentation

Documentation for `owmeta` is housed in two locations:

1. In the top-level project directory as `INSTALL.md` and `README.md`.
2. As a [Sphinx](#) project under the `docs` directory

By way of example, to add a page about useful facts concerning *C. elegans* to the documentation, include an entry in the list under `toctree` in `docs/index.rst` like:

```
worm-facts
```

and create the file `worm-facts.rst` under the `docs` directory and add a line:

```
.. _worm-facts:
```

to the top of your file, remembering to leave an empty line before adding all of your wonderful worm facts.

You can get a preview of what your documentation will look like when it is published by running `sphinx-build` on the `docs` directory:

```
sphinx-build -w sphinx-errors docs build_destination
```

The docs will be compiled to html which you can view by pointing your web browser at `build_destination/index.html`. If you want to view the documentation locally with the [ReadTheDocs](#) theme you’ll need to download and install it.

3.2.1 API Documentation

API documentation is generated by the Sphinx [autodoc](#) extension. The format should be easy to pick up on, but a reference is available [here](#). Just add a docstring to your function/class/method and add an `automodule` line to `owmeta/__init__.py` and your class should appear among the other documented classes.

3.2.2 Substitutions

Project-wide substitutions can be (conservatively!) added to allow for easily changing a value over all of the documentation. Currently defined substitutions can be found in `conf.py` in the `rst_epilog` setting. [More about substitutions](#)

3.2.3 Conventions

If you'd like to add a convention, list it here and start using it. It can be reviewed as part of a pull request.

1. Narrative text should be wrapped at 80 characters.
2. Long links should be extracted from narrative text. Use your judgement on what 'long' is, but if it causes the line width to stray beyond 80 characters that's a good indication.

3.3 owmeta coding standards

Pull requests are *required* to follow the PEP-8 Guidelines for contributions of Python code to owmeta, with some exceptions noted below. Compliance can be checked with the `pep8` tool and these command line arguments:

```
--max-line-length=120 --ignore=E261,E266,E265,E402,E121,E123,E126,E226,E24,E704,E128
```

Refer to the [pep8 documentation](#) for the meanings of these error codes.

Lines of code should only be wrapped before 120 chars for readability. Comments and string literals, including docstrings, can be wrapped to a shorter length.

Some violations can be corrected with `autopep8`.

CHAPTER 4

Issues

CHAPTER 5

Indices and tables

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

O

- `owmeta`, 3
- `owmeta.bibtex`, 6
- `owmeta.bibtex_customizations`, 7
- `owmeta.biology`, 8
- `owmeta.cell_common`, 8
- `owmeta.channel`, 8
- `owmeta.channel_common`, 10
- `owmeta.channelworm`, 10
- `owmeta.cli_hints`, 11
- `owmeta.commands`, 3
- `owmeta.data_trans`, 3
 - `owmeta.data_trans.bibtex`, 4
 - `owmeta.data_trans.common_data`, 5
 - `owmeta.data_trans.context_merge`, 5
 - `owmeta.data_trans.data_with_evidence_ds`, 6
- `owmeta.document`, 11
- `owmeta.documentContext`, 12
- `owmeta.evidence`, 13
- `owmeta.experiment`, 14
- `owmeta.neuroml`, 15
- `owmeta.plot`, 15
- `owmeta.utils`, 16
- `owmeta.website`, 16
- `owmeta.worm_common`, 16

A

appearsIn (*owmeta.channel.Channel* attribute), 8
author (*owmeta.document.Document* attribute), 12
author () (in module *owmeta.bibtex_customizations*), 7

B

BaseDocument (class in *owmeta.document*), 11
bibtex_to_document () (in module *owmeta.bibtex*), 6
BibTexDataSource (class in *owmeta.data_trans.bibtex*), 4
BibTexDataTranslator (class in *owmeta.data_trans.bibtex*), 4
BiologyType (class in *owmeta.biology*), 8
blockers (*owmeta.channelworm.PatchClampExperiment* attribute), 10

C

Ca_concentration (*owmeta.channelworm.PatchClampExperiment* attribute), 10
cell (*owmeta.channelworm.PatchClampExperiment* attribute), 10
cell_age (*owmeta.channelworm.PatchClampExperiment* attribute), 10
Channel (class in *owmeta.channel*), 8
CHANNEL_RDF_TYPE (in module *owmeta.channel_common*), 10
ChannelModel (class in *owmeta.channelworm*), 10
Cl_concentration (*owmeta.channelworm.PatchClampExperiment* attribute), 10
combined_context_property (*owmeta.data_trans.data_with_evidence_ds.DataWithEvidenceDataSource* attribute), 6
conductance (*owmeta.channelworm.ChannelModel* attribute), 10
content (*owmeta.neuroml.NeuroMLDocument* attribute), 15
context_property (*owmeta.data_trans.bibtex.EvidenceDataSource* attribute), 5

ContextMergeDataTranslator (class in *owmeta.data_trans.context_merge*), 5
customizations () (in module *owmeta.bibtex_customizations*), 7

D

data_context_property (*owmeta.data_trans.data_with_evidence_ds.DataWithEvidenceDataSource* attribute), 6
DataWithEvidenceDataSource (class in *owmeta.data_trans.data_with_evidence_ds*), 6
date (*owmeta.document.Document* attribute), 12
defined_augment () (*owmeta.channel.Channel* method), 8
defined_augment () (*owmeta.channel.ExpressionPattern* method), 9
defined_augment () (*owmeta.document.Document* method), 12
defined_augment () (*owmeta.evidence.Evidence* method), 13
defined_augment () (*owmeta.website.Website* method), 16
description (*owmeta.channel.Channel* attribute), 8
description (*owmeta.channel.ExpressionPattern* attribute), 9
Document (class in *owmeta.document*), 11
document_url (*owmeta.neuroml.NeuroMLDocument* attribute), 15
DocumentContext (class in *owmeta.documentContext*), 12
DocumentContextMeta (class in *owmeta.documentContext*), 12
doi (*owmeta.document.Document* attribute), 12
doi () (in module *owmeta.bibtex_customizations*), 7

E

Evidence (class in *owmeta.evidence*), 13
evidence_context_property (*owmeta.data_trans.data_with_evidence_ds.DataWithEvidenceDataSource* attribute), 6

evidence_for() (in module *owmeta.evidence*), 14
 EvidenceDataSource (class in *owmeta.data_trans.bibtex*), 4
 EvidenceError, 13
 Experiment (class in *owmeta.experiment*), 14
 expression_pattern (*owmeta.channel.Channel* attribute), 9
 ExpressionPattern (class in *owmeta.channel*), 9

G

gating (*owmeta.channelworm.ChannelModel* attribute), 10
 gene_class (*owmeta.channel.Channel* attribute), 9
 gene_name (*owmeta.channel.Channel* attribute), 9
 gene_WB_ID (*owmeta.channel.Channel* attribute), 9
 get_conditions() (*owmeta.experiment.Experiment* method), 14
 get_data() (*owmeta.plot.Plot* method), 16

H

HomologyChannelModel (class in *owmeta.channelworm*), 10

I

identifier_augment() (*owmeta.channel.Channel* method), 8
 identifier_augment() (*owmeta.channel.ExpressionPattern* method), 9
 identifier_augment() (*owmeta.document.Document* method), 12
 identifier_augment() (*owmeta.evidence.Evidence* method), 13
 identifier_augment() (*owmeta.website.Website* method), 16
 initial_voltage (*owmeta.channelworm.PatchClampExperiment* attribute), 11
 input_type (*owmeta.data_trans.bibtex.BibTexDataTranslator* attribute), 4
 ion (*owmeta.channelworm.ChannelModel* attribute), 10
 ion_channel (*owmeta.channelworm.PatchClampExperiment* attribute), 11

L

listify() (in module *owmeta.bibtex_customizations*), 7
 listify_one() (in module *owmeta.bibtex_customizations*), 7
 load() (in module *owmeta.bibtex*), 6
 load_from_file_named() (in module *owmeta.bibtex*), 6
 loads() (in module *owmeta.bibtex*), 6

M

membrane_capacitance (*owmeta.channelworm.PatchClampExperiment* attribute), 11
 model (*owmeta.channel.Channel* attribute), 9
 models (*owmeta.channel.Channel* attribute), 9
 modelType (*owmeta.channelworm.ChannelModel* attribute), 10
 mutants (*owmeta.channelworm.PatchClampExperiment* attribute), 11

N

name (*owmeta.channel.Channel* attribute), 9
 neuroML (*owmeta.channelworm.ChannelModel* attribute), 10
 NeuroMLDocument (class in *owmeta.neuroml*), 15
 NeuroMLProperty (class in *owmeta.neuroml*), 15
 note_url() (in module *owmeta.bibtex_customizations*), 8

O

output_type (*owmeta.data_trans.bibtex.BibTexDataTranslator* attribute), 4
 output_type (*owmeta.data_trans.context_merge.ContextMergeDataTranslator* attribute), 5
 owmeta (module), 3
 owmeta.bibtex (module), 6
 owmeta.bibtex_customizations (module), 7
 owmeta.biology (module), 8
 owmeta.cell_common (module), 8
 owmeta.channel (module), 8
 owmeta.channel_common (module), 10
 owmeta.channelworm (module), 10
 owmeta.cli_hints (module), 11
 owmeta.commands (module), 3
 owmeta.data_trans (module), 3
 owmeta.data_trans.bibtex (module), 4
 owmeta.data_trans.common_data (module), 5
 owmeta.data_trans.context_merge (module), 5
 owmeta.data_trans.data_with_evidence_ds (module), 6
 owmeta.document (module), 11
 owmeta.documentContext (module), 12
 owmeta.evidence (module), 13
 owmeta.experiment (module), 14
 owmeta.neuroml (module), 15
 owmeta.plot (module), 15
 owmeta.utils (module), 16
 owmeta.website (module), 16
 owmeta.worm_common (module), 16

P

parse_bibtex_into_documents() (in module

[owmeta.bibtex](#), 7
[patch_type \(owmeta.channelworm.PatchClampExperiment attribute\)](#), 11
[PatchClampChannelModel \(class in owmeta.channelworm\)](#), 10
[PatchClampExperiment \(class in owmeta.channelworm\)](#), 10
[pipette_solution \(owmeta.channelworm.PatchClampExperiment attribute\)](#), 11
[Plot \(class in owmeta.plot\)](#), 15
[pmid \(owmeta.document.Document attribute\)](#), 12
[proteins \(owmeta.channel.Channel attribute\)](#), 9
[PubMedRetrievalException](#), 11
[wormbaseid \(owmeta.channel.ExpressionPattern attribute\)](#), 9
[wormbaseid \(owmeta.document.Document attribute\)](#), 12
[WormbaseRetrievalException](#), 11
[wormbaseURL \(owmeta.channel.ExpressionPattern attribute\)](#), 9
[year \(owmeta.document.Document attribute\)](#), 12

Q

[query_context \(\) \(in module owmeta.evidence\)](#), 14

R

[reference \(owmeta.evidence.Evidence attribute\)](#), 14
[reference \(owmeta.experiment.Experiment attribute\)](#), 14
[refutes \(owmeta.evidence.Evidence attribute\)](#), 14

S

[set_data \(\) \(owmeta.plot.Plot method\)](#), 16
[subfamily \(owmeta.channel.Channel attribute\)](#), 9
[supports \(owmeta.evidence.Evidence attribute\)](#), 14

T

[title \(owmeta.document.Document attribute\)](#), 12
[title \(owmeta.website.Website attribute\)](#), 16
[translate \(\) \(owmeta.data_trans.bibtex.BibTexDataTranslator method\)](#), 4
[translate \(\) \(owmeta.data_trans.context_merge.ContextMergeDataTranslator method\)](#), 5

U

[update_from_wormbase \(\) \(owmeta.document.Document method\)](#), 12
[uri \(owmeta.document.Document attribute\)](#), 12
[url \(owmeta.website.Website attribute\)](#), 16
[url \(\) \(in module owmeta.bibtex_customizations\)](#), 8

V

[value_type \(owmeta.neuroml.NeuroMLProperty attribute\)](#), 15

W

[wbid \(owmeta.document.Document attribute\)](#), 12
[Website \(class in owmeta.website\)](#), 16
[wormbaseID \(owmeta.channel.ExpressionPattern attribute\)](#), 9